Presented at the 1998 NIPS-98 Workshop on Learning from Ambiguous and Complex Examples.

Event Prediction: Learning from Ambiguous Examples

Gary M. Weiss[*] **and Haym Hirsh**

Department of Computer Science
Rutgers University
New Brunswick, NJ 08903
gmweiss@att.com, hirsh@cs.rutgers.edu

Abstract

Event prediction is an important problem with many real-world applications. For the majority of these applications, it is not necessary to predict the exact time an event will occur—it is acceptable to predict that the event will occur within some time interval. The use of this time interval introduces ambiguity into the event prediction problem, permitting it to be viewed as a multiple-instance learning problem. We have developed timeweaver, a genetic-algorithm based learning system that is capable of solving these event prediction problems. Timeweaver handles the ambiguity in the learning problem by employing a special evaluation function. In this paper we also describe how some of the ambiguity can be eliminated by reformulating the learning problem.

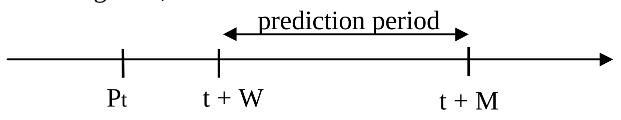
Introduction

There are many situations where one would like to learn to predict the occurrence of a specific type of event, a *target* event, before it actually occurs. Representative applications include predicting telecommunication equipment failures (Weiss & Hirsh, 1998) and predicting electric power system blackouts (Geurts & Wehenkel, 1998). The event prediction task involves taking existing event traces (which themselves contain target events) and from them *learning* a pattern that successfully predicts the target events. The learned pattern can then be applied to new event sequences. Every time a new event is received, the pattern is checked, and if the new event causes the pattern to be "completed", then a prediction of the target event is issued.

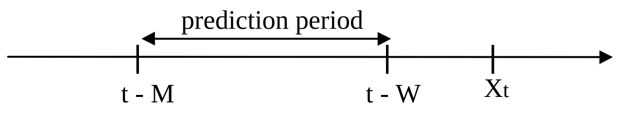
For these problems it is not necessary to predict an event the instant it occurs; rather, it is sufficient to predict it within some reasonable "prediction period". This may result in ambiguous examples, since we require a target event to be predicted only *once*, yet there may be *many* events within the interval in which a prediction must occur. This situation is equivalent to having a bag labeled positive in a multiple-instance learning problem, where only one item in the bag is required to be positive.

The Problem Formulation

An event *Et* is a timestamped observation that occurs at time *t* and is described by a set of feature-value pairs. An *event sequence* is a time-ordered sequence of events, $S = Et_1, Et_2, \ldots, Et_n$. The *target event* is the event to be predicted. We say a prediction occurring at time t, P_t, is correct if a target event occurs within its *prediction period.* As shown below, the prediction period is defined by a warning time, W, and a monitoring time, M.



The warning time, W, is the "lead time" necessary for a prediction to be useful and the monitoring time, M, determines how far into the future a target event can occur and the prediction still be considered correct. The warning time depends on the problem domain (e.g., how long it takes to replace a piece of equipment). There generally is more flexibility in assigning a value to the monitoring time. The larger the value of M the easier the prediction problem, but the less meaningful the prediction (imagine a monitoring time of 100 years). We have just defined the prediction period with respect to each prediction. For the purpose of *training* a learner, where we must classify each prediction as correct or incorrect, it is better to define the prediction period with respect to each target event. This is shown below, where the target event X occurs at time t:



A prediction is correct if it falls within the prediction period of some target event. The problem is to learn a prediction pattern P that correctly predicts the target events. A prediction is made upon observation of each event, so P: $Et_1, Et_2, .., Et_x \rightarrow \{+,-\}$, for each event Et_x. To complete the description of the problem, we need to define the semantics of a prediction. We assume that a user of our learning system, when applying it to new data, will take action (e.g., replace a piece of equipment) as soon as a *single* positive prediction occurs.

Event Prediction as Multiple-Instance Learning

The basic event prediction problem can be viewed as a multiple-instance learning problem. A positive or negative prediction occurs at each event and an example is associated with each prediction. For a target event to be predicted only a single prediction need occur within its prediction period—but multiple predictions will be possible if more than one event falls within this period. Thus, the examples that fall within the prediction period form a bag of predictions with a positive label, where only one of the examples is required to be a positive prediction. Note that in this section we have not specified what is *in* each example. While this may lead to some confusion, it is not necessary in order to understand

[*]Also AT&T Labs, Short Hills NJ 07078

how the ambiguity arises. Nonetheless, from the problem formulation one can see that each example includes the event that caused the prediction to be made as well as previous events in the sequence.

