Chapter 16
Predicting Adverse Drug Events from Electronic Medical Records

Jesse Davis, Vítor Santos Costa, Peggy Peissig, Michael Caldwell, and David Page

Abstract Learning from electronic medical records (EMR) poses many challenges from a knowledge representation point of view. This chapter focuses on how to cope with two specific challenges: the relational nature of EMRs and the uncertain dependence between a patient's past and future health status. We discuss three different approaches for allowing standard propositional learners to incorporate relational information. We evaluate these approaches on three real-world tasks where the goal is to use EMRs to predict whether a patient will have an adverse reaction to a medication.

16.1 Introduction

Personalized medicine represents a significant application for the health informatics community [13]. Its objective can be defined as follows:

Given: A patient's clinical history,
Do: Create an *individual* treatment plan.

Personalized medicine is possible due to the fundamental shift in health care practice caused by the advent and widespread use of electronic medical records (EMR). An EMR is a relational database that stores a patient's clinical history: disease diagnoses, procedures, prescriptions, lab results, and more. Figure 16.1 shows one very simplified EMR with two patients that includes phenotypic data, lab tests, diagnoses, and drug prescriptions. With EMR's relevant data residing on disk as opposed to paper charts, it is possible to apply machine learning and data mining techniques to these data to address important medical problems such as predicting which patients are most at risk for having an adverse reaction to a certain drug.

However, working with EMR data is challenging. EMR data violate some of the underlying assumptions made by classical statistical machine learning techniques, such as decision trees [19], support vector machines [22], and Bayesian networks [15]. These techniques are designed to work on propositional (tabulated) data. That is, they operate on data that resides in a single table, where each row represents a data point and the rows in the table are assumed to be independent. Namely, the obstacles of working with EMR data include:

A. Hommersom and P.J.F. Lucas (eds.), *Biomedical Knowledge Representation*, LNAI 9521, DOI 10.1007/978-3-319-28007-3_16

A)

PID	Birth Date	Gender
P1	09/02/50	Female
P2	03/19/75	Male

B)

PID	Date	Lab test	Result
P1	12/23/04	Glucose	43
P1	10/25/04	Glucose	45
P2	04/17/05	Lipid panel	278

C)

PID	Date	Diagnosis
P1	02/01/01	Flu
P1	05/02/03	Bleeding
P2	04/21/05	High Cholestrol

D)

PID	Date	Medication	Dose
P1	05/01/02	Warfarin	10mg
P1	02/02/03	Terconazole	10mg
P2	04/21/05	Zocor	20mg

Fig. 16.1 A simplified electronic health record. Table **A** contains information about each patient. Table **B** contains lab test results. Table **C** lists disease diagnoses. Table **D** has information about prescribed medications.

Multiple relations: Each type of data (e.g., drug prescription information, lab test results) is stored in a different table of a database. Traditionally, machine learning algorithms assume that data are stored in a single table. For example, see the tables in Fig. 16.1.

Uncertainty: The data are inherently noisy. For example, a diagnosis code of 410 for myocardial infarction (heart attack, or MI) may be entered to explain billing for tests to confirm or rule out an MI, rather than to indicate that the patient definitely had an MI on this date. It might even be entered to indicate that an earlier MI is relevant to today's visit.

Non-deterministic relationships: It is important to model the uncertain, non-deterministic relationships between patients' clinical histories and current and future predictions about their health status.

Differing quantities of information: Different patients may have dramatically different numbers of entries in any given EMR table, such as diagnoses or vitals.

Missing and/or incomplete data: Patients switch doctors and clinics over time, so a patient's entire clinical history is unlikely to reside in one database. Furthermore, information, such as the use of over-the-counter drugs, may not appear in the clinical history. In addition, patients rarely return to report when a given condition or symptom ceased, so this information is almost always missing.

Schema not designed to empower learning: Clinical databases are designed to optimize ease of data access and billing rather than learning and modeling.

Large amounts of data: As more clinics switch to electronic medical records, the amount of data available for analysis will exceed the capability of current machine learning techniques.

Longitudinal data: Working with data that contains time dependencies introduces several problems. The central problem we had to address in our work was deciding which data to include in our analysis.

These points raise interesting questions for knowledge representation, especially as they have an effect on the applicability of machine learning and data mining techniques. This chapter will focus on the first two challenges: how to effectively represent uncertainty given the multi-relational nature of the data.

