ActuarAI: Machine Learning Models for Patient Disease Forecasting and Representation

David Spencer Kartchner

A thesis submitted to the faculty of
Brigham Young University
in partial fulfillment of the requirements for the degree of

Master of Science

Jeffrey Humpherys, Chair
Tyler Jarvis
David Wingate

Department of Mathematics

Brigham Young University

ABSTRACT

ActuarAI: Machine Learning Models for Patient Disease Forecasting and Representation

David Spencer Kartchner
Department of Mathematics, BYU
Master of Science

Many strategies exist for representing patients and their medical histories contained in electronic health records (EHRs). These records tell a story of a patient's medical state across time. Accordingly, many have sought to mathematically capture that story by representing diseases, patient visits, and even a patient's internal health state in interpretive and predictive ways. This is particularly useful when we can encode a patient's medical state in such a way that it can predict his or her medical future of an individual, as this could allow high-impact health problems to be avoided or at least diagnosed in earlier states, thus mitigating many of the adverse effects. This thesis contributes to the current disease prediction literature in two main ways:

- We present a unified, representative review of methods for representing diseases, visits, and patients, along with published metrics for measuring their success. We discuss these in the context of natural language processing (NLP), from which many of these algorithms were adapted.
- We discuss other machine learning approaches to disease prediction that seem to have been disregarded in recent literature and demonstrate that simple models such as gradient boosted trees can outperform even advanced disease prediction deep learning models without needing to pre-learn disease representations.

Keywords: Machine Learning, Deep Learning, Health Care, Insurance Claims

Acknowledgments

Special thanks to Jeffrey Humpherys for the regular research meetings in which so many of these ideas developed and matured. Also, thanks to Andy Merrill for initially exposing me to disease-related problems in healthcare and for helping get IRB approval on many pieces of this thesis.

Contents

List of Tables

List of Figures

Chapter 1. Introduction: Predictive Modeling with Insurance Claims Data

Healthcare outcomes have a dramatic impact on the physical, social, and economic well-being of a society. Healthcare costs in the United States are currently at record highs and constitute more than 17% of national expenditures. U.S. healthcare spending as a percentage of gross domestic product (GDP) is approximately double that of other Organization for Economic Cooperation and Development (OEDC) member countries and has consistently grown faster than U.S. GDP over the past 60 years [4].

The burden of healthcare costs becomes more dramatic when one recognizes that these costs rest upon the shoulders of a small number of individuals. According to a 2015 study by the U.S. Government Accountability Office, nearly 50% of the U.S. healthcare expenditures come from the top 5% of patients[5]. Minorities and those who are economically or educationally disadvantaged are at disproportionate risk for developing chronic health problems [6, 7].

The majority of healthcare costs revolve around chronic diseases. In a 2014 study, the Agency for Healthcare Research and Quality (AHRQ) found that chronic conditions account for 86% of U.S. healthcare expenditures. Moreover, chronic diseases account for seven of the top 10 causes of death in the United States with the top two, heart disease and cancer, accounting for 46% by themselves [8]. A 2012 study found that 117 million - about half - of US adults live with one or more chronic conditions [9]. The risk and severity of these conditions further increases with age. Accordingly, various individuals have called for personalized efforts to preemptively identify and prevent chronic conditions [10, 11], assuming that such would reduce the suffering for individuals while reducing cost and uncertainty for providers. We discuss different types of data underlying healthcare AI research, some of the key problems in healthcare , and examples of current related research.

1.1 Medical Data

1.1.1 Medical Images.

Diagnostic Scans. Many medical screening and diagnostic procedures produce images. X-rays, ultrasounds, MRIs, CAT scans, PET scans, and blood tests all produce images which usually require expert analysis to interpret. A number of recent initiatives have focused on using medial imaging in conjunction with machine learning to improve diagnosis of many diseases, including skin cancer [12], lung cancer [13], Alzheimer's disease [14], malaria [15], and Parkinson's disease [16].

Microscopy Images. Many diseases originate at the cellular level and are related to genetic defects of malfunctions. Cancer, for instance, is strongly linked to failure of signaling for apoptosis [17], which leads to the accumulation of genetic defects and increased replication of defected cells, ultimately leading to tumor development. Malaria is caused by a parasite infiltrating host cells and using them to both reproduce and hide from the immune system. Viruses similarly infiltrate cells, taking over cellular machinery to block the expression of host genes and substitute their own DNA to proliferate. Finally, a wide variety of rare genetic diseases are associated with particular genetic mutations, including cystic fibrosis, sickle cell disease, muscular dystrophy, and Huntington disease.

Since so many diseases occur at an intracellular level, a number of companies and research initiatives are focused on obtaining a visual understanding of biology at a cellular level. Such initiatives include identifying and segmenting individual pieces of cellular machinery (e.g. nuclei or mitochondria [18, 19]) in molecular images, identifying disease phenotypes from medical images, and predicting drug-induced gene expression from previously imaged cells [20]. Such procedures rely on cell imaging, which involves staining portions of cells via dyes that bind to particular proteins, thus identifying cell walls, mitochondria, ribosomes, and other sub-cellular entities [21]. Cellular imaging has also shown potential to drastically improve the performance of drug hit prediction, thus substantially reducing time and cost of developing new medications.

1.1.2 Insurance Data. As almost all medical care in the developed world is paid for at least in part by a third party payer, insurance companies hold vast amounts of summary data on individuals primarily in the form of insurance claims. Each claim contains a wide range of variables about an individual health care encounter. Since this work focuses primarily on prediction of cost and disease onset from claims data, we discuss each of these variables in turn:

Diagnosis and Procedure Codes. Medical problems and actions taken to treat those problems are identified using taxonomies of codes. In the United States, diagnosis codes usually come from the International Classification of Diseases, 9th Revision (ICD-9), [22] or 10th Revision (ICD-10) [23]. These differ primarily in that ICD-10 represents diseases at a more granular level with more possibilities to detail related comorbidities and specific disease circumstances. ICD-10 contains over 70,000 different diagnoses, including a few humorous examples such as:

- W22.02: Walked into lamppost
- Z63.1: Problems in relationship with in-laws
- W61.43: Pecked by a turkey
- V97.33XD: Sucked into jet engine, subsequent encounter

As the name suggests, procedure codes constitute medical action taken during an encounter. On a given claim, diagnoses given provide medical justification for particular procedures, each of which is billable. These procedures could range from an ultrasound or office examination to a surgery or organ transplant. Procedures are usually coded under a taxonomy corresponding to ICD-9 or ICD-10, or according to the Current Procedural Terminology (CPT) taxonomy developed by the American Medical Association.

Drug Codes. Drug codes identify prescriptions filled by individuals and generally encapsulate drug type, function, dosage, number of doses, and form of delivery (e.g. capsule,

injection, patch, etc). National Drug Codes (NDCs) also contain information about which pharmaceutical firm manufactured a particular drug. Other types of drug coding systems include Generic Product Identifiers (GPIs) and the WHO's Anatomical Therapeutic Chemical (ATC) Classification System.

Other Claims Variables. In addition to codes describing the type of care administered, insurance claims also contain information on billed cost of each procedure (these costs are usually standardized), information regarding the facility at which the visit took place, and whether care occurred in an emergency department (ED), inpatient, outpatient, or clinical setting. These variables can be pertinent to tasks such as individualized risk adjustment and assessing the extent to which individuals are seeking preventative care (based on whether their encounters occur in primarily inpatient or outpatient settings).

Insurance providers generally maintain data about patient demographics. Many diseases disproportionately affect particular ethnic groups and most chronic diseases increase in prevalence with advancing age. Demographic context can therefore be critical to accurately interpret the severity of various diagnoses and assess the risk of various conditions.

1.1.3 Lab Values and Vital Signs. Binary diagnostic values are often obtained by means of continuous measurements of various vital signs or biochemical values. Diabetes, for instance, is diagnosed by requiring an individual to fast for 24 hours then drink a sugary liquid. A positive diagnosis corresponds to blood sugar in excess of 200 mg/dL after two hours. Similarly, ketoacidosis, a complication of diabetes caused by toxic levels of ketones in the blood, is identified by measuring the ketone levels in blood and urine. Chronic Kidney Disease is similarly diagnosed by measuring protein levels in blood and urine to determine the effectiveness of filtration. Hypertension is defined simply as having repeated blood pressure measurements in excess of 140/90. Because lab values and vital signs are continuous, they provide a much more nuanced picture into an individual's internal health state than diagnosis, procedure, and drug codes alone. Lab values, however, can be difficult to use because they are usually quite sparse. Some have sought to overcome the barrier of

sparsity by imputing lab values via kernel regression [24]. Others have used continuous vital sign monitors to predict the severe adverse events (e.g. events leading to cardiac arrest, unexpected admission to ICU, death, or disability) [25, 26] and the onset of neonatal sepsis [27].

1.1.4 Data Hierarchies and Taxonomies. A number of ontologies exist which hierarchically group medical codes by their general types of physiological conditions. One of the most commonly used is the Clinical Classification Software (CCS) system constructed by the Agency for Healthcare Research and Quality (AHRQ). Many machine disease prediction and medical code embedding frameworks use CCS to add additional relational information to models and to validate results. Constructed for ICD-9 diagnosis and procedure codes, CCS is designed to collapse the high granularity of ICD-9 into clinically relevant categories. CCS has both a is built as a four-level hierarchy. Diagnoses and procedures are respectively divided into 18 and 16 major high-level categories, with subcategories branching out in a tree beneath each. CCS also offers a single-level groupings which are formed by simply selecting particular nodes from the tree and lumping all of their descendant codes into a single category. An example of the CCS hierarchy corresponding to infectious disease is shown in Figure 1.1

1.2 Problems in Healthcare

1.2.1 Diagnostics. Treatment in healthcare settings begins with a diagnosis. Once a problem is diagnosed, treatment can be prescribed to mitigate the associated condition. The increasing volume of healthcare data contained in Electronic Health Records (EHRs) has caused many to consider the possibility of designing automated clinical support and disease detection systems based on patient history and risk factors [10]. A major focus in healthcare ML is designing tools both to improve speed, accuracy, and cost current diagnosis for patients, as well as predicting diagnosis to identify patients in need of preventative care.

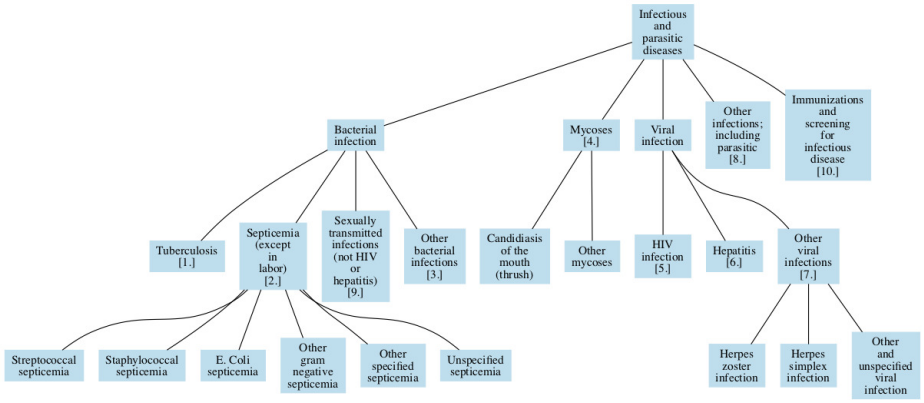


Figure 1.1: CCS hierarchy for infectious diseases. When a node corresponds to particular single-level CCS category, at category is given in brackets after the node label.

Current Diagnosis. Many diagnostic machine learning models revolve around automating diagnosis based on medical test results. For example, [15] develops a method for field cell microscopy images to efficiently and inexpensively identify malaria with limited testing facilities while [12] utilizes deep learning to diagnose skin cancer from images of potentially affected regions, achieving dermatologist -level accuracy. One of the most compelling cases of automated diagnosis is that of multiple-instance learning (MIL), which seeks to give a single diagnostic label to a bag of instances of patient data. Possible sources of this bag of instances include medical imaging, where MRIs, CT scans, and mammograms [28], as well as cell microscopy images [29, 30]. In each case, MIL allows a learning algorithm to look at multiple images that combine to deliver a single diagnosis. Recent work combines neural attention with MIL to identify which entities most contributed to the given diagnosis, making results interpretable and verifiable by medical experts.

Machine learning can additionally be used to assist physicians in making accurate diagnoses. Examples include [31], which employs deep learning on microscopy images of stained blood cells to segment cancerous cells from normal cells for faster diagnosis, and [14], which uses support vector machines on preprocessed MRI scans to improve diagnosis of Alzheimer's disease. Assistance can also support effective differential diagnosis, which is the process of

identifying a patient's underlying disease when there are multiple possible outcomes for a particular condition [2, 32].

Computer aided diagnostic tools can be a double edged sword. As diagnostic tools become increasingly robust, physicians run the risk of becoming less attentive to important patient details, assuming that computers will pick up the areas of concern. It is important to be transparent about the limitations of ML diagnostic models to promote thoughtful integration into diagnostic workflow rather than blind trust.

Predictive Diagnosis and Prognosis. Predictive diagnosis consists of identifying patients at high risk for developing high-impact conditions (usually chronic diseases) prior to actual diagnosis. Predictive prognosis is the related task of predicting a patient's disease progression and the occurrence of complications. A number of past studies have attempted to use patient laboratory tests [24, 33, 34], diagnoses [24, 32, 35, 36, 37, 38], and medications [32] as means of predicting disease onset. One advantage of such models is the ability to empirically learn predictors of disease risk from millions of patient histories, rather than relying on previously identified risk factors. Models as simple as an L_1 regularized logistic regression have been effective in identifying unknown potential risk factors [38] while simultaneously improving sensitivity and specificity of risk forecasting.

1.2.2 Risk Adjustment.

Risk adjustment is the process of predicting future patient costs, usually with the objective to modify premiums or capitation payments to accurately reflect the cost of individuals. Since healthcare costs are dominated by the top decile of patients, risk adjustment models typically rank individuals by probability of becoming (and remaining) high cost. Traditionally, risk adjustment models focused on patient demographic data to stratify cost bins, with significant improvements from incorporating EHR data (usually in the form of binary indicators of various conditions or for EHR codes) [39, 40, 41, 42, 43, 44]. Most models in literature use basic linear algorithms at best for their predictions, though recent work has shown that combining past data with tree-based and gradient boosting models can substantially improve results [43]. Current risk adjustment

models seek interpretability by using linear models (e.g. logistic regression) to identify *current* conditions indicative of future cost. The next generation of these models will combine cost forecasting with disease prediction, so that models output both a cost ranking and the *future* conditions that will drive that cost. Data scientists have an obligation to ensure that such predictions are being used to benefit patients rather than to selectively deny care to those who need it most.

Chapter 2. Code Embeddings

2.1 Context

Vector space embeddings, or simply embeddings, are mathematical representations of concepts that encode meaning and relationships. Embeddings were originally developed for natural language processing (NLP) to encode similarity between words represent each word as a vector $w_i \in \mathbb{R}^N$. Most algorithms to obtain embeddings leverage the *disributional hypothesis* [45], which states that words that appear in similar contexts likely have similar meanings. For example, consider the following set of sentences:

- $\mathbb{R}^n$ is a field.
- $\mathbb{R}^n$ is a Banach space.
- $\mathbb{C}$ is a field and a Banach space.
- Banach spaces are complete, normed, linear spaces.

