

# Temporal Data Classification using Linear Classifiers

Peter Revesz\* Thomas Triplet\*\*

## Keywords

Classification Integration, Constraint Database, Datalog, Data Integration, Decision Tree, Reclassification, SVM.

---

## Abstract

Data classification is usually based on measurements recorded at the same time. This paper considers temporal data classification where the input is a temporal database that describes measurements over a period of time in history while the predicted class is expected to occur in the future. We describe a new temporal classification method that improves the accuracy of standard classification methods. The benefits of the method are tested on weather forecasting using the meteorological database from the Texas Commission on Environmental Quality and on influenza using the Google Flu Trends database.

---

## 1 Introduction

Data classifiers, such as support vector machines or SVMs [28], decision trees [18], or other machine learning algorithms, are widely used. However, they are used to classify data that occur in the same time period. For example, a set of cars can be classified according to their fuel efficiency. That is acceptable because the fuel efficiency of cars is not expected to change much over time. Similarly,

---

\* Corresponding author. Department of Computer Science and Engineering, University of Nebraska-Lincoln, Lincoln, NE 68588, USA. Tel.: +1-402-472-3488; fax:+1-402-472-7767. email: revesz@cse.unl.edu

\*\*Department of Computer Science, Concordia University, Montreal, Quebec H3G 1M8, Canada. email: thomastriplet@gmail.com

we can classify a set of people according to their current heart condition. However, people’s heart condition can change over time. Therefore, it would be more interesting to classify people using the current information according to whether they are likely to develop a serious heart condition in the future.

Consider a patient who transfers from one doctor to another. The new doctor may give the patient a set of tests and use the new results to predict the patient’s prospects. The question arises whether this prediction could be enhanced if the new doctor would get the older test results of the patient. Intuitively, there are cases where the old test results could be useful for the doctor. For example, the blood pressure of a patient may be 130/80, which may be considered within normal. However, if it was 120/80 last year and 110/80 the year before, then the doctor may be still concerned about the steady rise of the patient’s blood pressure. On the other hand, if the patient’s blood pressure in the past was always around 130/80, then the doctor may be more confident of predicting the patient to be in good health. Therefore, the history of the patient is important in distinguishing between these two cases.

Nevertheless, the temporal history of data is usually overlooked in the machine learning area. There are only a few previous works that combine some kind of spatio-temporal data and classification algorithms. Qin and Obradovic [17] are interested in incrementally maintaining an SVM classifier when new data is added to a database. Therefore, [17] is not useful to predict the future health of a patient or other classes that one may want to predict for the future. Tseng and Lee [26] classify temporal data using probabilistic induction. Our earlier work [22] considered data integration and reclassification by classifiers when all the data was measured at the same time.

In this paper, we propose a new temporal classification method that instead of probabilistic induction [26] extends existing linear classifiers to deal with temporal data. Figure 1 compares the standard classifiers and the new temporal classifier method. The standard classifiers take as input the current (at time  $t$ ) values of the features in the feature space and the class label some  $n$  time units ahead (at time  $t + n$ ). The temporal classifiers take as input in addition to the current features and the class, the *history*, that is, the old values of the features up to some  $i$  time units back in time (that is, from time  $t - i$  to  $t - 1$ ).

Weather forecasting is a challenging task. It is also natural to study because the major interest is in the prediction of the weather ahead of time instead of describing the current conditions. We tested our temporal classifier on a meteorological database of the Texas Commission on Environmental Quality. At a first glance it would seem useless to look at the weather history back more than a couple of days. Surprisingly, we discovered that the history does matter more than expected and the classification can be improved if one looks back 15 days back in time.