We will discuss three different strategies for learning statistical models from relational data. The first approach, known as *propositionalization*, is to simply handcraft a set of features which are used to represent the multi-relational EMR as a single table. Then it becomes possible to apply traditional techniques from statistical machine learning to the modified data. The second approach builds on the first by employing a *pipeline* that automatically generates a set of features, uses these features to propositionalize the data, and then performs learning on the transformed data. The third approach is more advanced in that it *integrates* feature construction, feature selection and model learning into a single process.

To illustrate and evaluate the different approaches, we focus on the important task of predicting adverse drug reactions (ADRs) from EMR data. ADRs are the fourth-leading cause of death in the United States and represent a major risk to health, quality-of-life and the economy [16]. The pain reliever Vioxx™ alone was earning US$2.5 billion per year before it was found to double the risk of a heart attack and was pulled from the market while other similar drugs remain on the market [7]. Additionally, accurate predictive models for ADRs are actionable. If a model is found to be accurate in a prospective trial, it could be used to avoid giving a drug to those at highest risk of an ADR. Using three real-world ADR tasks, we find that the dynamic approach results in the best performance on two of the three data sets and that the handcrafted approach works reasonably well.

16.2 Background

We briefly review Bayesian networks, which are a well-known technique for representing and reasoning about uncertainty in data. We then discuss Datalog and how it can be used to represent relational data. The rest of the chapter will make use of both of these techniques to tackle the knowledge representation problems posed by EMRs.

16.2.1 Bayesian Networks

Bayesian networks [15] are probabilistic graphical models that encode a joint probability distribution over a set of random variables, where each random variable corresponds to an attribute. A Bayesian network compactly represents the joint probability distribution over a set of random variables by exploiting conditional independencies between random variables. We will use uppercase letters (e.g., X) to refer to a random variable and lower case letters (e.g., x) to refer to a specific value for that random variable. Given a set of random variables $X = \{X_1, \ldots, X_n\}$, a Bayesian network $B = \langle G, \Theta \rangle$ is defined as follows. G is a directed, acyclic graph that contains a node for each variable $X_i \in X$. For each variable (node) in the graph, the Bayesian network has a conditional probability table $\theta_{X_i|Parents(X_i)}$ giving the probability

distribution over the values that variable can take for each possible setting of its parents, and $\Theta = \{\theta_{X_1}, \ldots, \theta_{X_n}\}$. A Bayesian network B encodes the following probability distribution:

$$P_B(X_1, \ldots X_n) = \prod_{i=1}^{i=n} P(X_i | Parents(X_i)). \tag{16.1}$$

The Bayesian network learning task can be formalized as follows:

Given: Data set D that contains variables $X_i, \ldots, X_n$.
Learn: Network structure G, that is, which arcs appear in the network, and $\theta_{X_i|Parents(X_i)}$ for each node in the network.

One well-known Bayesian network classification model is called tree-augmented naïve Bayes (TAN) [6]. A TAN model has an outgoing arc from the class variable to each other attribute. It also allows each non-class variable to have at most one other parent in order to capture a limited set of dependencies between attributes. To decide which arcs to include in the augmented network, the algorithm does the following:

1. Construct a complete graph G_A, between all non-class attributes A_i. Weight each edge between i and j with the conditional mutual information, $CI(A_i, A_j|C)$.
2. Find a maximum weight spanning tree T over G_A. Convert T into a directed graph B. This is done by picking a node and making all edges outgoing from it.
3. Add an arc in B connecting C to each attribute A_i.

In step 1, CI represents the conditional mutual information, which is given by the following equation:

$$CI(A_i; A_j|C) = \sum_{a_i}^{A_i} \sum_{a_j}^{A_j} \sum_{c}^{C} P(a_i, a_j, c) log \frac{P(a_i, a_j|c)}{P(a_i|c)P(a_j|c)}. \tag{16.2}$$

This algorithm for constructing a TAN model has two nice theoretical properties [6]. First, it finds the TAN model that maximizes the log likelihood of the network structure given the data. Second, it finds this model in polynomial time.