From the above, a good word embedding algorithm would conclude that $\mathbb{R}^n$ and are similar concepts and also that both are related to complete, normed, linear spaces (all of which are correct). Thus, these word embeddings should like close to each other in $\mathbb{R}^N$, where "closeness" is usually defined using either Euclidian distance or cosine similarity. Once good embeddings are obtained, related words can be found by a simple nearest neighbors search.

From a clinical perspective, we can obtain vector space embeddings of medical codes from claims data to similarly learn which diagnoses and procedures likely occur in related contexts. One would expect that to commonly co-occurring diagnoses would represent related conditions. Discovering meaningful embeddings for medical concepts is an ongoing topic of research in medical literature [2, 46, 1]. We thus discuss various embedding algorithms and propose a framework for determining how well each embeds medical concepts.

2.2 Terminology and Definitions

We introduce notation and definitions for commonly used NLP terminology, as well as intuition for its adaptation to a medical context:

- **Word:** An atomic unit of text data, denoted w_i. In written texts, these are actual words. Words are diagnosis, procedure, or drug codes in a medical context.
- **Sentence:** A subset of words related for a specific purpose. In our case, these would correspond to medical visits.
- **Document:** A collection of related sentences. Corresponds to the medical history of a single patient.
- **Corpus:** A collection of documents. Corresponds to the histories of our entire patient population.
- **Vocabulary:** The set $\mathcal{C} = \{w_1, w_2, \dots w_{|\mathcal{C}|}\}$ of all words in your corpus. In our case, this consists of the sets of all possible medical codes in our data.
- **N-Gram:** An tuple $T \in \mathcal{C}^N$ consisting of N contiguous words in a document. For example, in the sentence "The quick brown fox jumped over the lazy dog", the 3-grams are:
  - (The, quick, brown)
  - (quick, brown, fox)
  - (brown, fox, jumped)
  - (fox, jumped, over)
  - (jumped, over, the)
  - (over, the, lazy)
  - (the, lazy, dog)

2.3 Word Embedding Algorithms

We discuss a variety of word embedding algorithms, many of the techniques of which we will later apply to learning representations of medical concepts.

2.3.1 Word2Vec. Tomas Mikolov's Word2Vec [47, 48] made word embeddings mainstream by demonstrating that his embeddings conserved the natural intuition of human linguistic understanding. The embeddings were also state of the art at the time of publication and are still competitive five years later. We thus examine the two techniques presented by Mikolov for finding word vectors: skip-gram with negative sampling and continuous bag-of-words . We focus on SGNS, since this seems to be the most popular. Our explanation borrows intuition from [49].

Skip-Gram with Negative Sampling. Skip-gram with negative sampling (SGNS) learns word embeddings by choosing embeddings for a center word $w_i \in \mathcal{C}_w$ and its contexts $c_{w,i} \in \mathcal{C}_c$, where $\mathcal{C}_w$ is the set of words to embed and $\mathcal{C}_c$ is the set of contexts for these words. Though $\mathcal{C}_w$ and $\mathcal{C}_c$ will have the same words, it is useful to consider them separately because we obtain different separate embeddings when words and contexts. The context of w consists of all of the words in a symmetric window of L words on either side of w, thus creating a 2L-gram that "skips" w. Let $C(w)$ be the set of all context words of w and let D be the collection of all word-context pairs, i.e.

$$D = \{(w_i, w_{i+l}) : l \in \mathbb{Z} \cap [-L, L], l \neq 0, 0 \leq i \leq |\mathcal{C}_w|\} \tag{2.1}$$

If we let θ be the parameters of our model (i.e. our word and context embedding vectors),

then we seek to choose θ to optimize the following cost function:

$$\theta = \underset{\theta}{\text{argmax}} \prod_{w \in Corpus} \Big[\prod_{c \in C(w)} p(c|w;\theta) \Big] \tag{2.2}$$

$$= \underset{\theta}{\text{argmax}} \prod_{(w,c) \in D} N_{(w,c)} p(c|w;\Theta) \tag{2.3}$$

where $N_{(w,c)}$ is the number of times the word-context pair (w, c) appears in the corpus. $p(c|w;\theta)$ is parameterized by the softmax function:

$$p(c|w;\theta) = \frac{e^{v_w \cdot v_c}}{\sum_{c' \in \mathcal{C}_c} e^{v_w \cdot v'_c}} \tag{2.4}$$

Substituting equation 2.4 into 2.3 and taking the log gives the following objective function:

$$\underset{\theta}{\text{argmax}} \sum_{(w,c) \in D} \Big(v_w \cdot v_c - log(\sum_{c' \in \mathcal{C}_c} e^{v_w \cdot v'_c} \Big) \tag{2.5}$$

This objective gives good embeddings in the sense that similar words should have similar vectors. Unfortunately, a corpus size can easily be in the billions with hundreds of thousands of words in the vocabulary, making this infeasible to compute. Accordingly, we heuristically modify the objective function to learn to differentiate between word-context from the training data and pairs that are not. Thus, for each given pair $(w, c) \in D$, we sample k contexts according to the distribution $\frac{p(c|Corpus)^{.75}}{C}$, where $p(c|Corpus)$ is simply the unigram distribution, (i.e. $p(c|Corpus) = \frac{N_c}{|Corpus|}$, where N_c is the number of times that the context c appears in our corpus) and C is a normalization constant. While one could simply use the unigram distribution to choose negative context words, the exponent causes us to up-sample rare words, thus ensuring that uncommon words are sufficiently represented to obtain good representations. Let $p(D = 1|w, c;\theta) = \frac{1}{1+e^{-v_w \cdot v_c}}$ be the probability that a word-context pair (w, c) is in the training data given our word embeddings, thus making $p(D = 0|w, c;\theta) = \frac{1}{1+e^{v_w \cdot v_c}}$ the probability that the context for word w was chosen randomly.

Our new optimization problem thus becomes:

$$\theta = \underset{\theta}{\text{argmax}} \prod_{(w,c)\in D} p(D=1|w,c;\theta) \prod_{(w,c')\in D'} p(D=0|w,c';\theta) \tag{2.6}$$

$$= \underset{\theta}{\text{argmax}} \sum_{(w,c)\in D} log(\frac{1}{1+e^{-v_w \cdot v_c}}) + \sum_{(w,c')\in D'} log(\frac{1}{1+e^{v_w \cdot v'_c}}) \tag{2.7}$$

Where D' is a collection of "negative" word-context pairs, sampled k times for each word w in our corpus. This objective function is then optimized via some variant of stochastic gradient descent. Once the optimization has converged, context embeddings $\{v_c\}$ are discarded and the each word is assigned its word embedding v_w. Though this model is heuristically driven, it has been show to produce good results in practice. Proposed modifications to improve performance include further down-sampling frequent words, using a dynamically-sized context window, where the window size is drawn uniformly from each word from $\{1, 2, \cdots, L\}$, and assigning each word the concatenation of its word and context embeddings.

2.3.2 GloVe. GloVe is a method originally developed for finding vector space word embeddings based on word context (i.e. co-occurrence with other words)[50]. GloVe learns to represent semantic meaning by considering words in our corpus pairwise and comparing the probability that each co-occurs with a given context word. To do so, define X_{ij} to be the number of times word j appears in the context of word i and $X_i = \sum_j X_{ij}$ to be the total number of times word i appears in the context of any word. GloVe seeks to find word vectors w_j and word context vectors $\tilde{w}_j$ that minimize the cost functional:

$$J = \sum_{i,j=1}^{\mathcal{C}} f(X_{ij})(w_i^T \tilde{w}_j + b_i + \tilde{b}_j - log(X_{ij}))^2 \tag{2.8}$$

which is essentially a weighted least squares problem. The pieces of J are as follows:

(i) The weighting function f is a non-decreasing, bounded function that increases the weight of frequent co-occurrences while down-weighting infrequent co-occurrences. Moreover, f is chosen to be relatively small for large x so that frequent words are not excessively overweighted. In practice, f is chosen to be

$$f = \begin{cases} (x/x_{max})^{\alpha} & x \leq x_{max} \\ 1 & x > x_{max} \end{cases}$$

where α=0.75 and $x_{max} = 100$ are noted to work well empirically by the authors.

(ii) w_i and $\tilde{w}_j$ are our word vector and context vector representations, respectively, with respective biases (intercepts) b_i and $\tilde{b}_j$. These are the parameters we seek to learn with our model.

(iii) $log(X_{ij})$ comes from considering taking the log of the probabilities $P_{ij} = P(j|i) = \frac{X_{ij}}{X_i}$ and noting that the denominator X_i is independent of j and can thus be absorbed into the b_i.

The full derivation of equation 2.8 can be found in [50]. This equation is then minimized numerically using gradient descent, then yielding the desired word embeddings.

2.4 Medically Adapted Embedding Algorithms

EHR's are similar to corpora of text in many ways. Each patient's EHR can easily be thought of as a document of text in which individual medical codes are words and visits are sentences. While such a framework can produce reasonably good disease prediction results [51], such models discard much of the temporal relationship between codes, as well as demographic variables that lend critical context to an individual's observed diagnoses.

Choi et al. [1] notes that a high-quality medical embeddings should satisfy three criteria, to which we add two more:

(i) Embeddings should meaningfully incorporate as much EHR data as possible, including visit-level groupings, temporal relationships, and demographic information.

(ii) Learned embeddings should be interpretable, thus allowing them to be used by medical personnel outside of isolated prediction tasks.

(iii) Embedding algorithms should be efficient enough to process industry-scale datasets, consisting of hundreds of thousands of patients and millions of visits.

(iv) Embeddings should improve prediction accuracy across across a variety of tasks over naive one-hot or bag-of-codes based approaches to medical representation.

(v) Embeddings should be able to be constructed without expert hand-engineering, to allow easy adaptability to various medical coding systems, particularly since these systems are regularly updated.

We summarize various approaches to code medical code embedding and compare their relative strengths and weaknesses.

2.4.1 Naive Application of NLP word Embedding Algorithms. Medical records contain nuanced data that contain both discrete medical concepts (i.e. diagnosis, procedure, and medication codes), as well as demographic data and continuous valued health measurements (usually in the form of lab tests). Though these demographic and test data contain relevant information, many papers demonstrate that one can obtain meaningful predictive results by simply discarding these variables and naively applying GloVe or Skip-gram to an individual's temporally ordered sequence of medical codes [52, 53]. Patient representations are then obtained by simply summing the embedded representations of an individual's medical history. Though simple, E. Choi et al. demonstrate significant performance gains in predictive accuracy across a number of classifiers [53] while providing qualitative demonstrations that embeddings discover meaningful and subtle relations between related conditions. We perform our own experiments on such naively obtained embeddings as both interpretable

representations and as inputs to predictive models. The details of these experiments can be found in Chapter 5.

A number of authors make a number of simple modifications to these naive methods to accommodate the differences between medical record histories and normal corpora of text. Both Y. Choi et al. [46] and E. Choi et al. [1, 32] note that temporal relationships between medical codes are the primary determinant for relatedness between codes. Y. Choi also notes that particular diagnosis can occur multiple times in a particular time window and thus proposes removing duplicate codes within a window, though it is unclear whether this improves embeddings or degrades them by downplaying the importance of particularly salient codes. There also appears to be general disagreement regarding the length of temporal period to consider, with window size ranging from a single day to three months. In actuality, it is likely that short-term relationships are more highly related than distant ones and that a weighting function that decays with temporal distance could capture the benefits of both approaches.

2.4.2 Med2Vec. E. Choi et al. [1] develops a framework for jointly learning medical code and visit representations in an interpretive manner. We describe the assumptions used to learn each type of representation, then present the unified training procedure to learn both simultaneously.

Code embedding procedure. Following the five aforementioned criteria, Med2Vec creates nonnegative code embeddings where each dimension can be interpreted as a latent dimension of disease with magnitude along each dimension as a measure of severity. This is done by training a skip-gram model where the embedding matrix is constrained to be positive using the element-wise ReLU function, $ReLU(x) = max(x, 0)$. We describe the procedure for learning `Med2Vec` embeddings, presenting all derivations at the level of a single patient for simplicity and clarity of notation. Let $\mathcal{C}$ represent our vocabulary of codes and $W_c \in \mathbb{R}^{m \times |\mathcal{C}|}$ be the regular skip-gram weight matrix with $W_c' = ReLU(W_c)$ as our embedding matrix. Then given a sequence of visits $\mathcal{V}_1, \mathcal{V}_2, \ldots, \mathcal{V}_T$, we can learn individual code representations

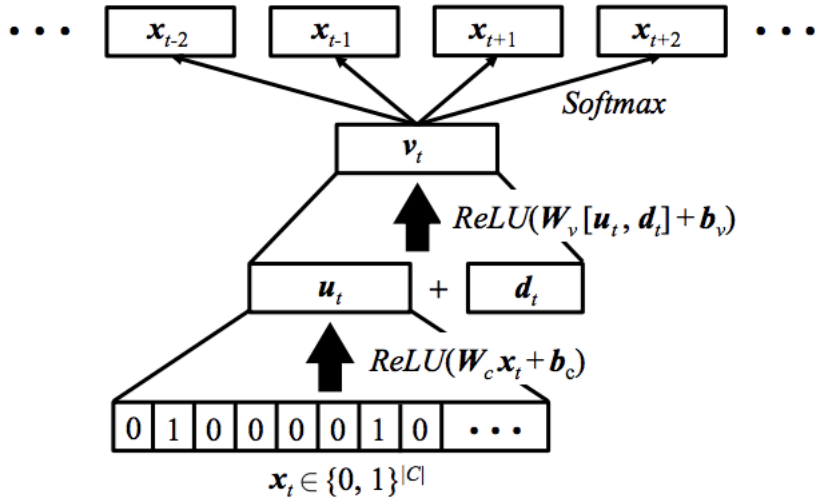


Figure 2.1: Diagram of `Med2Vec` architecture for learning visit-level embeddings, borrowed from [1]. Note that $\mathbf{x}_t$ is a multi-hot vector corresponding to the medical codes corresponding to the t^{th} visit and $\mathbf{d}_t$ is that patient's demographic information at the time of the t^{th} visit.

by maximizing the log likelihood:

$$\max_{W'_c} \frac{1}{T} \sum_{t=1}^{T} \sum_{i:c_i \in \mathcal{V}_t} \sum_{j:c_j \in \mathcal{V}_t, j \neq i} p(c_j|c_i)$$

where c_i is the i^{th} code in the visit $\mathcal{V}_t$. The probability $p(c_j|c_i)$ is defined using softmax as

$$p(c_j|c_i) = \frac{exp\big((\mathbf{W}'_c[:,j])^T(\mathbf{W}'_c[:,i])\big)}{\sum_{k=1}^{|\mathcal{C}|} exp\big((\mathbf{W}'_c[:,j])^T(\mathbf{W}'_c[:,i])\big)}$$

where $W'_c[:,j]$ denotes the j^{th} column of W'_c.