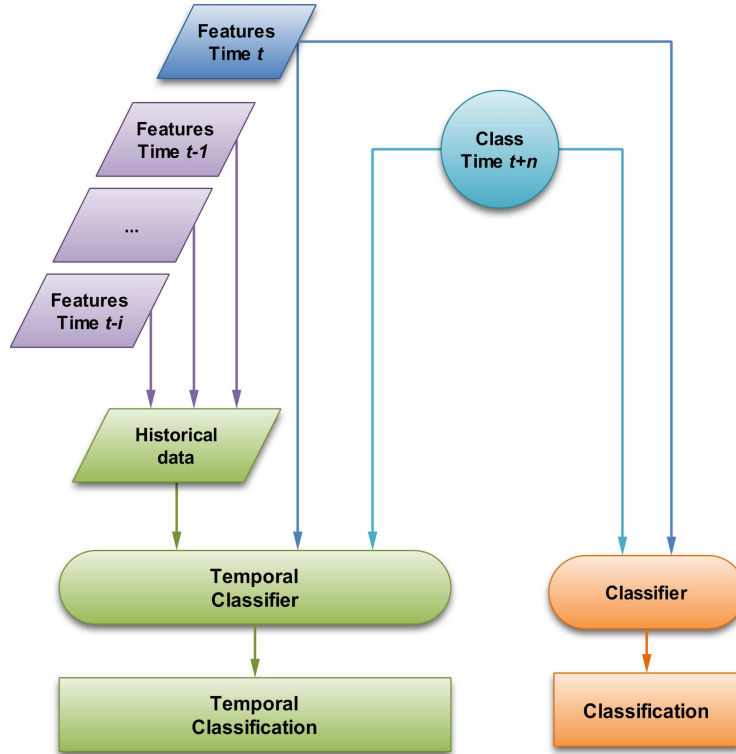


Fig. 1. Comparison of the standard and the temporal classification methods.

We were also surprised that the history of some features were considerably more useful than the history of the others. Moreover, the features that are the most important when looking at only time  $t$  are not the same as the features that are important when one looks at the weather history. That happens because the different the features have different permanency. For example, wind direction may change greatly from one hour to another. On the other hand, ozone levels are fairly constant.

The rest of the paper is organized as follows. Section 2 presents a review of classifiers and constraint databases. Section 3 describes our database representation and querying of linear classifiers. These representations are used in our implementations. Section 4 presents the new temporal classification method and a corresponding data mapping. Section 5 describes computer experiments to evaluate the performance of the temporal classification method. Section 6 compares the proposed temporal classification method with the popular IDW interpolation method. Finally, Section 7 gives some concluding remarks and open problems.

## 2 Review of Classifiers and Constraint Databases

In many problems, we need to classify items, that is, we need to predict some characteristic of an item based on several parameters of the item. Each parameter is represented by a variable which can take a numerical value. Each variable is called a *feature* and the set of variables is called a *feature space*. The number of features is the *dimension* of the feature space. The actual characteristic of the item we want to predict is called the *label* or *class* of the item.

To make the predictions, we use *classifiers*. Each classifier maps a feature space  $X$  to a set of labels  $Y$ . The classifiers are found by various methods using a set of *training examples*, which are items where both the set of features and the set of labels are known. A *linear classifier* maps a feature space  $X$  to a set of labels  $Y$  by a linear function. In general, a linear classifier  $f(\vec{x})$  can be expressed as follows:

$$f(\vec{x}) = \langle \vec{w} \cdot \vec{x} \rangle + b = \sum_i w_i x_i + b \quad (1)$$

where  $w_i \in \mathbb{R}$  are the *weights* of the classifiers and  $b \in \mathbb{R}$  is a constant. The value of  $f(\vec{x})$  for any item  $\vec{x}$  directly determines the predicted label, usually by a simple rule. For example, in binary classifications if  $f(\vec{x}) \geq 0$ , then the label is +1 else the label is -1 .

**Example 2.1** *Suppose that a disease is conditioned by two antibodies A and B. The feature space is  $X = \{\text{Antibody\_A}, \text{Antibody\_B}\}$  and the set of labels is  $Y = \{\text{Disease}, \text{No\_Disease}\}$ , where Disease corresponds to +1 and No\_Disease corresponds to -1. Then, a linear classifier is:*

$$f(\{\text{Antibody\_A}, \text{Antibody\_B}\}) = w_1 \text{Antibody\_A} + w_2 \text{Antibody\_B} + b$$

where  $w_1, w_2 \in \mathbb{R}$  are constant weights and  $b \in \mathbb{R}$  is a constant. We can use the value of  $f(\{\text{Antibody\_A}, \text{Antibody\_B}\})$  as follows:

- If  $f(\{\text{Antibody\_A}, \text{Antibody\_B}\}) \geq 0$  then the patient has Disease.
- If  $f(\{\text{Antibody\_A}, \text{Antibody\_B}\}) < 0$  then the patient has No\_Disease.