16.2.2 Datalog

Datalog is a subset of first-order logic whose alphabet consists of three types of symbols: constants, variables, and predicates. *Constants* (e.g., the drug name `propranolol`), which start with a lowercase letter, denote specific objects in the domain. *Variable* symbols (e.g., `Disease`), which start with an uppercase letter, range over objects in the domain. *Predicate* symbols `P`/n, where n refers to the arity of the predicate and $n \geq 0$, represent relations among objects. An *atom* is $P(t_1, \ldots, t_n)$ where each t_i is a constant or variable. A *ground atom* is an atom

where each t_i is a constant. A *literal* is an atom or its negation. A *clause* is a disjunction over a finite set of literals. A *definite clause* is a clause that contains exactly one positive literal. Definite clauses are often written as an implication $B \implies H$, where B is a conjunction of literals called the body and H is a single literal called the head. The following is an example of a definite clause:

$$\texttt{Drug}(\texttt{Pid}, \texttt{Date1}, \texttt{terconazole}) \wedge \texttt{Weight}(\texttt{Pid}, \texttt{Date1}, \texttt{W}) \\ \wedge \texttt{W} < 120 \Rightarrow \texttt{ADR}(\texttt{Pid}).$$

All variables in a definite clause are assumed to be universally quantified.

Non-recursive[1] Datalog, in combination with a closed-world assumption, is equivalent to relational algebra/calculus. Therefore, it is natural and easy to represent relational databases, such as EMRs, in Datalog. The most straightforward way to do this is to create one ground atom for each row of each table in the EMR. Consider Tables **C** and **D** in Fig. 16.1, where the data would result in the following ground atoms:

```
Diagnosis(p1, 02/01/01, flu)
Diagnosis(p1, 05/02/03, bleeding)
Diagnosis(p2, 04/21/05, high cholestrol)
...
Drug(p1, 05/01/02, warfarin, 10mg)
Drug(p1, 02/02/03, terconazole, 10mg)
Drug(p2, 04/21/05, zocor, 20mg)
...
```

16.3 Approaches

In this section we describe three different strategies for coping with the multi-relational nature of EMRs.

16.3.1 Handcrafting Features

The act of converting a relational database, such as an EMR, into a single table is known as propositionalization [8]. One simple strategy is to handcraft a set of features. While this process usually results in a loss of information, it makes it possible to

[1] A Datalog clause is non-recursive by definition if the predicate appearing in its head does not appear in its body. A Datalog program, or theory, is non-recursive if all its clauses are non-recursive.

apply standard machine learning techniques, such as Bayesian network learning, to the transformed data.

The most obvious and straightforward way to do this is to construct a set of binary features for each relevant relation in the domain. For example, consider the diagnosis relation in Fig. 16.1. In this case, one feature for each possible diagnosis code would be constructed that is true of a patient if it appears in the patient's EMR *at any point in the past.* In effect, this conversion makes the assumption that the only thing that matters about a patient's future health status is if they have ever been diagnosed with a specific disease in the past. When in the past the diagnosis was made is irrelevant. The same strategy would then be applied to the other relevant relations in the domain. In the example, this would yield one set of features about lab tests and another set of features about medications.

It is also possible to design more complicated features. One idea would be to incorporate time constraints into the features. For example, one feature could be defined that is true of a patient if he has been diagnosed with a specific disease within the past year. Another idea is to look at pairs of diseases or pairs of medications. One example is a feature that is true of a patient if he was prescribed two specific medicines at any point in the past, regardless of the prescription date (i.e., they do not need to be co-prescribed). Features could be defined that combine both time and diagnoses (or medications) in order to capture co-occurrence. For example, a feature could be proposed that is true of any patient that was prescribed two medications within three months of each other.

While simple, this approach has several potential limitations. Namely, there is a huge space of possible features to consider, and it is challenging to do this in a sensible and systematic way by hand. Furthermore, taking a more directed approach, especially when handcrafting complex features, requires significant domain expertise. Finally, even employing the simplest strategy can result in a very large number of features. For example, creating one binary feature for each diagnosis code that is true of a patient if (s)he has ever been diagnosed with that particular disease would lead to over 5,000 features alone!