Visit embedding procedure. `Med2Vec` simultaneously learns to create visit-level representations of patients, as shown in Figure 2.1. For each patient, every visit $\mathcal{V}_t$ is represented as a multi-hot, binary vector $\mathbf{x}_t \in \{0,1\}^{|\mathcal{C}|}$, where $\mathbf{x}_t[i]$ (the i^{th} entry of $\mathbf{x}_t$) is 1 iff the i^{th} medical code $c_i \in \mathcal{V}_t$. The goal is to learn a mapping $f_v : \{0,1\}^{|\mathcal{C}|} \rightarrow \mathbb{R}^n$ that takes any possible visit to a real-valued, n-dimensional vector. This is done by first learning an intermediate representation $\mathbf{u}_t = ReLU(\mathbf{W}_c\mathbf{x}_t + \mathbf{b}_c)$, where $\mathbf{W}_c$ is the code embedding matrix given above before being passed through the $ReLU$ and $\mathbf{b}_c$ is a bias term.

This intermediate representation is then concatenated with the patient's demographic information at time t and the process is repeated to obtain a visit level representation $\mathbf{v}_t = ReLU(\mathbf{W}_v[\mathbf{u}_t, \mathbf{d}_t] + \mathbf{b}_v)$, where $[\cdot, \cdot]$ represents vector concatenation.

Each visit represents a state in a patient's continuously developing medical history. It is therefore reasonable to assume that a patient's visit at time t should be related to visits that preceded and followed it. Given a visit representation $\mathbf{v}_t$, we train a softmax classifier to predict visits in a context window of size l by minimizing the cross entropy error between actual and predicted visits:

$$\min_{\mathbf{W}_s, \mathbf{b}_s} \frac{1}{T} \sum_{t=1}^{T} \sum_{-l<i<l, i\neq 0} \big(- \mathbf{x}_{t+i}^T log(\hat{\mathbf{y}}_t) - (\mathbf{1} - \mathbf{x}_{t+i})^T (\mathbf{1} - log(\hat{\mathbf{y}}_t)) \big) \tag{2.9}$$

where $\mathbf{1}$ is the vector with all 1's and $\mathbf{W}_s$ and $\mathbf{b}_s$ are the weight matrix bias vectors, respectively, for the softmax classifier:

$$\hat{\mathbf{y}}_t = \frac{exp\big(\mathbf{W}_s \mathbf{v}_t + \mathbf{b}_s\big)}{\sum_{j=1}^{|\mathcal{C}|} exp\big(\mathbf{W}_s[j,:]\mathbf{v}_t + \mathbf{b}_s[j]\big)} \tag{2.10}$$

Joint Training. Code and visit representations can be learned more robustly by combining the loss functions and optimizing them simultaneously. This encourages our model to learn code representations that are useful for visit representations and for our visit representations to learn from global code co-occurrence statistics. We thus learn our embeddings by optimizing the following loss function:

$$\begin{aligned} \underset{\mathbf{W}_{c,v,s}, \mathbf{b}_{c,v,s}}{\operatorname{argmin}} \frac{-1}{T} \sum_{t=1}^{T} \Big\{ & \sum_{-l<i<l, i\neq 0} \big(\mathbf{x}_{t+i}^T \log(\hat{\mathbf{y}}_t) + (\mathbf{1} - \mathbf{x}_{t+i})^T (\mathbf{1} - \log(\hat{\mathbf{y}}_t)) \big) \\ & + \sum_{i: c_i \in V_t} \sum_{j: c_j \in V_t, j\neq i} p(c_j | c_i) \Big\} \end{aligned} \tag{2.11}$$

2.4.3 GRaph-based Attention Model (GRAM). GRAM [2] a framework for incorporating knowledge from human-derived medical ontologies to give a model context regarding which codes should naturally be grouped together based on medical similarity. Due to the

high granularity of medical codes, two diagnoses could have nearly the same physiological meaning, but would appear as orthogonal one-hot vectors when training a traditional embedding algorithm. ICD-9, ICD-10, and CCS are all ontologies of disease concepts containing hierarchies of disease concepts and thus representing different each condition at various levels of granularity (for an example of the CCS hierarchy for infectious disease, see Figure 1.1). Two codes that are close to each other in a medical ontology are likely similar in real life, allowing us to improve knowledge transfer between medical records. This has the additional benefits of improving the representation of codes that appear infrequently in medical records as well as reducing the data requirements to learn meaningful results, thus improving the predictive capacity of smaller providers.

Most medical ontologies are structured as directed acyclic graphs (DAGs) (in fact, ICD and CCS are sets of trees). Thus, let our ontology $\mathcal{G}$ be a DAG, where we assume the concept hierarchy is presented in the form of *parent-child* relationships, where nodes higher up the DAG represent more general concepts and leaf nodes represent the medical codes in our code vocabulary $\mathcal{C}$. We will refer to the ontology $\mathcal{G}$ as the *knowledge DAG*.

Let $\mathcal{C}' = \{c_{|\mathcal{C}|+1}, c_{|\mathcal{C}|+2}, \cdots c_{|\mathcal{C}|+N}\}$ be the set of all ancestors of the codes in $\mathcal{C}$, giving us the hierarchy DAG with nodes $\mathcal{D} = \mathcal{C} + \mathcal{C}'$. Note that each parent node in $\mathcal{C}'$ represents a more general medical concept than its children, a concept of which each of its children are specific instances. This allows our model to pay more attention to parents when little child data exists.

Code embeddings are learned in an end-to-end fashion using an attention mechanism, which learns the optimal convex combination of leaf node embeddings and parent node embeddings to represent each concept. A diagram of this process, borrowed from [2], is depicted in Figure 2.2. This allows code embeddings to be roughly learned from parents in the absence of adequate code-level data. Specifically, if we let $\mathcal{A}(i)$ be the indices of all of the ancestors of $c_i \in \mathcal{C}$ and let $\mathbf{e}_j$ be the embedding of each node $\mathbf{d}_j \in \mathcal{D}$, then the embedding

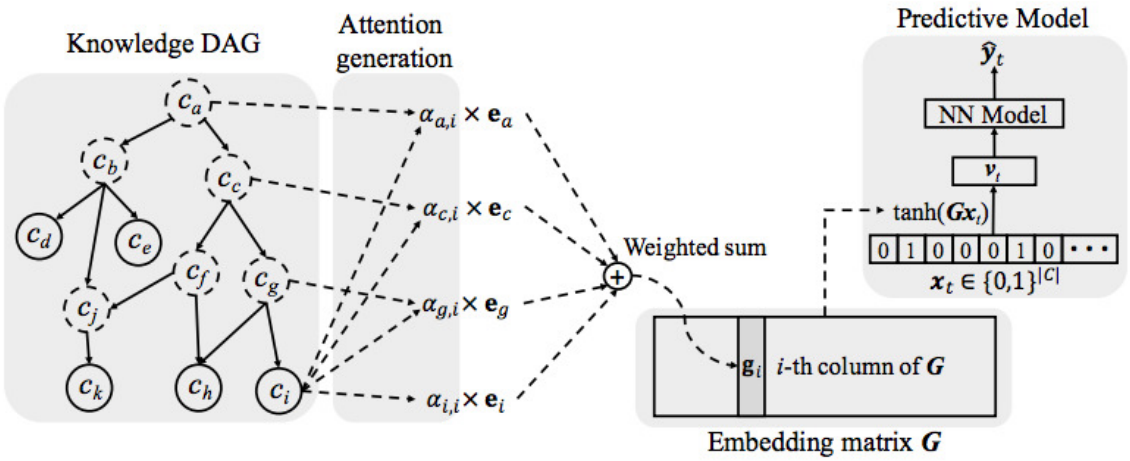


Figure 2.2: End-to-end GRAM architecture, as shown in [2]. Note that the attention mechanism represents each medical code by learning a convex combination of its initial embedding the the embeddings of its ancestors in the knowledge DAG. These embeddings are then used to parameterize visit representations for a predictive RNN model.

Φ_{c_i} is defined as the the convex combination:

$$\Phi_{c_i} = \sum_{j \in \mathcal{A}(i)} \alpha_{ij} \mathbf{e}_j \tag{2.12}$$

with the constraints

$$\sum_{j \in \mathcal{A}(i)} \alpha_{ij} = 1, \tag{2.13}$$

$$\alpha_{ij} \geq 0 \;\; \forall j \in \mathcal{A}(i) \tag{2.14}$$

The weights α_{ij} are computed via the softmax

$$\alpha_{ij} = \frac{exp\big(f(\mathbf{e}_i, \mathbf{e}_j)\big)}{\sum_{k \in \mathcal{A}(i)} exp\big(f(\mathbf{e}_i, \mathbf{e}_j)\big)} \tag{2.15}$$

where $f : \mathbb{R}^m \times \mathbb{R}^m \to \mathbb{R}$ if a function representing the compatibility between the preliminary node embeddings $\mathbf{e}_i$ and $\mathbf{e}_j$. We parameterize f by a single-layer feed-forward neural network

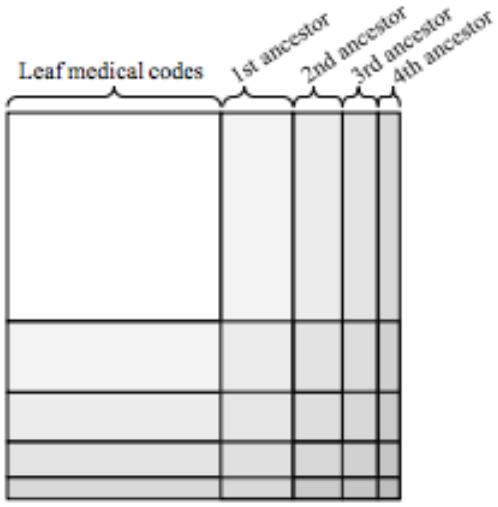


Figure 2.3: Co-occurrence matrix construction to pre-train node embeddings. Note that nodes further up in the knowledge DAG will co-occur with more things because they correspond to more concepts.

(i.e. a multi-layer perceptron) as:

$$f(\mathbf{e}_i, \mathbf{e}_j) = \mathbf{u}_a^T \tanh(\mathbf{W}_a \begin{bmatrix} \mathbf{e}_i \\ \mathbf{e}_j \end{bmatrix} + \mathbf{b_a}) \tag{2.16}$$

Here, $\mathbf{W}_a$ is the weight matrix for concatenation, $\mathbf{b}_a$ is a bias vector, and $\mathbf{u}_a$ is the weight vector for calculating the final scalar value.

In order to improve model performance and speed convergence, the embeddings $\mathbf{e}_j$ can be pretrained via GloVe, where each if two codes c_i and c_j co-occur, then each co-occurs with the other's ancestor nodes. Each of the ancestor nodes of c_i and c_j also co-occur with one another. A visualization of the construction, borrowed from [2], is given in Figure 2.3.

GRAM simultaneously trains the attention mechanism and a RNN that predicts the medical codes of the next visit to improve performance. Once embeddings Φ_{c_i} for each medical code c_i are obtained, they are concatenated into an embedding matrix $\mathbf{G} \in \mathbb{R}^{m \times |\mathcal{C}|}$, where m is the dimension of our embeddings. Visits are input as multi-hot vectors $\mathbf{x}_t \in$

$\{0,1\}^{|\mathcal{C}|}$ and visit representations $\mathbf{v}_t$ obtained using the embedding matrix $\mathbf{G}$ as

$$\mathbf{v}_t = \tanh(\mathbf{G}\mathbf{x}_t) \tag{2.17}$$

Let $\hat{\mathbf{y}}_t = \hat{\mathbf{x}}_{t+1}$ be the predicted multi-hot vector corresponding to a patient's $t+1^{th}$ visit. Then the entire visit prediction architecture can be described as:

$$\mathbf{v}_1, \mathbf{v}_2, \cdots, \mathbf{v}_t = \tanh(\mathbf{G}[\mathbf{x}_1, \mathbf{x}_2, \cdots, \mathbf{x}_t]) \tag{2.18}$$

$$\mathbf{h}_1, \mathbf{h}_2, \cdots, \mathbf{h}_t = \text{RNN}(\mathbf{v}_1, \mathbf{v}_2, \cdots, \mathbf{v}_t; \theta_r) \tag{2.19}$$

$$\hat{\mathbf{y}}_t = \hat{\mathbf{x}}_{t+1} = \text{Softmax}(\mathbf{W}\mathbf{h}_t + \mathbf{b}) \tag{2.20}$$

where θ_r parameterizes our RNN and $\mathbf{W}$ and $\mathbf{b}$ are the weight and bias for the softmax layer. The RNN used can be any RNN cell that avoids the vanishing gradient problem [54, 55], some of which are described in 3.5. Th RNN is then trained (along with the attention mechanism MLP) by minimizing the cross entropy loss for predicting the codes in the next visit:

$$\mathcal{L}(\mathbf{x}_1, \mathbf{x}_2, \cdots, \mathbf{x}_T) = \frac{1}{T-1} \sum_{t=1}^{T-1} \big(-\mathbf{y}_t^T \log(\hat{\mathbf{y}}_t) - (\mathbf{1} - \mathbf{y}_t)^T (\mathbf{1} - \log(\hat{\mathbf{y}}_t)) \big) \tag{2.21}$$

2.5 Embedding Evaluation

Because medical data is harder to come by and less mainstream than simple textual data, metrics for embeddings of medical concepts are less established than those for NLP tasks. However, a number of papers present both qualitative and quantitative evaluation measures, as follows:

2.5.1 Qualitative Evaluation. Qualitative embedding evaluations are by far the most common metric employed in the medical embedding literature. One good test for embedding goodness is to select random codes from the vocabulary, look at their nearest neighbors, and

see if those neighbors are, in fact, actually related to the chosen code. This evaluation is employed in [52, 2, 53, 46] and is useful in providing intuition of what medical relationships an algorithm is learning. While this method can provide good intuition, it is difficult to assess in practice because going through individual groupings is a time intensive process and requires substantial medical expertise.

2.5.2 Quantitative Evaluation. In [46], Y. Choi proposes two methods for quantifying the goodness of medical code embeddings. Both leverage physician-compiled taxonomies and relational systems and quantify how well embedded representations align with these groupings.

National Drug File Reference Terminology Relatedness. The National Drug File - Reference Terminology (NDF-RT) is a collection of information about drugs approved in the United States. NDF-RT contains information about drug ingredients, chemical structure, physiological effects, mechanism of action, dose form, pharmacokinetic properties (e.g. toxicity, how the drug interacts with various bodily systems), and diseases the drug can be used to treat or prevent. If one has both drug and diagnosis embeddings, this database can be used to see how well embeddings align with documented clinical knowledge. Let R define a relation in NDF-RT (i.e. either "may treat" or "may prevent") and let V be our set of medical concepts, where $V* \subset V$ is the subset of these concepts for which there is at least one pharmacological concept with a relation given in NDF-RT. Define a *seed pair* s to be a the directional vector between two concepts, for example $\Phi_{Tarceva} - \Phi_{LungCancer}$ (Tarceva is a drug commonly used to treat lung cancer). This allows us to quantify how well a medical parallel of linguistic analogical reasoning holds using the following relatedness *medical relatedness measure* (MRM) for a given seed pair and given neighborhood size k:

$$MRM_{NDF-RT}(V, R, k, s) = \frac{1}{|V*|} \sum_{v \in V*} 1_R\Big(\bigcup_{i=1}^{k} (v - s)(i)\Big) \tag{2.22}$$

where 1_R denotes the indicator function which returns 1 if any of the top k neighbors of the given medical concept are in the relation R and 0 otherwise. The notation $(v-s)(i)$ denotes the i^{th} neighbor of $v-s$ calculated after the subtraction $\Phi_v - \Phi_s$.