### 2.1 Support Vector Machines

Suppose that numerical values can be assigned to each of the  $n$  features in the feature space. Let  $\vec{x}_i \in \mathbb{R}^n$  with  $i \in [1..l]$  be a set of  $l$  training examples.

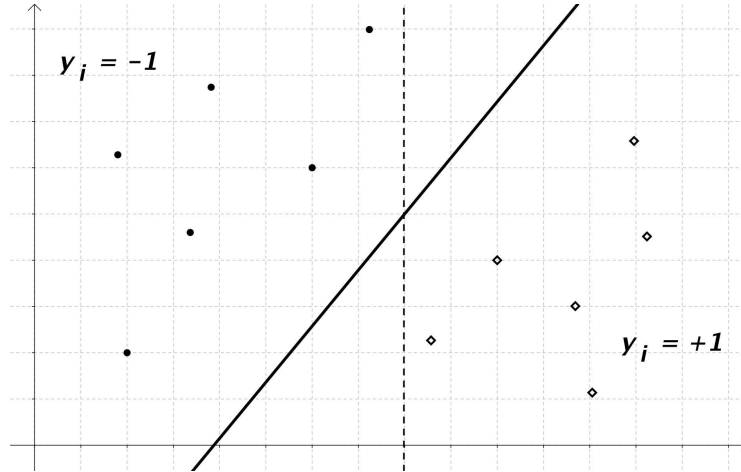


Fig. 2. A set of training examples with labels  $+1$  ( $\diamond$ ) and  $-1$  ( $\bullet$ ). This set is linearly separable because a linear decision function in the form of a hyperplane that separates all examples without error can be computed. Two possible hyperplanes that both classify the training set without error are shown (solid and dashed lines). The solid line is expected to be a better classifier than the dashed line because it has a wider *margin*, which is the distance between the closest points and the hyperplane.

Each training example  $\vec{x}_i$  can be represented as a point in the  $n$ -dimensional feature space.

*Support Vector Machines* (SVMs) [28] are increasingly popular classification tools. SVMs classify the items by constructing a hyperplane of dimension  $n - 1$  that will split all items into two sets of classes  $+1$  and  $-1$ . As shown in Figure 2, several separating hyperplanes may be suitable to split correctly a set of training examples. In this case, an SVM will construct the *maximum-margin hyperplane*, that is, the hyperplane which maximizes the distance to the closest training examples.

## 2.2 ID3 Decision Trees

Decision trees were frequently used in the nineties by artificial intelligence experts because they can be easily implemented and they provide an *explanation* of the result. A decision tree is a tree with the following properties:

- Each internal node tests an attribute.
- Each branch corresponds to the value of the attribute.
- Each leaf assigns a classification.

ID3 [18] is a greedy algorithm that builds decision trees. The ID3 decision tree and SVMs are both linear classifiers because their effects can be represented mathematically in the form of Equation (1).

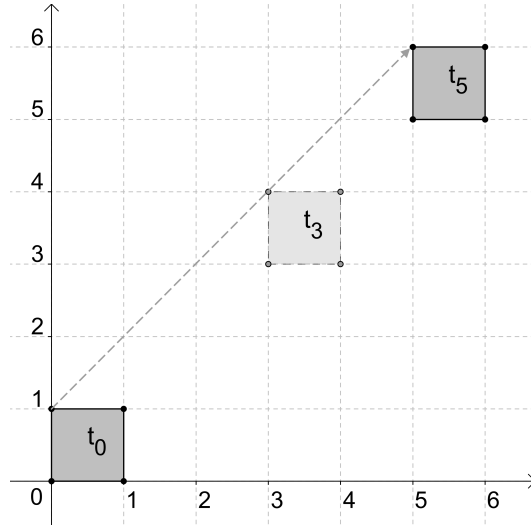


Fig. 3. A moving square.

### 2.3 Constraint Databases

Constraint databases [14, 20] form an extension of relational databases [7] where the database can contain variables that are usually constrained by linear or polynomial equations.

**Example 2.2** Figure 3 shows a moving square, which at time  $t = 0$  starts at the first square of the first quadrant of the plane and moves to the north-east with a speed of one unit per second to the north and one unit per second to the east.