16.3.2 Automatically Generating Features: A Multi-Step Approach

One way to alleviate the feature construction burden that the previous approach places on a modeler is to use an automated approach to generate the features. Note that it is possible to represent each of the features mentioned in the previous subsection as a query in Datalog. For example, the query `Diagnosis(Pid, _, flu)` would return the set of all patients that have ever been diagnosed with the `flu`. Essentially, this corresponds to using the body of a definite clause, whose head is the target concept, to define a feature. This insight suggests that one possibility is to employ techniques from the field of inductive logic programming (ILP) [11]. The goal of ILP is to learn hypotheses expressed as definite clauses in first-order logic. ILP is appropriate for

learning in multi-relational domains because the learned rules are not restricted to contain fields or attributes from a single table in a database. Commonly-used ILP systems include FOIL [20], Progol [14] and Aleph [21].

The ILP learning problem can be formulated as follows:

Given: Background knowledge B, a set of positive examples E^+, and a set of negative examples E^- all expressed in first-order definite clause logic.

Learn: A hypothesis H, which consists of definite clauses in first-order logic, such that $B \wedge H \models E^+$ and $B \wedge H \not\models E^-$.

In practice, it is often not possible to find either a pure rule or rule set. Thus, ILP systems relax the conditions that $B \wedge H \models E^+$ and $B \wedge H \not\models E^-$. Typically, this is done by allowing H to cover a small number of negative examples. That is, $B \wedge H \models E'^-$, where $E'^- \subset E^-$ and the goal is to make $|E'^-|$ as small as possible.

ILP systems learn rules for a fixed target concept, such as `ADR(Pid)`, by iteratively learning rules one at a time. Thus, the central procedure is learning a single definite clause. This is usually posed as the problem of searching through the space of possible clause bodies. We briefly describe the general-to-specific, breadth-first search through the space of candidate clauses used by the Progol algorithm [14]. First, a random positive example is selected to serve as the seed example. To guide the search process, it constructs the *bottom clause* by finding all facts that are relevant to the seed example. Second, a rule is constructed that contains just the target attribute, such as `ADR(Pid)`, on the right-hand side of the implication. This means that the feature matches all examples. Third, candidate clause bodies are constructed by adding literals that appear in the bottom clause to the left-hand side of the rule, which makes the feature more specific (i.e., it matches fewer examples). Restricting the candidate literals to those that appear in the bottom clause helps limit the search space while guaranteeing that each generated refinement matches at least one example.

Employing ILP to learn the feature definitions gives rise to the following procedure. In the first step, ILP is employed to learn a large set of rules. In the second step, each learned rule is used to define a binary feature. The feature receives a value of one for an example if the data about the example satisfies (i.e., proves) the clause and it receives a value of zero otherwise. This results in a single table, with one row for each example. In the third step, a classifier is learned from the newly constructed table.

16.3.3 VISTA: An Integrated Approach

Next, we describe VISTA [4], an alternative approach that is based on the idea of constructing the classifier as we learn the rules. VISTA integrates feature construction, feature selection, and model construction into one, dynamic process. Consequently, this approach scores rules by how much they improve the classifier, providing a tight coupling between rule generation and rule usage.

Like the multi-step approach described in the previous subsection, VISTA uses definite clauses to define features for the statistical model. VISTA starts by learning

a model M over an empty feature set FS. This corresponds to a model that predicts the prior probability of the target predicate. Then it repeatedly searches for new features for a fixed number of iterations. VISTA employs the Progol algorithm that is described in the previous section to generate candidate features.

VISTA converts each candidate clause into a feature, f, and evaluates f by learning a new model (e.g., the structure of a Bayesian network) that incorporates f. In principle, any structure learner could be used, but VISTA typically uses a tree-augmented naïve Bayes model [6]. VISTA evaluates each f by comparing the generalization ability of the current model FS versus a model learned over a feature set extended with f. VISTA does this by calculating the area under the precision-recall curve (AUC-PR) on a tuning set. AUC-PR is used because relational domains typically have many more negative examples than positive examples, and the AUC-PR ignores the potentially large number of true negative examples.[2] In each iteration, VISTA adds the feature f' to FS that results in the largest improvement in the score of the model. In order to be included in the model, f' must improve the score by a certain percentage-based threshold. This helps control overfitting by pruning relatively weak features that only improve the model score slightly. If no feature improves the model's score, then it simply proceeds to the next iteration. Algorithm 3 provides pseudocode for VISTA.