The predictions that fall between two prediction periods can be viewed as falling into a single bag of predictions with a negative label. That is, because these events do not fall within a target event's prediction period, *all* of the predictions should be negative. This is a consequence of the fact that only a single positive prediction is required to predict a target event. At this point, we have successfully described the event prediction problem as a multiple-instance learning problem.

Due to some real-world considerations, however, we cannot view the event prediction problem as a perfect multiple-instance learning problem. In multiple-instance learning, one does not distinguish between having one positive examples in a negative bag or having multiple positive examples in the bag—either way this counts as a single error. However, in our case multiple false positives may not be equivalent to a single false positive. To see why, we will consider the telecommunication domain we have studied. In this domain we replace a circuit pack once we predict it will fail. A single incorrect prediction will cause the circuit pack to be unnecessarily replaced. A second incorrect prediction may cause the circuit pack to be replaced again; thus we should count each incorrect prediction. However, it will take some time to replace this circuit pack—possibly a time equal to the warning time. So, if the second incorrect prediction occurs close to the first, it may not cause any additional action. Our strategy is to *discount* false predictions that occur close together. We have come up with a formula for doing this that takes into account some of the characteristics of the domain.

Evaluation Metrics

This section will describe statistics to evaluate the learned prediction "pattern". In order to be meaningful, these statistics will take into account the ambiguity in the examples. These statistics, shown in Figure 1, were developed to guide the search in our learning system.

We begin by introducing statistics that do *not* reflect the ambiguity in the data, and then modify them to take the ambiguity into account. The statistics we start with are *precision* and *recall*, since we would like to generate a solution that achieves both high precision and recall. For the event prediction problem, recall is defined as the percentage of target events correctly predicted and precision as the percentage of *predictions* that are correct.

$$\text{Recall} = \frac{\text{\# Target Events Predicted}}{\text{Total Target Events}} \qquad \text{Precision} = \frac{\text{TP}}{\text{TP + FP}}$$

$$\text{Normalized Precision} = \frac{\text{\# Target Events Predicted}}{\text{\# Target Events Predicted + FP}}$$

$$\text{Reduced Precision} = \frac{\text{\# Target Events Predicted}}{\text{\# Target Events Predicted + Discounted FP}}$$

TP = True Prediction FP = False Prediction

Figure 1: Evaluation Metrics

By defining recall based on the target events, rather than on the predictions, we avoid the ambiguity associated with the predictions. However, precision does not take into account the ambiguity we discussed earlier since it counts *each* prediction. *Normalized precision* eliminates the problem of counting multiple predictions of a single target event multiple times by replacing the number of correct predictions with the number of target events correctly predicted—thus "extra" predictions of a target event are ignored. Reduced precision takes into account the concerns we described in the previous section with the false predictions, by discounting the number of false predictions. There are many possible ways to discount the false predictions. We focused on the fact that a prediction is "active" for a time interval equal to its prediction period. If a second prediction occurs ½ a prediction period after the first prediction, the active period for the two predictions is only 1½ prediction periods, due to a ½ prediction period overlap. So, in this case, the discounted value of the two false predictions would equal 1½ instead of 2.

Timeweaver: An Event Prediction System

Timeweaver solves the event prediction problem by identifying predictive patterns in the event sequences. Individual patterns predict a subset of the target events with high precision and collectively the patterns cover most of the target events. This approach is implemented in *timeweaver*, a genetic algorithm-based learning system (Weiss & Hirsh, 1998). Timeweaver's search space is defined by a pattern language that allows sequential and temporal constraints to be specified between events. In order to handle the ambiguity in the examples, the genetic algorithm's evaluation function uses the recall and reduced precision statistics defined in Figure 1. The exact way in which these two statistics are combined is described in Weiss and Hirsh (1998).

Additional Sources of Ambiguity

We have shown how the event prediction problem can be viewed as a problem that requires learning from ambiguous examples and how this is a variant of the multiple-instance learning problem. Below we discuss one way in which the problem can be further complicated.

So far we have treated all correct predictions within the prediction period of a target event as being equivalent. That is, we have assumed that the *value* of a prediction is based on the step function shown in Figure 2. In the real world, however, things are generally not that simple and instead the value of a prediction might be represented by the curve.