Noting that s is generally set to be the difference of two medical concepts, this measure closely parallels the analogical reasoning task introduced by Mikolov et al. in [56]. If $\Phi_s = \Phi_{Tarceva} - \Phi_{LungCancer}$ as described above and we set $\Phi_v = \Phi_{Revlimid}$ (note that Revlimid is a treatment for acute leukemia), then we would expect the following to hold:

$$\Phi_v - \Phi_s = \Phi_{LungCancer} - \Phi_{Tarceva} + \Phi_{Revlimid} \tag{2.23}$$

$$\approx \Phi_{AcuteLeukemia} \tag{2.24}$$

This is analogous to the famous $king - man + woman \approx queen$ from [56]. Producing embedding vectors that perform well on this task is desirable because it suggests that these vectors have dimensions of interpretable meaning, an important property noted in [57]. One can also set $s = 0$ to obtain the nearest the k nearest neighbors of the code in question.

Clinical Classification Software Relatedness. Clinical Classification Software (CCS) Relatedness is a measure of how well code embeddings group together in the CCS medical hierarchy introduced in section 1.1.4. Let G be a particular level of granularity, i.e. the deepest level of the multi-level CCS tree on which we allow leaf nodes to exist. Let $V(G) \subset V$ denote all ICD-9 codes in our vocabulary and k denote the number of neighbors to consider. Then we define the CCS Medical Relatedness Measure as

$$MRM_{CCS}(V, G, k) = \frac{1}{|V(G)|} \sum_{v \in V(G)} \sum_{i=1}^{k} \frac{1_G(v(i))}{log_2(i+1)} \tag{2.25}$$

where $v(i)$ is the i^{th} nearest neighbor of v and 1_G is the indicator of whether $v(i)$ belongs to the same group as v according to the given granularity G. Note that when $k = 1$, this corresponds to the proportion of codes whose top neighbors are in the same CCS group.

While this metric is well suited to be used with embeddings of insurance concepts, it is

limited in that it's score isn't improved by related conditions that belong in different categories (i.e. chronic kidney disease and heart failure both related to underlying hypertension). Moreover, procedure codes cannot be evaluated if CPT codes are used, as is common for many insurance datasets.

Evaluation based on Predictive Usefulness. Perhaps the most common means of evaluating embeddings is based on which performs best in the prediction task at hand. If the ultimate objective is to improve early diagnosis of chronic diseases, one should favor the embedding that is best suited to doing so. Our primary measure of quantitative embedding performance, therefore, will be how suited each is to early prediction of disease onset.

Chapter 3. Machine Learning Models

3.1 Notation

In this chapter, we assume that we have a set of feature vectors $\mathcal{D} = \{\mathbf{x}_1, \mathbf{x}_2, \cdots, \mathbf{x}_N\}$, where each $\mathbf{x}_i \in \mathbb{R}^n$. Each feature vector $\mathbf{x}_i$ has a corresponding label label y_i which determines to which class it belongs. Our goal is to find the best possible *classifier*, which is a function $f : \mathbb{R}^n \to \{0, 1\}$, such that f assigns the correct labels to as many of the vectors $\mathbf{x}_i \in \mathcal{D}$ as possible. We call each pair $(\mathbf{x}_i, y_i)$ a data point.

3.2 Random Forests

3.2.1 Decision Trees. A decision tree is a binary classifier that labels data by splitting using the data's features to split them into homogeneous subsets and labeling each subset according to the majority class in the subset. A decision tree begins with a single leaf node containing all of the data and then chooses a particular feature j and a splitting threshold τ which divide the data into two subsets: $S_r = \{\mathbf{x}_i : \mathbf{x}_i[j] \geq \tau\}$ and $S_l = \{\mathbf{x}_i : \mathbf{x}_i[j] < \tau\}$. These subsets respectively correspond to the subsets of data represented at the right and left child nodes of our initial node. The feature and threshold are chosen in a greedy manner at each subsequent node to optimize a predetermined metric of data impurity. Common choices include minimizing Gini impurity [58] and maximizing information gain.

Gini Impurity. To define Gini impurity, let our data be defined into K classes. Let t_p denote a node in our decision tree left and right child nodes t_l and t_r, respectively. The Gini

impurity $G(t_p)$ of a particular node is defined as:

$$G(t_p) = \sum_{i \neq j} p(i|t_p)p(j|t_p) \tag{3.1}$$

$$= \sum_{i=1}^{K} p(i|t_p) \sum_{j \neq i} p(j|t_p) \tag{3.2}$$

$$= \sum_{i=1}^{K} p(i|t_p)(1 - p(i|t_p)) \tag{3.3}$$

$$= 1 - \sum_{i=1}^{K} (p(i|t_p))^2 \tag{3.4}$$

where $p(i|t_p)$ is the conditional probability of class i given that we are in node t_p. The change in impurity for a given node split is therefore:

$$\Delta G(t_p) = -G(t_p) + P_r G(t_r) + P_l G(t_l) \tag{3.5}$$

$$= -\sum_{i=1}^{K} (p(i|t_p))^2 + P_r \sum_{i=1}^{K} (p(i|t_r))^2 + P_l \sum_{i=1}^{K} (p(i|t_l))^2 \tag{3.6}$$

P_r and P_l denote the proportion of the data in t_p in it's right and left children, respectively. Let $\mathcal{L}$ represent the set of all leaf nodes of our given tree $\mathcal{T}$. Then the total Gini impurity of $\mathcal{T}$ is given by:

$$G(\mathcal{T}) = \sum_{\ell \in \mathcal{L}} P_\ell G(t_\ell) \tag{3.7}$$

where P_ℓ is the proportion of all of the data in the leaf node ℓ. threshold.

Information Gain. Maximizing information gain is equivalent iteratively choosing the splits that most decrease the entropy of our tree, which is defined as:

$$H(t_p) = -\sum_{i=1}^{K} p(i|t_p) \log_2(p(i|t_p)) \tag{3.8}$$

Similar to change in Gini impurity, information gain is simply the change in entropy between a parent and it's children:

$$IG(t_p, t_r, t_l) = H(t_p) - (P_r H(t_r) + P_l H(t_l)) \tag{3.9}$$

For both Information Gain and Gini impurity, the splitting process described above continues until each leaf node is perfectly pure or there is no node that can be split to improve the purity of our tree above a predefined threshold.

3.2.2 Random Forest: An Ensemble of Decision Trees. The random forests (RF) algorithm [59] is one of the most popular binary classification baselines for their simplicity and intuitive interpretability. A random forest is a collection of independent decision trees, each of which acts as a weak binary classifier for the dataset. Each tree is constructed using a bootstrap sample of training data and training features, meaning that only a random subset of data points and features are used for each tree. Classifications are made by allowing all trees in the sample to vote on the outcome for each datum, where votes are often weighted by each tree's confidence about the class of the datum in question. This bootstrapping and aggregation method defines an *ensemble classifier* and both increases accuracy and regularizes the classifier against overfitting.

3.3 XGBoost

3.3.1 Model Definition. *XGBoost: eXtreme Gradient Boosted Trees* is a smart method of building a forest of decision trees that incorporates regularization and a number of computational speed-ups to the method of gradient boosting proposed by Friedman in [60]. To begin, we define an objective function $obj(\Theta)$ as:

$$obj(\Theta) = L(\Theta) + \Omega(\Theta)$$

where $L(\Theta)$ is the *training loss function* of our model and $\Omega(\Theta)$ is the *regularization term*. Minimizing the training loss ensures that our model is predictive, while the regularization term penalizes our model for being too complex. This regularization term separates XGBoost from AdaBoost and other gradient boosting algorithms, which omit this term and are thus much more prone to overfitting. Combining these elements gives a model that is both simple and predictive.

3.3.2 Formulating the Objective Function. XGBoost classifies observations by taking the sum of scores generated by a forest of CARTs (Classification And Regression Trees) for each observation in a dataset. Let $f_k, k \in \{1, ..., K\}$ be a set of CARTs. Then the predicted score for an observation i is given by $\hat{y}_i = \sum_{k=1}^{K}$.

Training Loss. The loss function is defined as

$$L(\Theta) = \sum_{i=1}^{n} l(y_i, \hat{y}_i)$$

where $l(\cdot)$ could be square loss, cross entropy, logistic loss, gini impurity, etc.

Regularization. We define the complexity for a particular tree as

$$\Omega(f_k) = \gamma T + \frac{1}{2}\lambda \sum_{j=1}^{T} w_j^2$$

where T represents the number of leaf nodes of a given tree and w_j represents the score at leaf j. We can also replace $\lambda \sum_{j=1}^{T} w_j^2$ with $\alpha \sum_{j=1}^{T} |w_j|$ if we wish to perform L_1 regularization instead of L_2 regularization.

Objective Function. We define our objective function as follows:

$$obj^{(t)} = \sum_{i=1}^{n} l(y_i, \hat{y}_i^{(t-1)} + f_t(x_i)) + \sum_{k=1}^{t} \Omega(f_k)$$

When adding trees, we want to add each new tree f_t in a way to minimize our objective function. We use a second order approximation using the first two terms of the Taylor

expansion of our loss function to yield the following objective function at step t:

$$obj^{(t)} = \sum_{i=1}^{n}[l(y_i, \hat{y}_i^{(t-1)}) + g_i f_t(x_i) + \frac{1}{2}h_i f_t^2(x_i)] + \sum_{k=1}^{t}\Omega(f_k)$$

where $g_i = \partial_{\hat{y}_i^{(t-1)}} l(y_i, \hat{y}_i^{(t-1)})$ and $h_i = \partial^2_{\hat{y}_i^{(t-1)}} l(y_i, \hat{y}_i^{(t-1)})$. This simplifies to:

$$obj^{(t)} = \sum_{i=1}^{n}[g_i f_t(x_i) + \frac{1}{2}h_i f_t^2(x_i)] + \Omega(f_k) + constant$$

Plugging in for our regularization term, we obtain the following objective to optimize at each step:

$$obj^{(t)} = \sum_{i=1}^{n}[g_i f_t(x_i) + \frac{1}{2}h_i f_t^2(x_i)] + \gamma T + \frac{1}{2}\lambda\sum_{j=1}^{T} w_j^2 \tag{3.10}$$

$$= \sum_{j=1}^{T}[(\sum_{i\in I_j}(g_i)w_j + \frac{1}{2}(\sum_{i\in I_j}(h_i) + \lambda)w_j^2] + \gamma T \tag{3.11}$$

where $I_j = \{i : q(x_i) = j\}$ and $q : \mathbb{R}^d \to \{1, 2, \cdots, T\}$ is a function assigning each observation to a particular leaf node. If we let $G_j = \sum_{i\in I_j} g_i$ and $H_j = \sum_{i\in I_j} h_i$ then we obtain:

$$obj^{(t)} = \sum_{j=1}^{T}[G_i w_j + \frac{1}{2}(H_i + \lambda)w_j^2] + \gamma T$$

which yields optimal values:

$$w_j^* = -\frac{G_j}{H_j + \lambda}$$

$$obj^* = -\frac{1}{2}\sum_{j=1}^{T}(\frac{G_j^2}{H_j + \lambda}) + \gamma T$$

Based on this above strategy, when building our trees, we calculate the gain from a particular tree split as:

$$Gain = \frac{1}{2}\Big[\frac{G_L^2}{H_L + \lambda} + \frac{G_R^2}{H_R + \lambda} - \frac{(G_L + G_R)^2}{H_L + H_R + \lambda}\Big] - \gamma$$

The above equation thus mandates that splits can only occur if they improve the objective function more γ, which is exactly what pruning accomplishes in normal random forests. By incorporating this into the tree building process, XGBoost avoids the overhead of building and removing superfluous nodes.

3.4 Light Gradient Boosting Machine

The Light Gradient Boosting Machine (LGBM) is very similar to XGBoost, with three main modifications designed to improve speed and performance:

(i) **Leaf-wise splitting:** When constructing a tree in XGBoost, one selects a maximum depth d creates an ensemble of decision trees with 2^d nodes, so long as each parent met the appropriate splitting criteria. LGBM, on the other hand, iterates through leaves and selects the optimal leaf to split, regardless of its current level in the tree. This theoretically improves classification accuracy with a given number of splits.

(ii) **Gradient-based One Sided Sampling (GOSS):** GOSS is a means of sampling training instances to improve the calculation of information gain for each split. Noting that data instances with small gradients are generally already well trained, GOSS focuses on the data with the highest gradients to improve tree construction. Specifically, it takes the top $a \times 100\%$ of training instances with the largest gradients (call this set S_a) and then randomly samples gradients from $b \times 100\%$ of the remaining data (call this S_b). The information gain is then calculated as

$$IG = IG(S_a) + \frac{1-a}{b} IG(S_b)$$

Taking the instances with top gradients focuses our updates on under-trained instances while adding the reweighting term prevents the underlying distribution from changing significantly.

(iii) **Exclusive Feature Bundling:** Exclusive feature bundling is a means of speeding training by bundling sets of mutually exclusive sparse features (meaning features that are never concurrently nonzero) into single features. Though finding the optimal feature bundle is NP-hard, an approximate, greedy solution can be found in quadratic time by relaxing the mutual exclusivity constraint to allow the bundled features to have a fixed number of overlaps. By condensing numerous sparse features into a few dense features, fewer variables need to be evaluated to determine each split, thus leading to potentially large decreases in training time.

3.5 Recurrent Neural Networks

Recurrent Neural Networks (RNNs) are designed to extract meaningful relationships out of variable-length sequential data such as text or medical codes. Unlike traditional neural networks, which assume that all inputs are independent, the output of an RNN at each time step is dependent upon the computations in previous time steps.

One major problem with RNNs is the vanishing gradient problem [54, 55], which makes it difficult for vanilla RNNs to learn temporal relationships of more than a few time steps. This can be mitigated by using long short-term memory (LSTM) cells [61] or gated recurrent unit (GRU) cells [62]. These keep an internal state that is modified by sequential input, allowing cells to learn potentially arbitrarily long dependencies. Depictions of the LSTM and GRU cell architectures, borrowed from [3], are given in Figure 3.1. We discuss each in detail below.

3.5.1 Long Short Term Memory (LSTM) Cells.

LSTM cells encode long-term temporal relationships by keeping an internal state C_t that is passed between time steps. This state is updated with each new input x_t the network and is used to inform the output h_t at each time step.

Each LSTM cell takes intakes the inputs from the current time step x_t as well as the output of the previous step h_{t-1} and the hidden state from the previous time step C_{t-1}.