Algorithm 3. VISTA (Training Set T, Validation Set V, Maximum Iteration $iter$)

```
FS = {∅}
M =BuildTANModel(T, FS)
score =AUCPR(M, V)
repeat
  bestScore = score
  f_best = ∅
  /*Generate Candidate Features*/
  Cand = GenCandidates()
  for all (f ∈ Cand) do
    M' =BuildTANModel(T, FS ∪ f)
    score' =AUCPR(M', V)
    if (score' > bestScore) then
      f_best = f
      bestScore = score'
    end if
  end for
  if (f_best ≠ ∅) then
    FS = FS ∪ f_best
    M =BuildTANModel(T, FS)
    score =AUCPR(M, V)
  end if
until Reaching iteration iter
return: FS
```

[2] In principle, VISTA can use any evaluation metric to evaluate the quality of the model such as (conditional) likelihood, accuracy, or ROC analysis.

16.4 Empirical Evaluation

In this section, we evaluate the three approaches outlined in Sect. 16.3 on three real-world data sets. In all tasks, we are given patients that take a certain medication, and the goal is to model the patients that have a related ADR. We first describe the data sets we use and our methodology. Then we present and discuss our experimental results.

16.4.1 Task Descriptions

Our data comes from a large multi-specialty clinic that has been using electronic medical records since 1985 and has electronic data back to the early 1960s. We have received institutional review board (IRB) approval to undertake these studies. For all tasks, we have access to information about observations (e.g., vital signs, family history, etc.), lab test results, disease diagnoses, and medications. We only use patient data up to one week before that patient's first prescription of the drug under consideration. This ensures that we are building predictive models only from data generated before a patient is prescribed that drug.

The characteristics of the data for each task can be found in Table 16.1. On each task we consider only patients who took a medication, and the goal is to distinguish between patients who went on to experience an adverse event (i.e., positive examples) and those who did not (i.e., negative examples). We now briefly describe each task.

Selective Cox-2 inhibitors (e.g., Vioxx™) are a class of pain relief drugs that were found to increase a patient's risk of having a myocardial infarction (MI) (i.e., a heart attack). For the Cox-2 data set, positive examples consist of patients who had a MI after taking a selective Cox-2 inhibitor. To create a set of negative examples, we took patients that were prescribed a selective Cox-2 inhibitor and did not have an MI. Furthermore, we matched the negative examples to have the same age and gender distribution as the positive examples to control for those risk factors.

Angiotensin-converting enzyme inhibitors **(ACEi)** are a class of drugs commonly prescribed to treat high blood pressure and congestive heart failure. It is known that

Table 16.1 Data set characteristics.

	Selective Cox-2	Warfarin	ACEi
Pos. examples	160	144	102
Neg. examples	2,134	1,440	1,020
Unique drugs	2,590	2,316	2,044
Unique diagnoses	7,912	8,389	7,286
Drug facts	3,518,467	603,503	335,065
Diagnoses facts	3,653,487	691,591	436,934

in some people, ACEi may result in angioedema (a swelling beneath the skin). To create the ACEi data set, we selected all patients with at least one prescription of an ACEi drug in their electronic health record. Within this population, we defined positive examples to be those patients who have a diagnosis of angioedema at any point after their first ACEi prescription.

Warfarin is a commonly prescribed blood thinner that is known to increase the risk of internal bleeding for some individuals. To create the Warfarin data set, we selected all patients who have at least one prescription of Warfarin in their electronic health record. We defined positive examples to be those patients with a bleeding event (any of 219 distinct diagnoses in the ICD9 hierarchy representing bleeding events) at any point after their first Warfarin prescription.

16.4.2 Methodology

We perform stratified, ten-fold cross-validation for each task and compare the following algorithms.

Handcrafted. In this model, we construct a set of handcrafted features to propositionalize the EMR. We create one binary feature for each possible diagnosis code, medication, and lab test. The feature is true of a patient if the appropriate diagnose, medication, or lab test appears in the portion of the patient's EMR used for training. For each test fold, we use information gain on the training set to select the 50 most informative features. A TAN classifier is trained that uses these 50 attributes.

Multi Step. First, we use ILP to learn a set of rules on the training data. We use the Aleph ILP system [21], which is a re-implementation of the Progol algorithm [14], to learn rules. The background knowledge used to construct the rules includes diagnosis codes, medications, and lab tests as before, but also allows temporal relations between events and comparing the results of observations against a learned threshold. We run Aleph under the `induce_max` command in order to fully exploit all the training examples. Second, we create a data set by converting each rule learned by Aleph into a binary attribute, which is true of an example if the rule covers the example. Third, we train a TAN classifier over the newly transformed data set.