#### Moving Square

X	Y	T
$x$	$y$	$t$
$x \geq t, x \leq t + 1, y \geq t, y \leq t + 1, t \geq 0$		

When  $t = 0$ , then the constraints are  $x \geq 0, x \leq 1, y \geq 0, y \leq 1$ , which is the unit square in the first quadrant. We can calculate similarly the position of the square at any time  $t > 0$  seconds. For example, when  $t = 5$  seconds, then the constraints become  $x \geq 5, x \leq 6, y \geq 5, y \leq 6$ , which is another square with lower left corner  $(5, 5)$  and upper right corner  $(6, 6)$ .

Constraint databases can be queried by both Datalog and SQL queries [1, 19, 27]. Constraint database systems include CCUBE [4], DEDALE [10], IRIS [3], and MLPQ [21].

Constraint databases, which were initiated by Kanellakis et al. [13], have many applications ranging from spatial databases [24, 6] through moving objects [11,

2] to epidemiology [23]. However, only Geist [8] and Johnson et al. [12] applied them to classification problems. In particular, both Geist [8] and Johnson et al. [12] discussed the representation of decision trees by constraint databases.

## 2.4 IDW Interpolation

When we consider a variable over some surface, the values at the unmeasured locations can be expected to be related to the values at the nearby measured locations. Based on this observation, a simple nearest-neighbor interpolation method interpolates the variable at the unmeasured location as the average of all the values at the measured points that are within a fixed radius from the unmeasured location, as shown by the solid circles around the unmeasured locations  $X$  and  $Y$  in Figure 4. A problem with this simple method is that the set of surrounding locations may contain different numbers of points, possibly zero points, in which case, the interpolated value cannot be obtained.

A slight improvement of this simple method is to vary the size of the radius to allow always a fixed minimum number of measured locations to calculate the interpolated value. If the minimum number of measured locations is 4, then the radius around the unmeasured location  $Y$  needs to be increased to the dashed circle as shown in Figure 4. This improvement still fails the natural intuition that the interpolated value should be influenced most by the nearby points and less by the more distant points within the (enlarged) circle.

The above problem is solved by the popular *Inverse Distance Weighted (IDW)* interpolation method, which weights the value of each neighboring measured location based on its distance to the unmeasured location. More precisely, the interpolation value  $v$  of the unmeasured location  $x$  is the weighted average of the values of the  $k$ -nearest measured locations  $x_i$  as defined in Equation 2, where  $v_i$  is the value of  $x_i$  and  $w_i(x)$  is the weight of  $x_i$  based on its distance to  $x$ .

$$v(x) = \frac{\sum_{i=0}^k w_i(x)v_i}{\sum_{i=0}^k w_i(x)} \quad (2)$$

The weight function is largest at zero distance and decreases as the distance increases. The most commonly used weight function (defined in Equation 3) is based on the power function and was introduced by Shepard [25], where

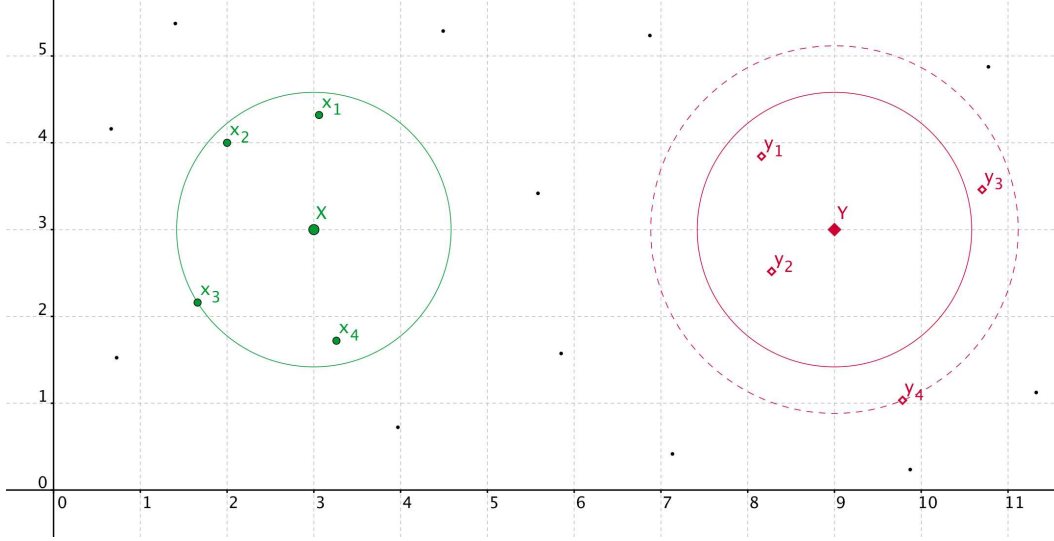


Fig. 4. Interpolation of the values at unmeasured locations  $X$  and  $Y$  can be based on the average of the values at those measured locations  $x_i$  and  $y_i$  for  $1 \leq i \leq 4$  that fall either within a fixed radius (solid circles around  $X$  and  $Y$ ) or within a varying radii circles that include a minimum of 4 measured locations (the solid circle around  $X$  and the dashed circle around  $Y$ ).