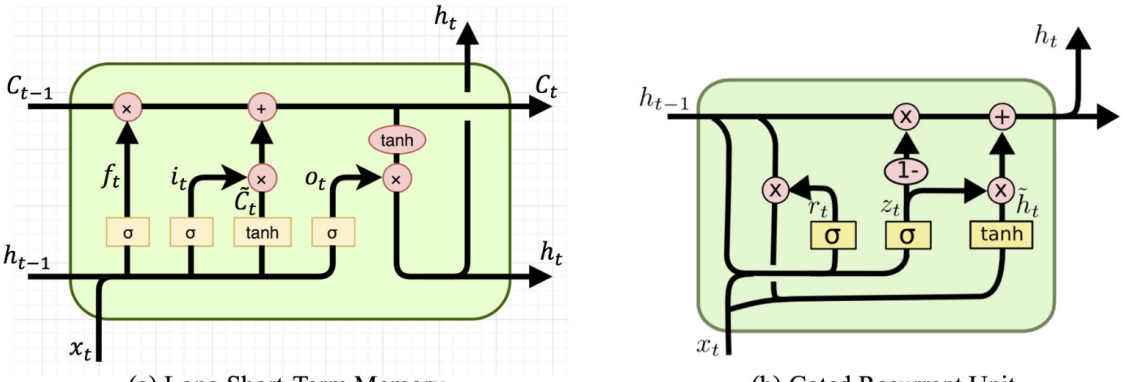


Figure 3.1: Depiction of LSTM and GRU architectures, borrowed from [3]. In the above, $\times$ denotes elementwise multiplication, $+$ denotes addition, and σ denotes the sigmoid function, which is taken after taking a learned affine transformation of the inputs.

The hidden state is updated and a new output h_t is calculated as follows: First, define the intermediate values:

$$f_t = \sigma(U^{(f)}h_{t-1} + W^{(f)}x_t + b^(f)) \tag{3.12}$$

$$i_t = \sigma(U^{(i)}h_{t-1} + W^{(i)}x_t + b^(i)) \tag{3.13}$$

$$o_t = \sigma(U^{(o)}h_{t-1} + W^{(o)}x_t + b^(o)) \tag{3.14}$$

$$\tilde{C}_t = \tanh(U^{(g)}h_{t-1} + W^{(g)}x_t + b^(g)) \tag{3.15}$$

Then, calculate the desired updates as follows:

$$C_t = \sigma(f_t * C_{t-1} + i_t * \tilde{C}_t) \tag{3.16}$$

$$h_t = \tanh(C_t) * o_t \tag{3.17}$$

In the calculation of C_t, $f_t * C_{t-1}$ is known as the "forget gate" because learns what part of the previous state to discard, whereas the piece $i_t * \tilde{C}_t$ could be considered the new knowledge learned by the internal state.

3.5.2 Gated Recurrent Unit (GRU) Cells. The GRU is similar to the LSTM, only that it dispenses with the internal hidden state and only uses the input for the current time step x_t and output of the previous time step h_{t-1}. It does this by calculating the intermediate values:

$$r_t = \sigma(U^{(r)}h_{t-1} + W^{(r)}x_t) \tag{3.18}$$

$$z_t = \sigma(U^{(z)}h_{t-1} + W^{(z)}x_t) \tag{3.19}$$

$$\tilde{h}_t = \tanh(U^{(h)}(r_t \odot h_{t-1}) + W^{(r)}x_t) \tag{3.20}$$

$$\tag{3.21}$$

Following which we return the output:

$$h_t = (1 - z_t) \odot h_{t-1} + (z_t \odot \tilde{h}_t) \tag{3.22}$$

3.6 Visualization Algorithms

3.6.1 t-SNE. In order to create meaningful visualizations of our clusters, we use a non-linear projection called "t-SNE," which stands for t-distributed Stochastic Neighbor Embedding [63]. The objective of t-SNE is to project data in a way that preserves local data structure (i.e. data that are close together in N-dimensional space are also are mapped together) while also keeping dissimilar data separate (i.e. the objective of principal component or linear discriminant projections). t-SNE does this by modeling similarity between points as conditional probabilities, where $P_{j|i}$ is defined to be the probability of choosing point j as a neighbor to point i if neighbors are picked proportional to the density of a normal distribution centered at point i. Mathematically, this is defined as:

$$p_{j|i} = \frac{exp(-||x_i - x_j||^2/2\sigma^2)}{\sum_{k \neq i} exp(-||x_i - x_k||^2/2\sigma^2)} \tag{3.23}$$

where x_i, x_j, and x_k are all points in our high-dimensional space and $||\cdot||$ is the Euclidian norm. We then define pseudo joint probabilities $p_{ij} = \frac{p_{i|j}+p_{j|i}}{2n}$, where n is the number of data points. Doing so gives us symmetric joint probabilities (i.e. $p_{ji} = p_{ij}$) and ensures that $\sum_j p_{ij} > \frac{1}{2n}$ for all i, is necessary to ensure that outlying points contribute meaningfully to the cost function that determines their low-dimensional representation.

We define the joint probability between our projected points y_i similarly using symmetric probabilities that follow a t-distribution with 1 degree of freedom as :

$$q_{ij} = \frac{(1 + ||y_i - y_j||^2)^{-1}}{\sum_{k \neq i}(1 + ||y_i - y_k||^2)^{-1}} \tag{3.24}$$

Student's t-distribution is chosen because its heavy tails will cause even moderately dissimilar points to be mapped far apart in projected space, which can be seen in the cost function (3.25) below. To create this function, we note that we want to find points y_i in our projected space that preserve our joint probabilities (i.e. $p_{ij} = q_{ij}$). We thus minimize the Kullback-Liebler divergence of the high-dimension distribution P and the low dimension distribution Q which is defined as:

$$C = KL(P||Q) = \sum_i \sum_j p_{ij} log(\frac{p_{ij}}{q_{ij}}) \tag{3.25}$$

Minimizing this cost function is equivalent to minimizing the cross entropy, since the two differ only by an additive constant. We use gradient descent (often with a momentum term to speed convergence) to minimize the above cost function, yielding a low-dimensional distribution that preserves joint probabilities between points. A more detailed exploration of the derivation of t-SNE can be found in [63].

4.1 Receiver Operating Characteristic (ROC)

4.1.1 ROC Curves. ROC curves are a visual means of evaluating how well a binary classifier rank-orders predictions for a binary prediction task (such as disease forecasting). We assume that our classifier gives each data point a score between 0 and 1 and then assigns predictions by choosing a threshold and labeling everything with a score above the threshold 1 and everything below 0. To create a ROC curve, one chooses N evenly spaced thresholds between 0 and 1 and calculates the true positive rate (TPR) and false positive rate (FPR) at each threshold. Let TP, FP, TN, and FN denote the number of true positives, false positives, true negatives, and false negatives predicted for a given threshold. Also let P and N denote the total numbers of positive and negative entries in our dataset. Then TPR and FPR are defined as follows:

$$TPR = \frac{TP}{P} \tag{4.1}$$

$$= \frac{TP}{TP + FN} \tag{4.2}$$

$$FPR = \frac{FP}{P} \tag{4.3}$$

$$= \frac{FP}{TP + FN} \tag{4.4}$$

Note that as the threshold value increases, the TPR and FPR both increase monotonically. An example of ROC curves for prediction of type II diabetes mellitus is given in figure 4.1

4.1.2 Area Under Receiver Operating Characteristic (AUROC) Curve.

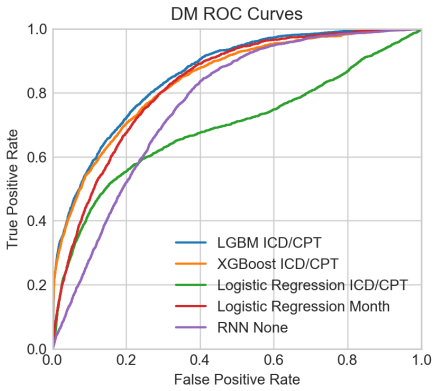


Figure 4.1: ROC curve for type II diabetes.

Definition and Probabilistic Interpretation. Once a ROC curve has been constructed, a common metric of evaluation is to look at area under the curve (AUC), which we will specifically denote as AUROC to differentiate it from the area under other curves used to evaluate an estimator. Intuitively, a higher AUC means that the classifier is doing a better job at differentiating between classes. In fact, the ROC AUC can be shown to be exactly the probability that a classifier assigns higher scores to positive examples than to negative ones. To do so, consider a binary classifier $\hat{p}(\cdot)$ that assigns $\mathbf{x}$ a score between 0 and 1, where the actual class of $\mathbf{x}$ is determined by choosing a threshold $\tau \in (0, 1)$ and assigning labels such that

$$\hat{y}(\mathbf{x}) = \begin{cases} 1, & \hat{p}(\mathbf{x}) > \tau \\ 0, & otherwise \end{cases}$$

Let $y(\mathbf{x})$ map $\mathbf{x}$ to its correct label and define two functions $T(\tau)$ and $F(\tau)$ as follows to respectively represent the TPR and FPR at a given threshold:

$$T(\tau) = P(\hat{p}(\mathbf{x}) > \tau | y(\mathbf{x}) = 1) \tag{4.5}$$

$$F(\tau) = P(\hat{p}(\mathbf{x}) > \tau | y(\mathbf{x}) = 0) \tag{4.6}$$

Writing the TPR as a function of the FPR allows us to define the AUROC as follows (as

in [64]):

$$AUROC = \int_0^1 T(F^{-1}(f_0))df_0 \tag{4.7}$$

$$= \int_0^1 P(\hat{p}(\mathbf{x}) > F^{-1}(f_0) \,|\, y(\mathbf{x}) = 1)df_0 \tag{4.8}$$

$$= \int_0^1 P(\hat{p}(\mathbf{x}) > F^{-1}(F(\tau)) \,|\, y(\mathbf{x}) = 1)\frac{\partial F(\tau)}{\partial \tau}d\tau \tag{4.9}$$

$$= \int_0^1 P(\hat{p}(\mathbf{x}) > \tau \,|\, y(\mathbf{x}) = 1) \cdot P(\hat{p}(\mathbf{z}) = \tau \,|\, y(\mathbf{z}) = 0)d\tau \tag{4.10}$$

$$= \int_0^1 P(\hat{p}(\mathbf{x}) > \hat{p}(\mathbf{z}) \,\&\, \hat{p}(\mathbf{z}) = \tau \,|\, y(\mathbf{x}) = 1 \,\&\, y(\mathbf{z}) = 0) \tag{4.11}$$

$$= P(\hat{p}(\mathbf{x}) > \hat{p}(\mathbf{z}) \,|\, y(\mathbf{x}) = 1 \,\&\, y(\mathbf{z}) = 0) \tag{4.12}$$

Problems with AUROC. One key disadvantage of the AUC score is that it can be easier to get a high ROC score on highly imbalanced datasets [65]. Moreover, when class imbalance is an issue, AUC doesn't give us much of an idea of the *precision* of a classifier (also known as the *positive predictive value*, which is the proportion of data points predicted to have positive labels that actually have positive labels. Using similar notation to above, precision is formally defined as:

$$precision = \frac{TP}{TP + FP} \tag{4.13}$$

This is bad because a classifier can have a superb AUC score but fail to deliver any actionable predictions on the data. Consider, for example, a case in which only 1% of our data has positive labels (this is very reasonable for disease prediction tasks, in which the prevalence of a disease in the training population can actually be much lower). Suppose that for any threshold for which we have false negatives every data point correctly predicted to be positive, there will exactly 9 false positives. We can therefore assume that the TPR increases linearly as a function of the FPR. This means that to correctly label every positive

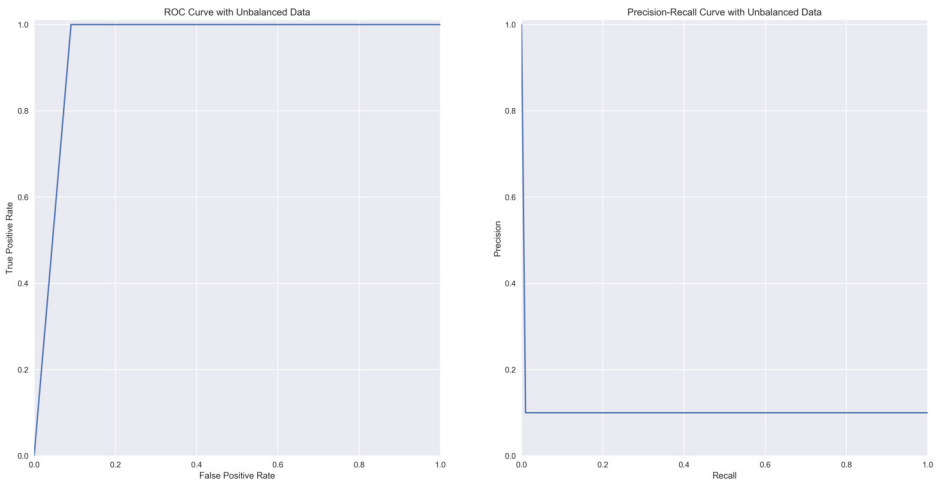


Figure 4.2: ROC and PR curves calculated from same classifier with unbalanced data. ROC curve looks very good for a classifier, but the precision and recall are very poor in practice. ROC curve has AUROC = **0.955** while PR curve has AP **0.1**. Note that the precision is trivially 1 when recall is 0 because there are no points labeled as positive.

data point, we must simply look at the 10% of the data receiving the highest score. At this cutoff, we will have a 100% true positive rate and a 9.1% false negative rate. An example ROC curve is shown in Figure 4.2. It is easy show analytically that this curve will have an AUC score of 0.955 (a very high score for disease prediction) despite never predicting more than 1 in 10 cases correctly. These predictions aren't particularly useful from the standpoint of a medical provider since 90% of early intervention efforts would end up being wasted on those who wouldn't be diagnosed anyway. Accordingly, we discuss means of alleviating these problems using *precision-recall*, as proposed in [65]

4.2 Precision-Recall (PR) Metrics

Precision, as defined in Equation 4.13, the proportion of entries predicted to be positive that are actually positive. *Recall* is simply true positive rate — the total proportion of the positive cases identified by a classifier. A precision-recall (PR) curve details the how accurately we can predict true positives as a function of the proportion of true positives we identify. This is

useful because it often better enables us to evaluate the practical usefulness of a classifier, for which it is often more important to have accurate predictions than to predict every positive case (i.e. in the case of allocating limited financial or medical resources)

Though the difference between comparing precision and recall instead of FPR and recall (TPR) is subtle, the two can have very different results when evaluating a classifier, as noted in [66]. Figure 4.2 shows ROC and PR curves for the same in classifier, used to classify data with heavy class imbalance as described in Section 4.1.2. Some have proposed that area under the precision-recall curve (AUPRC) is a better metric than ROC because it is not as easily confounded by class imbalance [65]. In some cases, linear interpolation between points on the PR curve can cause AUPRC estimates to be overly optimistic. Accordingly, we calculate the closely related average precision (AP) of each of our classifiers instead of the AUPRC. AP is defined as:

$$AP = P_n(R_n - R_{n-1}) \tag{4.14}$$

Where P_i and R_i are the precision and recall for the i^{th} threshold, respectively. We will report both AP and AUROC for our classifiers for robustness and comparability with other papers.

Chapter 5. Experiments

We present a synopsis of an insurance claims dataset used for constructing embeddings and for predicting future disease onset for five high-impact chronic diseases. We discuss general strategies for data preparation that allow us to intelligently preprocess our data so that we can test the usefulness of our embeddings in our disease prediction experiments without "cheating" by accidentally encoding the desired predictions in the embeddings themselves.