VISTA. We follow a greedy algorithm. Starting from a network that contains the class node only, we search for clauses that when added to a classifier will improve its performance. We define a network to be an improvement over a previous classifier if it increases the area under the precision-recall curve (AUC-PR) by at least 2 %. First, we sub-divide the nine folds in the training data into five training and four tuning folds. The training folds are used to generate the candidate classifiers. We first use these folds to discover the clauses and then to train the TAN classifiers. The tuning folds are kept separate. They are used to compute the AUC-PR for the new TAN classifier and decide whether a feature should be included in the model or not. As a stop criteria, we use an arbitrary time limit of three hours for learning each model.

All three approaches make use of a TAN classifier learning algorithm where we compute maximum likelihood parameters of the model and use Laplace smoothing to prevent zero probabilities.

When reporting results, we focus on precision-recall analysis. In precision-recall space, recall is plotted on the x-axis and precision on the y-axis. Recall (also called the true positive rate) is defined as the proportion of positive examples that are correctly classified as positive. Precision reports the fraction of examples classified as positive that are truly positive. Often times, precision-recall analysis is preferred to ROC analysis in domains, such as ours, that have a large class skew [5]. Note that in ROC analysis, a very small false positive rate can correspond to a large number of false positives, if there are a large number of negative examples. In contrast, precision-recall analysis ignores the potentially large number of true negatives. We also report the results for random guessing, which corresponds to an AUC-PR equal to the proportion of positive examples in the test set [2].

16.4.3 Results and Discussion

Table 16.2 shows the average AUC-PR for each of the tasks. First, regardless of the task, each approach also does significantly better than random guessing. Thus, each approach is picking up signal in the data. VISTA results in the best performance on two of the three tasks. This indicates that there is some benefit to using the dynamic, automated approach. The handcrafted approach also exhibits good performance, and has the best performance on the Warfarin task. Interestingly, this approach yields better results than the multi-step approach. One possible explanation is that ILP tends to be biased towards constructing a smaller set of strong, complex features whereas on this task it may be beneficial to have a larger set of weak, simple features. In the future, it is worth exploring a model that uses a combination of simple and complex features. Additionally, ILP systems generate rules that predict the positive examples. In contrast, the other two approaches are able to select features that are predictive of either the positive or negative class, which may yield a benefit.

Figures 16.2, 16.3, and 16.4 show the precision-recall curves for each task. Note that on this task, for drugs on the market it is probably more meaningful to focus on the high precision, low recall (i.e., recall ≤ 0.3) parts of the plots. This is because if we act only based on this portion of the curve then we would only change current clinical practice by denying the drug to patients who will almost all suffer the ADE if

Table 16.2 Average AUC-PR and its standard deviation for each approach. The best result for each task is shown in bold.

	Selective Cox-2	Warfarin	ACEi
VISTA	**0.614 ± 0.11**	0.171 ± 0.06	**0.328 ± 0.06**
Multi Step	0.557 ± 0.14	0.188 ± 0.09	0.261 ± 0.09
Hand Crafted	0.553 ± 0.15	**0.252 ± 0.07**	0.274 ± 0.10
Random Guessing	0.070 ± 0.00	0.091 ± 0.00	0.091 ± 0.00

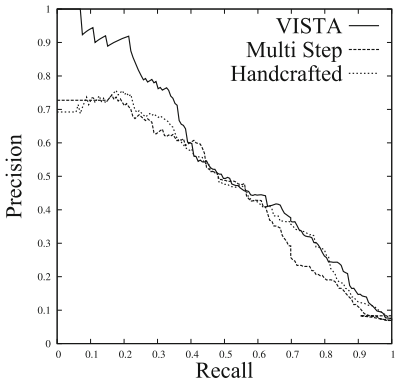

Fig. 16.2 Precision-recall curves for the Selective Cox-2 task.