$d(x, x_i)$  is the distance between  $x$  and  $x_i$  and  $p \in \mathbb{R}^+$  is called the *power parameter*.

$$w_i(x) = \frac{1}{d(x, x_i)^p} \quad (3)$$

Li and Revesz [15] introduced a constraint database representation of IDW interpolation. This representation can be queried using a constraint database system, facilitating convenient solutions to otherwise hard problems, such as, complex spatiotemporal reasoning about epidemiological data [23]. In Section 6, we evaluate our *temporal classification* by comparing it with the constraint database representation of IDW interpolation.

### 3 Representation and Querying of Linear Classifiers

This section describes the representation of linear classifiers in constraint databases [14, 20], which were reviewed in Section 2.3. In each case, the constraint database representation can be queried using any linear constraint database system. We also describe a few typical queries that are useful for classifying new data.

### 3.1 Representation and Querying of SVMs

The Texas Commission on Environmental Quality (TCEQ) database (see Section 5.1 for details) contains weather data for over 7 years. For simplicity, consider the following smaller version with only six consecutive days, where for each day  $D$ , the features are: Precipitation  $P$ , Solar Radiation  $R$ , and Wind Speed (north-south component)  $W$ , and the label is Temperature  $T$ , which is "High" or "Low."

**Texas\_Weather**

D	P	R	W	T
1	1.73	2.47	-1.3	Low
2	0.95	3.13	9.32	High
3	3.57	3.56	4.29	Low
4	0.24	1.84	1.51	Low
5	0.0	1.19	3.77	High
6	0.31	4.72	-0.06	High

To classify the above data, we can use an SVM linear classifier. First, we need to assign a numerical value to symbolic features because SVMs are unable to handle non-numerical values. For instance, we assign the value  $t = -1$  whenever  $t = 'low'$  and  $t = +1$  whenever  $t = 'high'$ . Then, we use the *LibSVM* [5] library to build a linear classification using a SVM. Like most other SVM packages, *LibSVM* does not output the equation of the maximum-margin separating hyperplane. Instead, it returns the coordinates of the support vectors in the  $n$ -dimensional feature space. Hence, we implement routines that calculate the equation of the separating hyperplane given the support vectors. In this case the SVM can be represented by the following linear constraint relation:

**Texas\_SVM**

P	R	W	T
$p$	$r$	$w$	$t$

$$-0.442838p + 0.476746r + 2.608779w - 0.355809 = t$$

Given the  $Texas\_Weather(d, p, r, w)$  and the  $Texas\_SVM(p, r, w, t)$  relations, the following Datalog query finds for each day the distance  $t$  to the hyperplane separating the two temperature classes.

Temp\_SVM(d, t) :- Texas\_Weather(d, p, r, w), Texas\_SVM(p, r, w, t).

Finally, we can use the *SVM* relation to do the predictions, based on whether we are above or below the hyperplane.

Predict(d, y) :- Temp\_SVM(d, t), 'high' = y, t >= 0.

Predict(d, y) :- Temp\_SVM(d, t), 'low' = y, t < 0.

Instead of the above Datalog queries, one can use the logically equivalent SQL query:

```
CREATE VIEW Predict AS
  SELECT D.d, "High"
  FROM Texas_Weather as D, Texas_SVM as T
  WHERE D.p = T.p AND D.r = T.r AND D.w = T.w AND T.t >= 0
UNION
  SELECT D.d, "Low"
  FROM Texas_Weather as D, Texas_SVM as T
  WHERE D.p = T.p AND D.r = T.r AND D.w = T.w AND T.t < 0
```

### 3.2 Representation and Querying of ID3 Decision Trees

Figure 5 shows the ID3 decision tree for the *Texas\_Weather\_Data* in Section 3.1. Note that in this ID3 decision tree only the *Precipitation* feature is used. That is because the value of *Precipitation* is enough to classify the data for each day in the small database. For a larger database some precipitation values are repeated and other features need to be looked at to make a classification.