5.1 Data

We compare the previously described embedding and disease prediction strategies using a private insurance claims dataset consisting of 472,120 individuals. All individuals lived in a large metropolitan area and were monitored between 2002 and 2010. All diagnoses are labeled with ICD-9 diagnosis codes, with procedures labeled using CPT procedure codes. Insurance claims data are structured such that each claim lists either a single filled prescription for medication, or a single medical procedure performed and up to four diagnosis codes justifying that procedure. Of the four diagnosis codes given, the first (primary) diagnosis is given as a full code while the second through fourth (secondary) diagnoses are truncated at 4 digits. This prevents us from identifying the exact ICD-9 code to which each corresponds, though we can still get a good idea about the disease category each represents just by looking at these digits.

Many of the individuals in our dataset received drug coverage through a third-party provider, so we do not include drug codes in our analysis for uniformity across patients.

In addition to individual medical histories, our data also contains age and sex of each patient as demographic info. We will incorporate this information to improve our modeling performance. Basic summary statistics on our data are given in Table 5.1.

In our experiments, we will try to predict five diseases: Heart Failure (HF), Diabetes Melitus type II (DM), Chronic Kidney Disease (CKD), Acute Myocardial Infarction (AMI),

and Chronic Obstructive Pulmonary Disease (COPD). Each of these diseases was selected for being both life-altering and preventable. We hope that being able to forecast the onset of these diseases will allow medical practitioners, insurance providers, and individuals to take measures to avoid the onset of these conditions.

In addition to medical history of each individual, we include age and sex as demographic information to better improve our predictions.

Category	Number Patients
All Patients	472,120
HF	2,336
DM	9,916
CKD	2,589
AMI	1,574
COPD	3,953
Mean Age	33

Table 5.1: Basic statistical and demographic characteristics of dataset

An individual was identified as *positive* for one of the above diseases if we observed two or more codes corresponding to that disease in his or her medical record, as done in [2, 24]. Codes used to identify each disease are given in Table 5.2. We require at least two codes to avoid false positives caused by medical coding errors. Individuals with exactly one code are labeled as ambiguous and are dropped.

Disease	ICD-9 Codes	CPT Codes
HF	398.91, 402.*1, 404.*1, 404.*3, 428.*	None
AMI	410.*	None
DM	250.*0, 250.*2	None
CKD	403.*, 404.*, 582.*, 583.*, 585.*, 586.*, 588.0, V56	90918, 90919, 9092*, 9093*, 9094*, 9095*, 9096*, 9097*, 9098*, 9099*
COPD	491.*, 492.*, 494.*, 496	None

Table 5.2: Diagnosis and procedure codes used to identify each disease. * indicates that codes with any digit(s) in this place correspond to the disease in question. For CKD, all CPT codes correspond to dialysis-related procedures.

Due to the competitive nature of the insurance market, many individuals switch insurance providers frequently as their employers seek to find ways to cut costs. As such, most of the individuals in our dataset were not enrolled continuously during the nine period covered by the data, but were enrolled for a subset of the years, sometimes with multi-year gaps in data coverage.

These coverage gaps present a number of problems for our experiments. First, embedding algorithms that expect continuous coverage could learn spurious patterns when trying to predict a next visit across a large coverage gap. However, the ratio of coverage gaps to total visits is small, thus we do not expect this to substantially bias our embedding results. Second, one must adjust the normal retrospective cohort study methodology, which uses fixed observation and prediction periods, to account for asynchronous enrollment. Failure to do so would drastically limit the already small number of individuals in the sample with each disease, increasing chances of overfitting and thereby limiting the generalizability of the results.

To address this second problem, we align our patients by time of failure, as done in [67]. To do so, we sequentially parse through each patient's history. Once we have observed two codes corresponding to one of our diseases, we mark that individual as "positive" for that disease truncate that patient's history after a random date between 30 and 365 days prior to the onset of the given disease. While this requires us to create a separate dataset for each disease, it ensures that we have sufficiently large populations in each to find interesting and meaningful results.

For both sets of experiments, we use the same train-test-validation split of 60/20/20 to prevent the embeddings from explicitly learning to encode sequential relationships in the test data. We use the validation data for hyperparameter tuning and early stopping. When we perform our disease prediction experiments, we will additionally remove all patients who did not have at least one year of medical history in the observation period. Numbers of total patients included in each prediction experiment are summarized in Table 5.1.

5.2 Constructing and Evaluating Embeddings

5.2.1 Embedding Construction. In this section, we construct and qualitatively compare three sets of embeddings for diagnosis and procedure codes. First, we naively construct embeddings using only the sequence of codes (rather than temporal relationships) to define co-occurrence. We refer to these as our `naive` embeddings. Second, we construct embeddings using short-term temporal dependencies between codes. In this case, we define two codes as co-occurring if they occur in the same visit (i.e. in the same day). This is reminiscent of the means used in [2, 53, 32] to pre-train embeddings for neural networks. We refer to these as `short-term` embeddings. Finally, we test whether accounting for long term temporal relationships improves embedding quality, as noted in [46]. We refer to these as `long-term` embeddings. A summary of the different co-occurrence definitions used to construct our embeddings is given in Table 5.3.

Embedding	Co-occurrence Definition
`short-term`	Codes occur in same visit (i.e. on same day)
`long-term`	Codes occur within a month of each other

Table 5.3: Summary of embedding types, as well as a metric of their consistency with known medical ontologies.

In each case, we use GloVe [50] to train our embedding vectors, since it can be easily adapted to use a temporal instead of sequential definition of co-occurrence and continues to show near state-of-the-art performance on NLP tasks. As done in the GloVe paper, we drop all codes that do not occur at least 10 times in our training dataset, since these can add unnecessary noise to our model.

We construct all of our embeddings using only the 60% of our data designated for training. We do this to ensure that when we later compare the effectiveness of the learned embeddings in disease prediction, we do not inadvertently "cheat" by encoding data from the test or validation sets in our embeddings.

5.2.2 Embedding Evaluation.

We evaluate our embeddings in three ways:

(i) We perform a qualitative evaluation by looking at the nearest neighbors of key diagnosis codes and determining whether they align with published medical knowledge. Based on the distributional hypothesis [45], codes that occur together frequently ought to correspond to linked conditions. Specifically, we cluster our embeddings using variations of k-means clustering and then examine the contents of a few of these clusters to get an idea of whether or not this assumption holds.

(ii) We test the embeddings in a variety of machine learning algorithms to see which can best be aggregated together to predict future disease onset. This is discussed further in Section 5.3.

Clustering and Qualitative Evaluation. Once we have obtained embeddings for our data, we cluster our data using the K-Means algorithm. We find these clusters using the following two methods:

(i) Let initial cluster centroids be the mean of the embedded vector representations for each of the 278 CCS groups. This presents an intuitive choice for clusters, but also fixes the number of clusters at 279, where the 279th cluster corresponds to oddball procedures not related to a single diagnosis group, such as x-rays.

(ii) Choose initial centroids according to the `k-means++` procedure described in [68] as follows:

   (a) Randomly choose the first centroid to be a point from the dataset to be clustered.

   (b) For each successive centroid, pick x_j to be the next centroid with probability $p_j = \frac{D(x_j)}{\sum_{j=1}^{n} D(x_j)}$, where $D(x_j)$ is the distance from x_j to the nearest centroid. This step is repeated until k centroids have been chosen.

Clustering Results. Here, we qualitatively evaluate our clusters based on how well each captures known disease comorbidities. In the absence of exhaustive data on known

comorbidities, we heuristically assess both cluster validity and the presence of comorbidities by inspecting a few key clusters element-by-element to determine whether each data point is actually related to the main theme of the cluster. We plot a few 2-dimensional projections of our clusters in Figure 5.1

To illustrate how this procedure works, consider the cluster containing the diagnosis code for advanced chronic kidney disease, known as end-stage renal disease (ESRD). Once an individual enters end-stage kidney disease, his or her kidneys have lost so much function that dialysis is required multiple times a week to properly filter toxins from the blood. Worse, chronic kidney disease is irreversible, so individuals in end-stage must either receive a kidney transplant or receive dialysis for the rest of their lives. Thus, we would expect codes in our ESRD cluster to be related to advanced kidney damage, dialysis, kidney transplants, and associated conditions.

Table 5.4 contains a reasonably representative subset of the codes contained in our ESRD cluster. It is readily apparent that entries 1, 3, 5, 7, 8, 9, and 10 correspond directly to renal problems, as expected. Entry 2, malnutrition, is also commonly associated with ESRD because dialysis and associated chronic inflammation can cause patients to suffer from protein-energy malnutrition, which occurs when patients either have insufficient calories or protein to meet their metabolic needs [69]. Similarly, osteomalacia (entry 6) is softening of the bones caused by lack of vitamin D, the metabolism of which is impeded by kidney disease [70]. Moreover, chromosome anomalies (entry 4) have been identified as risk factors for chronic kidney disease [71] and intracranial hemorrhage has been associated with polycystic kidney disease [72]. Finally, cytotoxic panel-reactive antibody screenings (entry 12) are performed on individuals awaiting organ (i.e. kidney) donation to assess the compatibility of an organ transplant from a prospective donor. Thus we see that each of the codes we investigated from our ESRD cluster is closely linked to ESRD, which lends strength to the hypothesis that clusterings could be useful in identifying disease comorbidities. We further see that our combining clustering with our embeddings is able to capture relationships

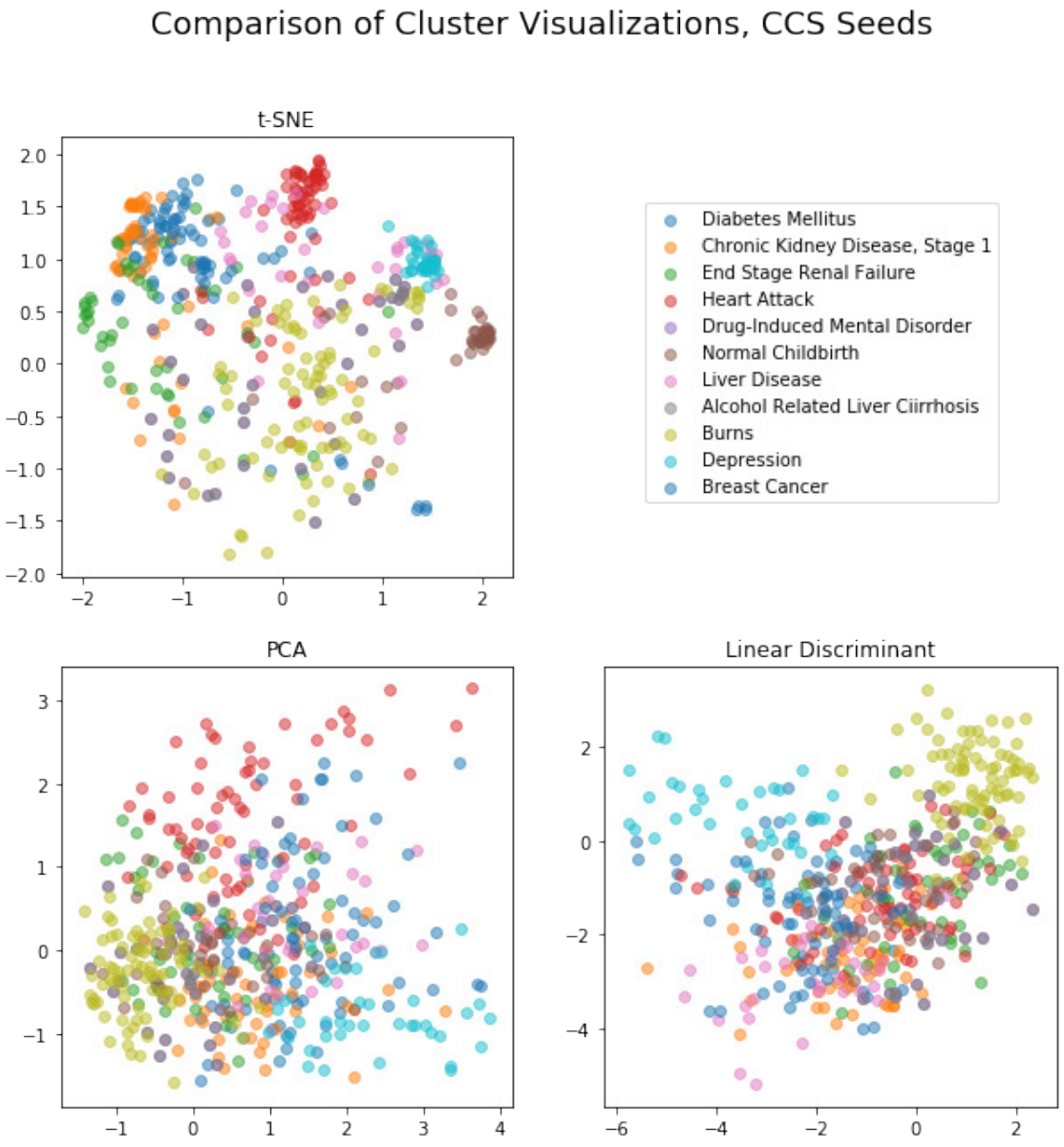


Figure 5.1: Projections of our clusters using t-SNE, linear discriminant analysis, and principal components.

	End-Stage Renal Disease Related Code	Code Type	CCS Group
1	Hemodialysis procedure requiring repeated evaluation(s) with or without substantial revision of dialysis prescription	Procedure	58
2	Malnutrition of moderate degree	Diagnosis	52
3	renal dialysis status	Diagnosis	158
4	Conditions due to anomaly of unspecified chromosome	Diagnosis	217
5	End-stage renal disease (ESRD) related services per full month; for patients twenty years of age and older	Procedure	58
6	Osteomalacia, unspecified	Diagnosis	52
7	Renal osteodystrophy	Diagnosis	161
8	End-stage renal disease (ESRD) related services monthly, for patients 20 years of age and older; with 4 or more face-to-face physician visits per month	Procedure	227
9	Secondary hyperparathyroidism (of renal origin)	Diagnosis	161
10	Encounter for extracorporeal dialysis	Diagnosis	158
11	Other and unspecified intracranial hemorrhage following injury without mention of open intracranial wound, with no loss of consciousness	Diagnosis	233
12	Serum screening for cytotoxic percent reactive antibody (PRA); standard method	Procedure	235

Table 5.4: Subset diagnoses and procedures contained in cluster corresponding to end-stage renal disease. Note that more than half of these conditions explicitly relate to renal problems or dialysis.

between diagnoses and procedures not captured in CCS groupings alone, as can be seen by the diversity of CCS groupings in table 5.4.

We note that while the results of clustering around ESRD are promising, other diagnoses exhibit worse results. Burns, for example, are clustered together with insect bites, presumably because both deal with skin irritation and blistering. In spite of this, we believe that our clusters could be a useful tool because we would expect chronic diseases (e.g. kidney disease) to show more consistent, long-term patterns of comorbidities than incidental injuries (e.g. burns). A broader, more systematic exploration of these patterns is a potential area of future research.

Nearest Neighbors. In addition to looking at clusters, we check that the nearest neighbors of codes align with medical knowledge. The nearest neighbors to a ESRD and social

phobia are shown in few common diseases shown in Tables 5.5 and 5.6.