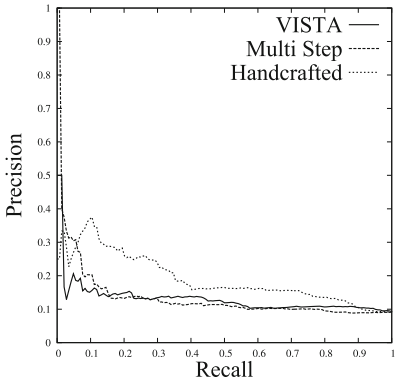

Fig. 16.3 Precision-recall curves for the Warfarin task.

they take the drug, without denying the drug unnecessarily to most individuals who need it. Exceptions to this preference to operate at the left of the PR curve would be if (1) the ADR is severe compared with the benefit of the drug, (2) there is an alternative treatment available, or (3) this is a new drug being added to the market, and we want to add it as safely as possible. Focusing on this region of PR space shows a similar picture as looking at average AUC-PR. Again, VISTA has the best performance on two tasks and the handcrafted approach does the best on the third task.

16.5 Related Work

There has been much previous work on using ILP for feature construction. Such work treats ILP-constructed rules as Boolean features, re-represents each example as a feature vector, and then uses a feature-vector learner to produce a final classifier. The first work on propositionalization is the LINUS system [12]. LINUS transforms the examples from deductive database format into attribute-value tuples and pairs these tuples to a propositional learner. LINUS primarily uses propositional algorithms

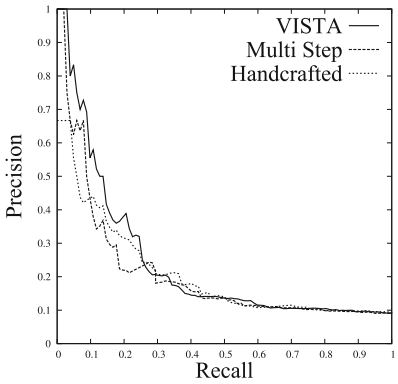

Fig. 16.4 Precision-recall curves for the ACEi task.

that generate `if-then` rules. LINUS then converts the propositional rules back into the deductive database format.

Previous work has also used ILP-learned rules as features in a propositional classifier. For example, [17] do this using a naïve Bayes classifier. Some other work, especially on propositionalization of first-order logic [1], has been developed that converts the training set to propositions and then applies feature vector techniques to the converted data. This is similar to what we do, however we first perform learning in the first-order setting to determine which features to construct. This results in significantly shorter feature vectors than in other work.

The most closely related work to VISTA includes the nFOIL [9] and kFOIL systems [10]. These systems differ in that they use different statistical learners, naïve Bayes for nFOIL and a kernel in kFOIL, and use FOIL instead of the Progol algorithm for proposing the features. Furthermore, VISTA works with AUC-PR which allows it to tackle problems that have significant class skew, which is common in medical domains. The work on structural logistic regression [18] also integrates feature generation and model selection. This work defines features using SQL queries and the statistical learner is logistic regression, but these are not especially important differences. The drawback to this approach is that it is extremely computationally expensive. In fact, they report only searching over queries that contain at most two relations. In ILP, this would be equivalent to only evaluating clauses that contain at most two literals.

In a different context, the issue of converting multiple tables into a single table is also addressed by data warehouses [3]. Typically, data warehouses often use either a star or snowflake schema. These schemas are centered on a single so-called "fact table," which is then connected to several different, multi-dimensional attributes. Each attribute value is often organized according to a hierarchy. For example, a place hierarchy may be city, county, state, and so forth. Traditionally, data warehouses focus on supporting ad-hoc user queries that produce a single table by rolling-up or drilling-down along one of the attribute dimensions. This is in constrast to our focus on building predictive models from data. Additionally, we make no assumption about the schema of data and the work presented in this chapter automatically constructs a single table.

16.6 Conclusions

This chapter addressed the challenges associated with learning statistical models from multi-relational electronic medical record (EMR) data. Specifically, we discussed how to construct features from the multi-relational EMR that can be used by a standard statistical machine learning algorithm such as Bayesian networks. We presented three different approaches: handcrafting a set of features, a multi-step algorithm that automatically learns features, and an integrated algorithm that combines feature construction with model learning. Empirically, we report results on predicting three ADRs from real-world EMR data. We found that the dynamic approach performed the best on two of the three tasks and that handcrafting the features also yielded good results.