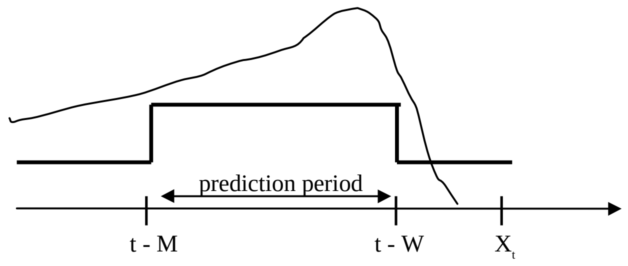


Figure 2: the value of a prediction

With this curve, the value of the prediction decreases rapidly as the prediction occurs closer to the target event. As the prediction occurs further and further away from the target event, it become less valuable, since it is less meaningful; however, the value is likely to decrease slowly. The optimal time for a prediction would fall somewhere in the prediction period associated with the simpler step function. With this formulation, there is still ambiguity. Multiple predictions may still occur, but now not all predictions have the same value. With this more complex value function, we count the first prediction, since we assume action is initiated immediately, but now we need to use the *value* returned by this first prediction. It is fairly straightforward to modify our evaluation functions to take this more complex value function into account.

Eliminating Ambiguity

Some problems are inherently ambiguous while for others the ambiguity is introduced in order to provide a representation that is suitable for a specific learning algorithm or family of learning algorithms. In this section we discuss how some of the ambiguity in the learning problem can be removed by reformulating the problem.

Reformulating the Problem

The event prediction problem can be reformulated to eliminate the "basic" ambiguity in the event prediction problem—where this ambiguity arises from the fact that a prediction is made after *each* event in the sequence. We can eliminate this ambiguity by breaking the events into examples such that there is only one example associated with each target event. Figure 3 shows how this is done.

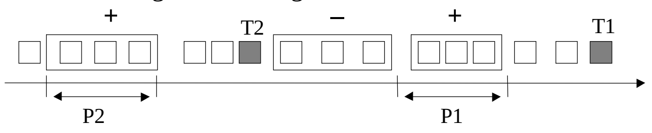


Figure 3: Eliminating Ambiguity

In the figure, each event is represented by a square and the two target events, T1 and T2, have prediction periods, P1 and P2, respectively. The events within each prediction period form a single example. This causes two positive examples to be formed. Negative examples are formed from the events between consecutive prediction periods that do not fall into the warning period of a target event. The reason that the events in the warning period are excluded is that predictions within this period are not really wrong—they are just not *useful*; if we did include them, then that might prevent us from learning the correct concept. In some cases, the negative examples may cover a very large number of events. In this case we may choose to break the large negative examples into two or more smaller examples. Depending on the learning system and the problem at hand, this may yield better results. In recent work we have broken the negative examples up so that each negative example spans a time period equal to the prediction period—thus the negative and positive examples tend to cover roughly the same number of events.

Representation Issues

In order to reformulate the problem as just described, we must be able to encode multiple events into a single example, such that the events themselves are preserved, as well as the sequential and temporal information. With timeweaver, this is trivial. Because timeweaver operates on the raw event sequence data, all we need to do is represent each example as a separate sequence of events. Another approach we are exploring is to represent the examples using first-order relations, and then use FOIL, a relational learner, to learn to predict the target events.

It is worth pointing out that it is difficult to encode multiple events as a single example using a learner, like C4.5, that requires a propositional representation. One way of doing this encoding is to generate new features to encode sequence information (e.g., n-grams), but this may result in a tremendous number of features, which may include useless information while still missing important sequence and temporal information. Other methods for doing this type of encoding have also been investigated (Dietterich and Michalski, 1985).

Discussion

Although this paper focuses on learning from ambiguous examples, we would like to point out that until recently, we did not explicitly think of our work in this manner. We simply developed a system to solve a particular problem that seemed interesting because it did not easily fit into the standard machine learning paradigms. However, we believe that there is value in explicitly thinking in terms, especially if it brings more attention to a class of interesting, real-world, problems that may not have received sufficient attention.

We would also like to point out that the ability to specify the evaluation function was critical in developing our "multiple-instance" learner. Although all heuristic learners have some form of evaluation function, many off-the-shelf learners do not permit the user to modify the evaluation function, except in some limited, predefined, way. We believe this will become an increasingly important issue as machine learning is applied to a wider range of problems. It is also important to note that for the event prediction problem, and probably for many other problems, it is critical to understand the domain under study in order to determine if examples are ambiguous, and if so, the nature of the ambiguity.

References

Dietterich, T., and Michalski, R. 1985. Discovering patterns in sequences of events, *Artificial Intelligence*, 25:187-232.

Geurts P. & Wehenkel, L. 1998. Early Prediction of Electric Power System Blackouts by Temporal Machine Learning. In *Predicting the Future: AI Approaches to Time-Series Problems, papers from the 1998 Workshop*, Technical Report WS-98-07, AAAI Press, 21-28.

Weiss, G. M., and Hirsh, H. 1998. Learning to Predict Rare Events in Event Sequences. In *Proceedings of the Fourth International Conference on Knowledge Discovery and Data Mining*, AAAI Press, 359-363.