	End Stage Renal Disease Nearest Neighbors	**Code Type**
1	Hemodialysis procedure with single physician evaluation	Procedure
2	chronic renal failure	Diagnosis
3	End-stage renal disease (ESRD) related services monthly, for patients 20 years of age and older; with 4 or more face-to-face physician visits per month	Procedure
4	Impaired renal function with necrosis	Diagnosis
5	End-stage renal disease (ESRD) related services monthly, for patients 20 years of age and older; with 2-3 face-to-face physician visits per month	Procedure
6	Renal failure, unspecified	Diagnosis
7	Hypertensive chronic kidney disease - Unspecified	Diagnosis
8	Chronic kidney disease, unspecified	Diagnosis
9	Hypertensive chronic kidney disease, unspecified, with chronic kidney disease stage V or end stage renal disease	Diagnosis
10	End-stage renal disease (ESRD) related services per full month; for patients twenty years of age and older	Procedure

Table 5.5: Nearest neighbors of End Stage Renal Disease given by embedded points. Note that all related conditions correspond to renal (kidney) failure or associated treatments

	Social Phobia Nearest Neighbors	**Code Type**
1	Posttraumatic stress disorder	Diagnosis
2	Major depressive affective disorder, recurrent episode, mild	Diagnosis
3	Unspecified personality disorder	Diagnosis
4	Adjustment disorder with anxiety	Diagnosis
5	Individual psychotherapy, insight oriented, behavior modifying and/or supportive, in an office or outpatient facility, approximately 20 to 30 minutes face-to-face with the patient	Procedure
6	Other occupational circumstances or maladjustment	Diagnosis
7	Other bipolar disorders	Diagnosis
8	Bipolar I disorder, most recent episode (or current) depressed, moderate	Diagnosis
9	Gambling and betting	Diagnosis
10	Psychic factors associated with diseases classified elsewhere	Diagnosis

Table 5.6: Nearest neighbors of Social Phobia in embedded data. Note that all 10 neighbors are related to psychological conditions

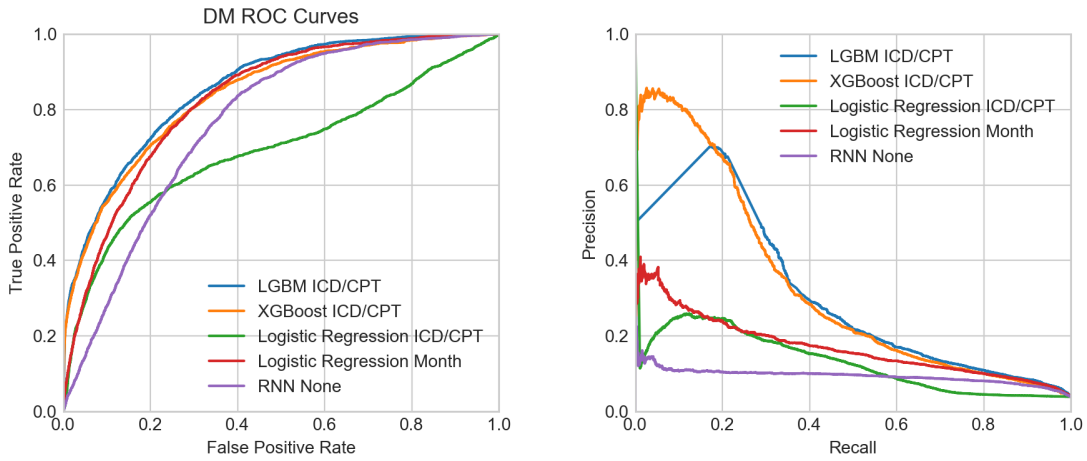
5.3 Individual Disease Prediction

Once we have split, cleaned, labeled, and embedded our data, we are ready to compare a variety of disease prediction models to determine both which models can best capture patterns of disease progression and which embeddings can best encode disease relationships related to a patient's medical future. We align our patients by time of disease onset and restrict each patient to have exactly one year of data in the observation period as discussed in Section 5.1. We select parameters for each of our models based on which performs best on a grid parameter grid search evaluated on holdout validation data. We search over the following parameters for each model:

- **Logistic Regression:**
  - Regularization: $\{L_1, L_2\}$
  - C (Regularization Weight): $\{0.01, 0.1, 1, 10, 100\}$
- **Random Forest:**
  - Number of Trees: $\{100, 200, 400\}$
  - Max Depth: $\{3, 6, 9\}$
  - Min Samples per Leaf: $\{2, 8, 32\}$
  - Decision Rule: $\{Gini, CrossEntropy\}$
- **XGBoost:**
  - Number of Trees: $\{100, 200, 400\}$
  - Learning Rate: $\{0.01, 0.1, 1\}$
  - Max Depth: $\{3, 6, 9\}$
  - Proportion of Columns per Tree: $\{.1, .2, .4\}$
- **LightGBM:**

- Number of Trees: $\{100, 200, 400\}$
- Learning Rate: $\{0.01, 0.1, 1\}$
- Max Depth: $\{3, 6, 9\}$
- Proportion of Columns per Tree: $\{.1, .2, .4\}$
- Boosting Algorithm: $\{GBM, DART\}$

- **RNN:**
  - RNN State Regularization Weight: $\{.001, 0.01, .01\}$
  - RNN State Size: $\{128, 256\}$
  - Fully-Connected Layer Size: $\{256, 512\}$



(a) ROC curve for type II diabetes.

(b) PR curve for type II diabetes.

Figure 5.2

In addition to the parameter grid search, we compensate for our class imbalance by upweighting the disease class to have the same total weight as the healthy class for LR, RF, XGB, and LGBM. We address class imbalance for our RNN by sampling training batches to have a 1:1 healthy:sick ratio for training.

In order to get metrics on how well non-RNN algorithms can learn on embedded claims sequences, we aggregate each by summing together the embeddings of claims occurring in the

observation period, weighting each by the frequency that its code appeared in the individual's claims during that time. RNNs are fed sequential visits, where each visit is represented by a multi-hot vector of the codes it contained.

Once we have selected our best models using validation data, we evaluate each on our test dataset and obtain ROC and PR curves for each. We also calculate AUROC and AUPRC scores for each, recorded in Tables 5.7 and 5.8, respectively. We plot ROC and PR curves for DM in Figure 5.2 and leave the rest to the appendix.

Model	**HF**	**DM**	**CKD**	**AMI**	**COPD**
LR+Raw	.5500	.6925	.5561	.5908	.5670
LR+Short-Term	.8794	.8195	.8424	**.8676**	.7989
LR+Long-Term	.8818	.8280	.8442	.8626	.8014
RF+Raw	.8594	.8224	.8321	.8281	.7836
RF+Short-Term	.8631	.8097	.8280	.8474	.7765
RF+Long-Term	.8762	.8127	.8359	.8498	.7797
XGB+Raw	.8056	.8408	.7769	.7982	.7631
XGB+Short-Term	.8614	.8030	.8105	.8425	.7645
XGB+Long-Term	.8714	.8083	.8294	.8326	.7576
LGBM+Raw	.8778	**.8568**	.8416	.8483	**.8093**
LGBM+Short-Term	.8773	.8221	.8408	.8632	.7998
LGBM+Long-Term	.8863	.8261	.8368	.8607	.8010
LSTM+Raw	.8309	.7659	.7916	.8437	.7643
LSTM+Short-Term	**.8899**	.7681	.8469	.8456	.7609
LSTM+Long-Term	.8894	.7743	**.8518**	.8486	.7606

Table 5.7: AUROC scores from disease prediction experiments.

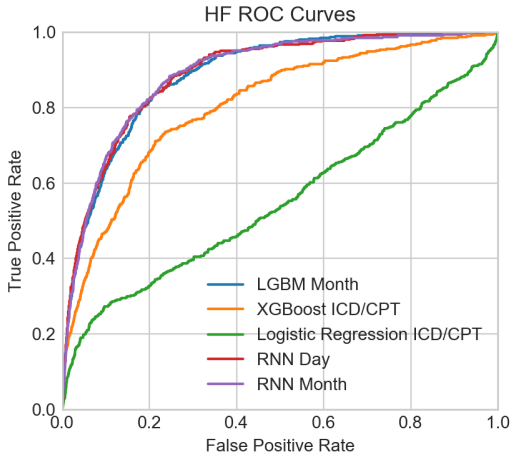
Despite the emphasis on RNNs in recent disease prediction literature, LSTMs did not consistently outperform simpler baseline models like LGBM or even logistic regression. LSTMs never have the highest AP for any disease and only win on 2 of 5 diseases with AUROC. These simpler models are both faster and more interpretable than LSTMs, and should therefore not be discarded in disease prediction tasks. We do note that learned embeddings substan-

Model	HF	DM	CKD	AMI	COPD
LR+Raw	.0288	.1330	.0237	.0156	.0377
LR+Short-Term	**.1052**	.1552	**.0705**	.0338	.0649
LR+Long-Term	.1006	.1692	.0662	.0343	.0636
RF+Raw	.0559	.2392	.0526	.0236	.0560
RF+Short-Term	.0588	.1795	.0548	.0279	.0516
RF+Long-Term	.0687	.1893	.0551	.0296	.0543
XGB+Raw	.0704	**.3297**	.0648	.0233	.0647
XGB+Short-Term	.0664	.1898	.0639	.0296	.0429
XGB+Long-Term	.0766	.2101	.0669	.0291	.0454
LGBM+Raw	.0780	.3080	.0625	.0336	**.0733**
LGBM+Short-Term	.0769	.1801	.0562	**.0388**	.0641
LGBM+Long-Term	.0884	.2045	.0612	.0331	.0635
LSTM+Raw	.0452	.0940	.0285	.0228	.0432
LSTM+Short-Term	.0823	.0943	.0544	.0219	.0393
LSTM+Long-Term	.0779	.0977	.0639	.0236	.0392

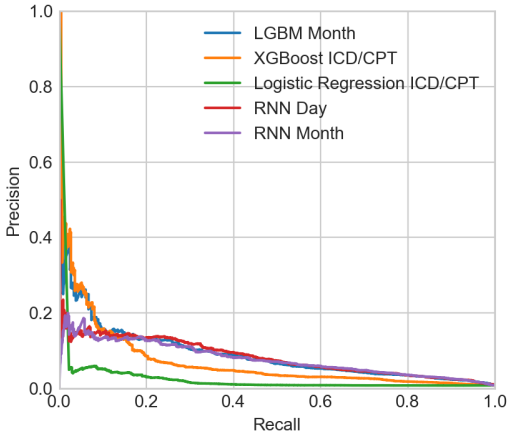
Table 5.8: AUPRC scores from disease prediction experiments.

tially improved the performance of logistic regression models, indicating that embeddings do learn meaningful disease features that can be captured by linear models without significant post-processing.

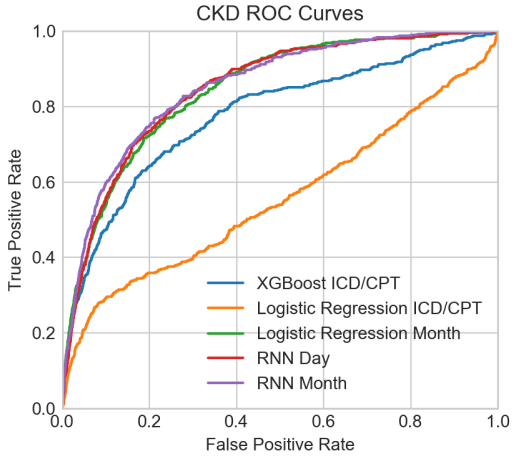
Appendix A. ROC and PR Curves for other diseases



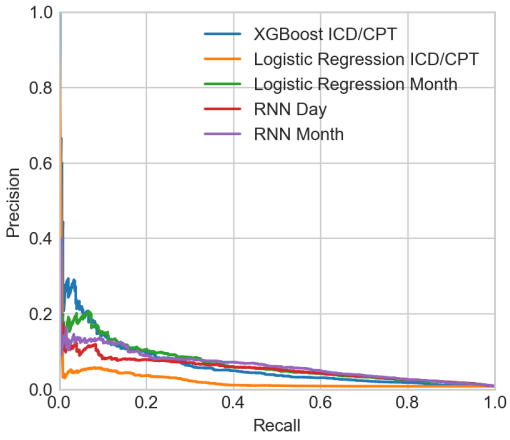
(a) ROC curve for heart failure.



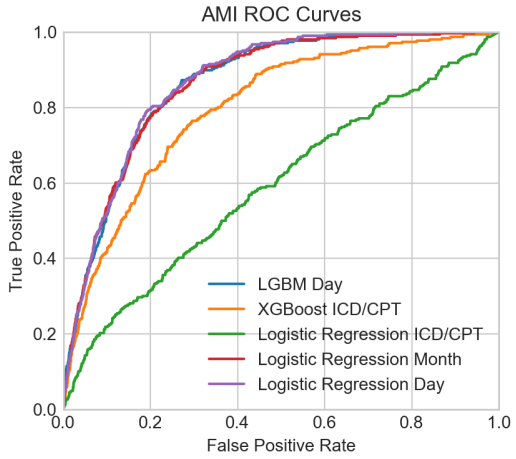
(b) PR curve for heart failure.



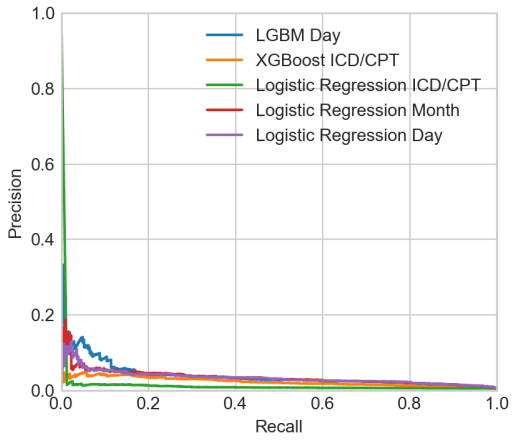
(a) ROC curve for chronic kidney disease.



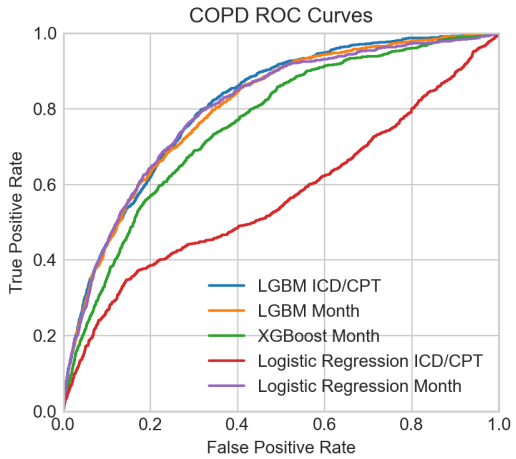
(b) PR curve for chronic kidney disease.



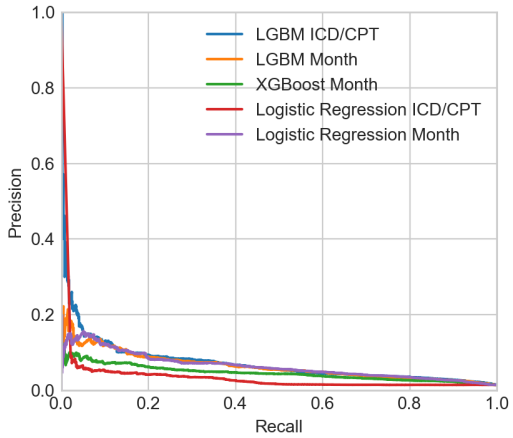
(a) ROC curve for acute myocardial infarction (heart attack).