Acknowledgements JD is partially supported by the Research Fund K.U.Leuven (CREA/11/015 and OT/11/051), EU FP7 Marie Curie Career Integration Grant (#294068) and FWO-Vlaanderen (G.0356.12). VSC is funded by ERDF through Programme COMPETE and by the Portuguese Government through FCT Foundation for Science and Technology projects LEAP (PTDC/EIA-CCO/112158/2009) and ADE (PTDC/EIA-EIA/121686/2010). MC, PP, EB and DP gratefully acknowledge the support of NIGMS grant R01GM097618-01.

References

1. Alphonse, E., Rouveirol, C.: Lazy propositionalisation for relational learning. In: 14th European Conference on Artificial Intelligence, pp. 256–260. IOS Press (2000)
2. Boyd, K., Davis, J., Page, D., Costa, V.S.: Unachievable region in precision-recall space and its effect on empirical evaluation. In: Proceedings of the 29th International Conference on Machine Learning. Omnipress (2012)
3. Chaudhuri, S., Dayal, U.: An overview of data warehousing and OLAP technology. SIGMOD Rec. **26**(1), 65–74 (1997)
4. Davis, J., Ong, I., Struyf, J., Burnside, E., Page, D., Costa, V.S.: Change of representation for statistical relational learning. In: Proceedings of the 20th International Joint Conference on Artificial Intelligence, pp. 2719–2726 (2007)
5. Davis, J., Goadrich, M.: The relationship between precision-recall and ROC curves. In: Proceedings of the 23rd International Conference on Machine learning, pp. 233–240. ACM Press (2006)
6. Friedman, N., Geiger, D., Goldszmidt, M.: Bayesian networks classifiers. Mach. Learn. **29**, 131–163 (1997)
7. Kearney, P., Baigent, C., Godwin, J., Halls, H., Emberson, J., Patrono, C.: Do selective cyclooxygenase-2 inhibitors and traditional non-steroidal anti-inflammatory drugs increase the risk of atherothrombosis? meta-analysis of randomised trials. BMJ **332**, 1302–1308 (2006)
8. Kramer, S., Lavrac, N., Flach, P.: Propositionalization approaches to relational data mining. In: Džeroski, S., Lavrac, N. (eds.) Relational Data Mining Part III, pp. 262–291. Springer, Heidelberg (2001)
9. Landwehr, N., Kersting, K., and Raedt, L. D. nFOIL: Integrating Naive Bayes and FOIL. In: Proceeding of the 20th National Conference on Artificial Intelligence, pp. 795–800 (2005)
10. Landwehr, N., Passerini, A., Raedt, L.D., Frasconi, P.: kFOIL: Learning simple relational kernels. In: Proceedings of the 21st National Conference on Artificial Intelligence (2006)
11. Lavrač, N., Džeroski, S. (eds.): Relational Data Mining. Springer, Heidelberg (2001)

12. Lavrač, N., Džeroski, S.: Inductive learning of relations from noisy examples. In: Muggleton, S. (ed.) Inductive Logic Programming, pp. 495–516. Academic Press, London (1992)
13. McCarty, C., Wilke, R., Giampietro, P., Wesbrook, S., Caldwell, M.: Personalized Medicine Research Project (PMRP): design, methods and recruitment for a large population-based biobank. Personalized Med. **2**, 49–79 (2005)
14. Muggleton, S.: Inverse entailment and Progol. New Gener. Comput. **13**, 245–286 (1995)
15. Pearl, J.: Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference. Morgan Kaufmann, San Mateo, California (1988)
16. Platt, R., Carnahan, R.: The US food and drug administration's mini-sentinel program. Pharmacoepidemiol. Drug Saf. **21**, 1–303 (2012)
17. Pompe, U., Kononenko, I.: Naive Bayesian classifier within ILP-R. In: De Raedt, L. (ed.), Proceedings of the 5th International Conference on Inductive Logic Programming, pp. 417–436 (1995)
18. Popescul, A., Ungar, L., Lawrence, S., Pennock, D. Statistical relational learning for document mining. In: Proceeding of the 3rd International Conference on Data Mining, pp. 275–282 (2003)
19. Quinlan, J.R.: Induction of decision trees. Mach. Learn. **1**, 81–106 (1986)
20. Quinlan, J.R.: Learning logical definitions from relations. Mach. Learn. **5**, 239–266 (1990)
21. Srinivasan, A.: The Aleph Manual (2001)
22. Vapnik, V.: The Nature of Statistical Learning Theory. Information Science and Statistics. Springer, Heidelberg (1999)