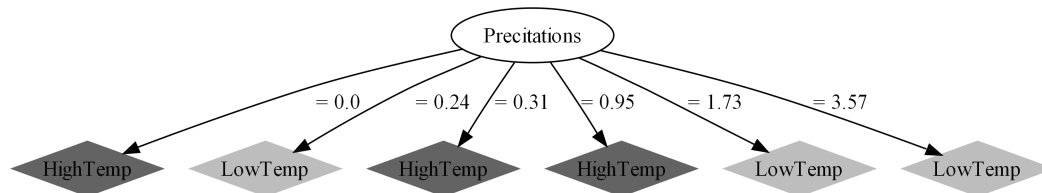


Fig. 5. Decision Tree for the prediction of the temperature using the *weather* dataset.

A straightforward translation from the ID3 decision tree in Figure 5 to a linear constraint database yields the following.

### Texas\_ID3

P	R	W	T	
$p$	$r$	$w$	$t$	$p = 1.73, t = 'Low'$
$p$	$r$	$w$	$t$	$p = 0.95, t = 'High'$
$p$	$r$	$w$	$t$	$p = 3.57, t = 'Low'$
$p$	$r$	$w$	$t$	$p = 0.24, t = 'High'$
$p$	$r$	$w$	$t$	$p = 0.0, t = 'Low'$
$p$	$r$	$w$	$t$	$p = 0.31, t = 'High'$

Given the  $Texas\_Weather(d, p, r, w)$  and the  $Texas\_ID3(p, r, w, t)$  relations, the following Datalog query can be used to predict the temperature for each day:

Predict( $d, t$ ) :- Texas\_Weather( $d, p, r, w$ ), Texas\_ID3( $p, r, w, t$ ).

Instead of Datalog queries, one can use the logically equivalent SQL query:

```
CREATE VIEW Predict AS
SELECT D.d, T.t
FROM Texas_Weather as D, Texas_ID3 as T
WHERE D.p = T.p AND D.r = T.r AND D.w = T.w
```

### 3.3 Representation and Querying of ID3-Interval Decision Trees

A straightforward translation from the original decision tree to a linear constraint database does not yield a good result for problems where the attributes can have real number values instead of only discrete values. Real number values are often used when we measure some attribute like the wind speed in miles-per-hour or the temperature in degrees Celsius.

Hence we improve the naive translation by introducing comparison constraints  $>, <, \geq, \leq$  to allow continuous values for some attributes. That is, we translate each node of the decision tree by analyzing all of its children. First, the children of each node are sorted based on the possible values of the attribute. Then, we define an interval around each discrete value based on the values of the previous and the following children. The lower bound of the interval is defined as the median value between the value of the current child and the value of the previous child. Similarly, the upper bound of the interval is defined as the median value of the current and the following children. For instance, assume we have the values  $\{10, 14, 20\}$  for an attribute for the children. This will lead