(b) PR curve for acute myocardial infarction (heart attack).



(a) ROC curve for chronic obstructive pulmonary disease.



(b) PR curve for chronic obstructive pulmonary disease.

Bibliography

[1] 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*, KDD '16, pages 1495–1504, New York, NY, USA, 2016. ACM.

[2] Edward Choi, Mohammad Taha Bahadori, Le Song, Walter F. Stewart, and Jimeng Sun. Gram: Graph-based attention model for healthcare representation learning. In *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '17, pages 787–795, New York, NY, USA, 2017. ACM.

[3] Zhang Hao. Lstm and gru – formula summary, 2017.

[4] Aaron C Catlin and Cathy A Cowan. History of health spending in the united states, 1960-2013. *Baltimore, MD: Centers for Medicare and Medicaid Services*, 2015.

[5] Medicaid: A small share of enrollees consistently accounted for a large share of expenditures. Number 15-460. Government Accountability Office, May 2015.

[6] M A Winkleby, D E Jatulis, E Frank, and S P Fortmann. Socioeconomic status and health: how education, income, and occupation contribute to risk factors for cardiovascular disease. *American Journal of Public Health*, 82(6):816–820, 1992. PMID: 1585961.

[7] Michael Marmot and Jessica J Allen. Social determinants of health equity, 2014.

[8] Centers for Disease Control and Prevention. Leading causes of death and numbers of deaths, by sex, race, and hispanic origin: United states, 1980 and 2014. *Health*, 2015.

[9] Brian W Ward, Jeannine S Schiller, and Richard A Goodman. Peer reviewed: Multiple chronic conditions among us adults: A 2012 update. *Preventing chronic disease*, 11, 2014.

[10] Ralph Snyderman. Personalized health care: from theory to practice. *Biotechnology Journal*, 7(8):973–979, August 2012.

[11] Ralph Snyderman and R Sanders Williams. Prospective medicine: the next health care transformation. *Academic Medicine*, 78(11):1079–1084, 2003.

[12] Andre Esteva, Brett Kuprel, Roberto A Novoa, Justin Ko, Susan M Swetter, Helen M Blau, and Sebastian Thrun. Dermatologist-level classification of skin cancer with deep neural networks. *Nature*, 542(7639):115, 2017.

[13] QingZeng Song, Lei Zhao, XingKe Luo, and XueChen Dou. Using deep learning for classification of lung nodules on computed tomography images. *Journal of healthcare engineering*, 2017, 2017.

[14] L. Khedher, J. Ramrez, J.M. Grriz, A. Brahim, and F. Segovia. Early diagnosis of alzheimers disease based on partial least squares, principal component analysis and support vector machine using segmented mri images. *Neurocomputing*, 151:139 – 150, 2015.

[15] C. B. Delahunt, C. Mehanian, L. Hu, S. K. McGuire, C. R. Champlin, M. P. Horning, B. K. Wilson, and C. M. Thompon. Automated microscopy and machine learning for expert-level malaria field diagnosis. In *2015 IEEE Global Humanitarian Technology Conference (GHTC)*, pages 393–399, Oct 2015.

[16] C. Salvatore, A. Cerasa, I. Castiglioni, F. Gallivanone, A. Augimeri, M. Lopez, G. Arabia, M. Morelli, M.C. Gilardi, and A. Quattrone. Machine learning on brain mri data for differential diagnosis of parkinson's disease and progressive supranuclear palsy. *Journal of Neuroscience Methods*, 222:230 – 237, 2014.

[17] Jessica Plati, Octavian Bucur, and Roya Khosravi-Far. Apoptotic cell signaling in cancer progression and therapy. *Integrative biology*, 3(4):279–296, 2011.

[18] Xiaowei Chen, Xiaobo Zhou, and Stephen TC Wong. Automated segmentation, classification, and tracking of cancer cell nuclei in time-lapse microscopy. *IEEE Transactions on Biomedical Engineering*, 53(4):762–766, 2006.

[19] Aurélien Lucchi, Kevin Smith, Radhakrishna Achanta, Graham Knott, and Pascal Fua. Supervoxel-based segmentation of mitochondria in em image stacks with learned shape features. *IEEE transactions on medical imaging*, 31(2):474–486, 2012.

[20] Rachel Hodos, Ping Zhang, Hao Chih Lee, Qiaonan Duan, Zichen Wang, Neil R. Clark, Avi Ma'ayan, Fei Wang, Brian Kidd, Jianying Hu, David Sontag, and Joel Dudley. Cell-specific prediction and application of drug-induced gene expression profiles. In *Proceedings of the Pacific Symposium on Biocomputing (PSB)*, 2018.

[21] Mark-Anthony Bray, Shantanu Singh, Han Han, Chadwick T Davis, Blake Borgeson, Cathy Hartland, Maria Kost-Alimova, Sigrun M Gustafsdottir, Christopher C Gibson, and Anne E Carpenter. Cell painting, a high-content image-based assay for morphological profiling using multiplexed fluorescent dyes. *Nature protocols*, 11(9):1757, 2016.

[22] World Health Organization et al. International classification of diseases:[9th] ninth revision, basic tabulation list with alphabetic index. 1978.

[23] World Health Organization. *International statistical classification of diseases and related health problems*, volume 1. World Health Organization, 2004.

[24] Narges Razavian and David Sontag. Temporal convolutional neural networks for diagnosis from lab tests. *CoRR*, abs/1511.07938, 2015.

[25] Una Kyriacos, J Jelsma, and S Jordan. Monitoring vital signs using early warning scoring systems: a review of the literature. *Journal of nursing management*, 19(3):311–330, 2011.

[26] L Tarassenko, A Hann, and D Young. Integrated monitoring and analysis for early warning of patient deterioration. *British journal of anaesthesia*, 97(1):64–68, 2006.

[27] Brynne A Sullivan and Karen D Fairchild. Predictive monitoring for sepsis and necrotizing enterocolitis to prevent shock. In *Seminars in Fetal and Neonatal Medicine*, volume 20, pages 255–261. Elsevier, 2015.

[28] Maximilian Ilse, Jakub M. Tomczak, and Max Welling. Attention-based deep multiple instance learning. *CoRR*, abs/1802.04712, 2018.

[29] Oren Z Kraus, Jimmy Lei Ba, and Brendan J Frey. Classifying and segmenting microscopy images with deep multiple instance learning. *Bioinformatics*, 32(12):i52–i59, 2016.

[30] Melih Kandemir, Chong Zhang, and Fred A Hamprecht. Empowering multiple instance histopathology cancer diagnosis by cell graphs. In *International Conference on Medical Image Computing and Computer-Assisted Intervention*, pages 228–235. Springer, 2014.

[31] Subrajeet Mohapatra, Dipti Patra, and Sanghamitra Satpathy. An ensemble classifier system for early diagnosis of acute lymphoblastic leukemia in blood microscopic images. *Neural Computing and Applications*, 24(7):1887–1904, Jun 2014.

[32] Edward Choi, Mohammad Taha Bahadori, Andy Schuetz, Walter F. Stewart, and Jimeng Sun. Doctor ai: Predicting clinical events via recurrent neural networks. In Finale Doshi-Velez, Jim Fackler, David Kale, Byron Wallace, and Jenna Wiens, editors, *Proceedings of the 1st Machine Learning for Healthcare Conference*, volume 56 of *Proceedings of Machine Learning Research*, pages 301–318, Children's Hospital LA, Los Angeles, CA, USA, 18–19 Aug 2016. PMLR.

[33] Adler Perotte, Rajesh Ranganath, Jamie S. Hirsch, David Blei, and Nomie Elhadad. Risk prediction for chronic kidney disease progression using heterogeneous electronic health record data and time series analysis. *Journal of the American Medical Informatics Association: JAMIA*, 22(4):872–880, July 2015.

[34] Navdeep Tangri, Lesley A Stevens, John Griffith, Hocine Tighiouart, Ognjenka Djurdjev, David Naimark, Adeera Levin, and Andrew S Levey. A predictive model for progression of chronic kidney disease to kidney failure. *Jama*, 305(15):1553–1559, 2011.

[35] Mohammed Khalilia, Sounak Chakraborty, and Mihail Popescu. Predicting disease risks from highly imbalanced data using random forest. *BMC medical informatics and decision making*, 11(1):1, 2011.

[36] Xiang Wang, David Sontag, and Fei Wang. Unsupervised learning of disease progression models. In *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 85–94. ACM, 2014.

[37] Zachary C Lipton, David C Kale, Charles Elkan, and Randall Wetzell. Learning to diagnose with lstm recurrent neural networks. *arXiv preprint arXiv:1511.03677*, 2015.

[38] Narges Razavian, Saul Blecker, Ann Marie Schmidt, Aaron Smith-McLallen, Somesh Nigam, and David Sontag. Population-Level Prediction of Type 2 Diabetes From Claims Data and Analysis of Risk Factors. *Big Data*, 3(4):277–287, December 2015.

[39] Michael A. Cucciare and William O'Donohue. Predicting future healthcare costs: how well does riskadjustment work? *Journal of Health Organization and Management*, 20(2):150–162, 2006. PMID: 16869351.

[40] David W. Frost, Shankar Vembu, Jiayi Wang, Karen Tu, Quaid Morris, and Howard B. Abrams. Using the electronic medical record to identify patients at high risk for frequent emergency department visits and high system costs. *The American Journal of Medicine*, 130(5):601.e17 – 601.e22, 2017.

[41] Mary E. Charlson, Robert E. Charlson, Janey C. Peterson, Spyridon S. Marinopoulos, William M. Briggs, and James P. Hollenberg. The charlson comorbidity index is adapted to predict costs of chronic disease in primary care patients. *Journal of Clinical Epidemiology*, 61(12):1234 – 1240, 2008.

[42] John A. Fleishman and Joel W. Cohen. Using information on clinical conditions to predict high-cost patients. *Health Services Research*, 45(2):532–552, 2010.

[43] David Kartchner, Andy Merrill, and Jonathan Wrathall. Cost reduction via patient targeting and outreach: A statistical approach. In *Healthcare Informatics (ICHI), 2017 IEEE International Conference on*, pages 513–517. IEEE, 2017.

[44] Jonathan A Wrathall and Tom Belnap. Reducing healthcare costs through patient targeting: Risk adjustment modeling to predict patients remaining high-cost. *eGEMs (Generating Evidence and Methods to improve patient outcomes)*, 5, 2017.

[45] Zellig S. Harris. Distributional structure. *Word*, 10:146–162, 1954.

[46] Youngduck Choi, Chill Yi-I Chiu, and David Sontag. Learning Low-Dimensional Representations of Medical Concepts. *AMIA Summits on Translational Science Proceedings*, 2016:41–50, July 2016.

[47] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. Distributed representations of words and phrases and their compositionality. In C. J. C. Burges, L. Bottou, M. Welling, Z. Ghahramani, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 26*, pages 3111–3119. Curran Associates, Inc., 2013.

[48] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient estimation of word representations in vector space. *CoRR*, abs/1301.3781, 2013.

[49] Yoav Goldberg and Omer Levy. word2vec explained: deriving mikolov et al.'s negative-sampling word-embedding method. *CoRR*, abs/1402.3722, 2014.

[50] Jeffrey Pennington, Richard Socher, and Christopher D. Manning. Glove: Global vectors for word representation. In *Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543, 2014.

[51] T. Christensen, A. Frandsen, S. Glazier, J. Humpherys, and D. Kartchner. Machine learning methods for disease prediction with claims data. In *2018 IEEE International Conference on Healthcare Informatics (ICHI)*, pages 467–4674, June 2018.

[52] D. Kartchner, T. Christensen, J. Humpherys, and S. Wade. Code2vec: Embedding and clustering medical diagnosis data. In *2017 IEEE International Conference on Healthcare Informatics (ICHI)*, pages 386–390, Aug 2017.

[53] Edward Choi, Andy Schuetz, Walter F. Stewart, and Jimeng Sun. Medical concept representation learning from electronic health records and its application on heart failure prediction. *CoRR*, abs/1602.03686, 2016.

[54] 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.

[55] Sepp Hochreiter, Yoshua Bengio, Paolo Frasconi, Jürgen Schmidhuber, et al. Gradient flow in recurrent nets: the difficulty of learning long-term dependencies, 2001.

[56] Tomas Mikolov, Wen-tau Yih, and Geoffrey Zweig. Linguistic regularities in continuous space word representations. In *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 746–751, 2013.

[57] Yoshua Bengio et al. Learning deep architectures for ai. *Foundations and trends® in Machine Learning*, 2(1):1–127, 2009.

[58] Roman Timofeev. Classification and regression trees (cart) theory and applications. *Humboldt University, Berlin*, 2004.

[59] Leo Breiman. Random forests. *Machine learning*, 45(1):5–32, 2001.

[60] Tianqi Chen and Carlos Guestrin. Xgboost: A scalable tree boosting system. In *Proceedings of the 22Nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '16, pages 785–794, New York, NY, USA, 2016. ACM.

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

[62] Kyunghyun Cho, Bart van Merriënboer, Dzmitry Bahdanau, and Yoshua Bengio. On the properties of neural machine translation: Encoder-decoder approaches. *arXiv preprint arXiv:1409.1259*, 2014.

[63] L.J.P. van der Maaten and G.E. Hinton. Visualizing high-dimensional data using t-sne. 2008.

[64] Alexj Gossmann. Probabilistic interpretation of AUC.

[65] Anand Avati, Kenneth Jung, Stephanie Harman, Lance Downing, Andrew Ng, and Nigam H Shah. Improving palliative care with deep learning. In *Bioinformatics and Biomedicine (BIBM), 2017 IEEE International Conference on*, pages 311–316. IEEE, 2017.

[66] Jesse Davis and Mark Goadrich. The relationship between precision-recall and roc curves. In *Proceedings of the 23rd International Conference on Machine Learning*, ICML '06, pages 233–240, New York, NY, USA, 2006. ACM.

[67] Rajesh Ranganath, Adler Perotte, Noémie Elhadad, and David Blei. Deep survival analysis. *arXiv preprint arXiv:1608.02158*, 2016.

[68] David Arthur and Sergei Vassilvitskii. K-means++: The advantages of careful seeding. In *Proceedings of the Eighteenth Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA '07, pages 1027–1035, Philadelphia, PA, USA, 2007. Society for Industrial and Applied Mathematics.

[69] Martin K. Kuhlmann, Andreas Kribben, Michael Wittwer, and Walter H. Hrl. Optamalnutrition in chronic renal failure. *Nephrology Dialysis Transplantation*, 22(3):iii13, 2007.

[70] Healthline. Osteomalacia, 2017.

[71] F. Hildebrant. Renal medicine 1: Genetic kidney diseases. *The Lancet*, 375(9722):1287–1295.

[72] S J Ryu. Intracranial hemorrhage in patients with polycystic kidney disease. *Stroke*, 21(2):291–294, 1990.