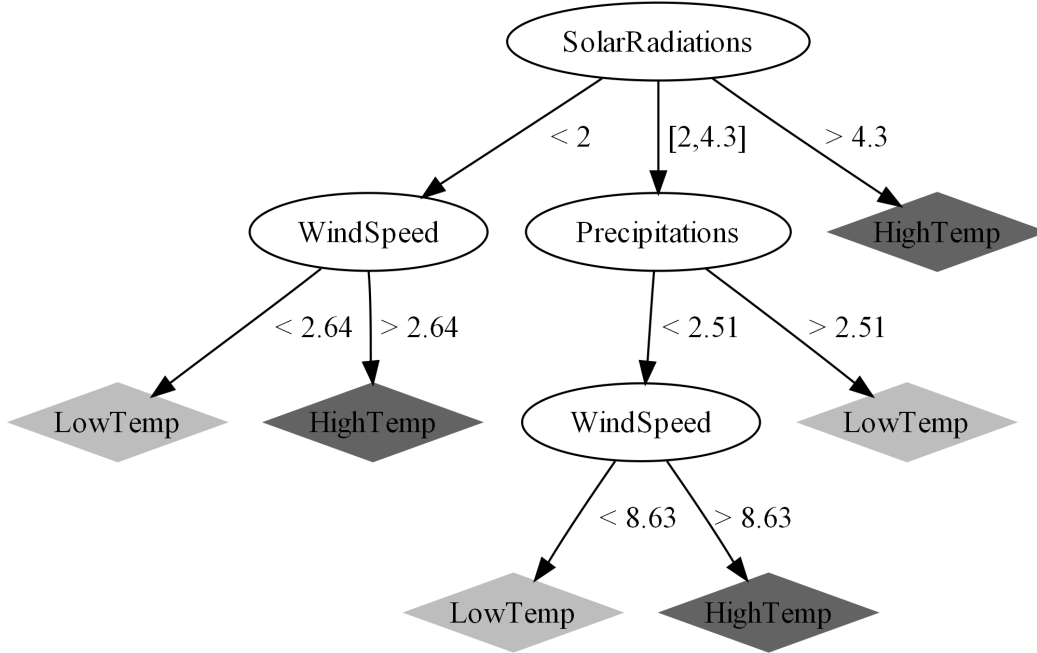


Fig. 6. Decision Tree for the prediction of the temperature using the *weather* dataset.

to the intervals  $\{(-\infty, 12], (12, 17], (17, +\infty)\}$ .

Figure 6, which shows a modified decision tree, based on the above heuristic. Translating that modified decision tree yields the following constraint relation:

### Texas\_ID3-Interval

P	R	W	T	
$p$	$r$	$w$	$t$	$r < 2, w < 2.64, t = 'Low'$
$p$	$r$	$w$	$t$	$r < 2, w \geq 2.64, t = 'High'$
$p$	$r$	$w$	$t$	$r \geq 2, r < 4.3, p < 2.51, w < 8.63, t = 'Low'$
$p$	$r$	$w$	$t$	$r \geq 2, r < 4.3, p < 2.51, w \geq 8.63, t = 'High'$
$p$	$r$	$w$	$t$	$r \geq 2, r < 4.3, p \geq 2.51, t = 'Low'$
$p$	$r$	$w$	$t$	$r \geq 4.3, t = 'High'$

The querying of ID3-Interval decision tree representations can be done like the querying of ID3 decision tree representations after replacing *Texas\_ID3* with *Texas\_ID3 – Interval*.

## 4 A Temporal Classification Method

The *Texas\_Weather* database in Section 3.1 is an atypical data for linear classifiers because it involves a temporal dimension. Although one may consider each day as an independent instance and simply ignore the temporal dimension, as we did earlier, it probably would not be the best solution. Instead, we propose below a temporal classification method for dealing with temporal data. The temporal classification method is based on an alternative representation of the database.

As an example, the *Texas\_Weather*( $d, p, r, w, t$ ) relation can be rewritten into the temporal relation

$$Texas\_Weather\_History(d, p_{d-2}, r_{d-2}, w_{d-2}, p_{d-1}, r_{d-1}, w_{d-1}, p_d, r_d, w_d, t)$$

where for any feature  $f \in \{p, r, w\}$  the  $f_i$  indicates the day  $i$  when the measurements are taken. Note that even though we did not use in *Texas\_Weather* any subscript, the implicit subscript for the features was always  $d$ . Now the subscripts go back in time, in this particular representation two days back to  $d - 1$  and  $d - 2$ . The *Texas\_Weather\_History* relation is the following.

**Texas\_Weather\_History**

$D$	$P_{d-2}$	$R_{d-2}$	$W_{d-2}$	$P_{d-1}$	$R_{d-1}$	$W_{d-1}$	$P_d$	$R_d$	$W_d$	$T$
3	1.73	2.47	-1.3	0.95	3.13	9.32	3.57	3.56	4.29	Low
4	0.95	3.13	9.32	3.57	3.56	4.29	0.24	1.84	1.51	Low
5	3.57	3.56	4.29	0.24	1.84	1.51	0.0	1.19	3.77	High
6	0.24	1.84	1.51	0.0	1.19	3.77	0.31	4.72	-0.06	High

The *Texas\_Weather\_History* relation uses the same set of feature measures as the *Texas\_Weather* relation because the data in the  $P_{d-2}, R_{d-2}, W_{d-2}$  and the  $P_{d-1}, R_{d-1}, W_{d-1}$  columns are shifted values of the  $P_d, R_d, W_d$  columns. However, when the *Texas\_Weather\_History* relation is used instead of the *Texas\_Weather* relation to generate one of the linear classifiers, then represented and queried as in Section 3, then there is a potential for improvement because each training data includes a more complete set of features.

For example, if today’s precipitation is a relevant feature in predicting the temperature a week ahead, then it is likely that yesterday’s and the day before yesterday’s precipitations are also relevant features in predicting the temperature a week ahead. That seems to be the case because the precipitation from any particular day tends to stay in the ground and affect the temperature for many more days. Moreover, since the average precipitation of three consecu-

tive days varies less than the precipitation on a single day, the former may be more reliable than the latter for the prediction of the temperature a week ahead. These intuitions lead us to believe that the alternative representation is advantageous for classifying temporal data.

In general, the alternative representation allows one to go back  $i$  number of days and look ahead  $n$  days, as outlined in Figure 1. The original representation is a representation that looks back 0 days and looks ahead the same number  $n$  of days. Therefore, the transformation from a basic to an alternative representation, which we denote by  $\implies$ , can be described as:

$$Texas\_Weather^{0,n} \implies Texas\_Weather\_History^{i,n}$$

where for any relation the first superscript is the days of historical data and the second superscript is the days predicted in the future. Based on the above ideas, Section 5.1 develops an algorithm for predicting meteorological data. Similar algorithms can be developed for other temporal data sets.

## 5 Experimental Evaluation of the Temporal Classification Method using the TCEQ Data

### 5.1 Experiments with the complete TCEQ Data

We experimentally compared the regular classification and the temporal classification methods. In some experiments both the regular and the temporal classification methods used SVMs and in some other experiments both methods used decision trees. In particular, we used the SVM implementation from the LibSVM [5] library and our implementation of the ID3-Interval algorithm described in Section 3.2.

The experiments used the Texas Commission on Environmental Quality (TCEQ) database (available from <http://archive.ics.uci.edu/ml>), which recorded meteorological data between 1998 and 2004. From the TCEQ database, we used only the data for Houston, Texas and the following forty features and the class to predict.

- 1-24. **sr**: hourly solar radiation measurements
- 25. **asr**: average solar radiation
- 26. **ozone**: ozone pollution (0 = no, 1 = yes)
- 27. **tb**: base temperature where net ozone production begins
- 28-30. **dew**: dew point (at 850, 700 and 500 hPa)
- 31-33. **ht**: geopotential height (at 850, 700 and 500 hPa)
- 34-36. **wind-SN**: south-north wind speed component (at 850, 700 and 500 hPa)
- 37-39. **wind-EW**: east-west wind speed component (at 850, 700 and 500 hPa)
- 40. **precip**: precipitation
- 41. **T**: temperature class to predict

For **sr**, **dew**, **ht**, **wind-SN**, **wind-EW** we use a subscript to indicate the hour or the hPa level. We also use the following procedure to predict the temperature  $T$ , where  $n$  is a training set size control parameter:

- (1) Normalize the dataset.
- (2) Randomly select 60 records from the dataset as a testing set.
- (3) Randomly select  $n$  percent of the remaining records as a training set.
- (4) Build a SVM, ID3, or ID3-Interval classification using the training data.
- (5) Test the accuracy of the classification on the testing set.

In step (1), the data was normalized by making for each feature the lowest value to be  $-1$  and the highest value to be  $+1$  and proportionally mapped into the interval  $[-1, +1]$  all the other values. This normalization was a precaution against any bias by the classifications. The normalization also allowed a clearer comparison of the SVM weights of the features.

For testing the regular classifiers, we used the above procedure with  $\text{TCEQ}^{0,2}$ , which we obtained from the original  $\text{TCEQ}^{0,0}$  database by shifting backwards by two days the  $T$  column values. For testing the temporal classifiers, we made the transformation

$$\text{TCEQ}^{0,2} \implies \text{TCEQ}^{15,2}$$

as described in Section 4.

Figure 7 reports the average results of repeating the above procedure twelve times for  $n$  equal to 5, 15, 25,  $\dots$ , 95 using the original ID3 algorithm. The vertical line segments show the standard deviation. We assess the statistical significance of the improvement by performing a paired T-test. The  $p$  (the observed significance level) and  $\delta$  (the average improvement) values, which are given in the caption, show a statistically very significant improvement

( $p < 1\%$ ). Similarly, Figure 8 reports the average results using SVMs.

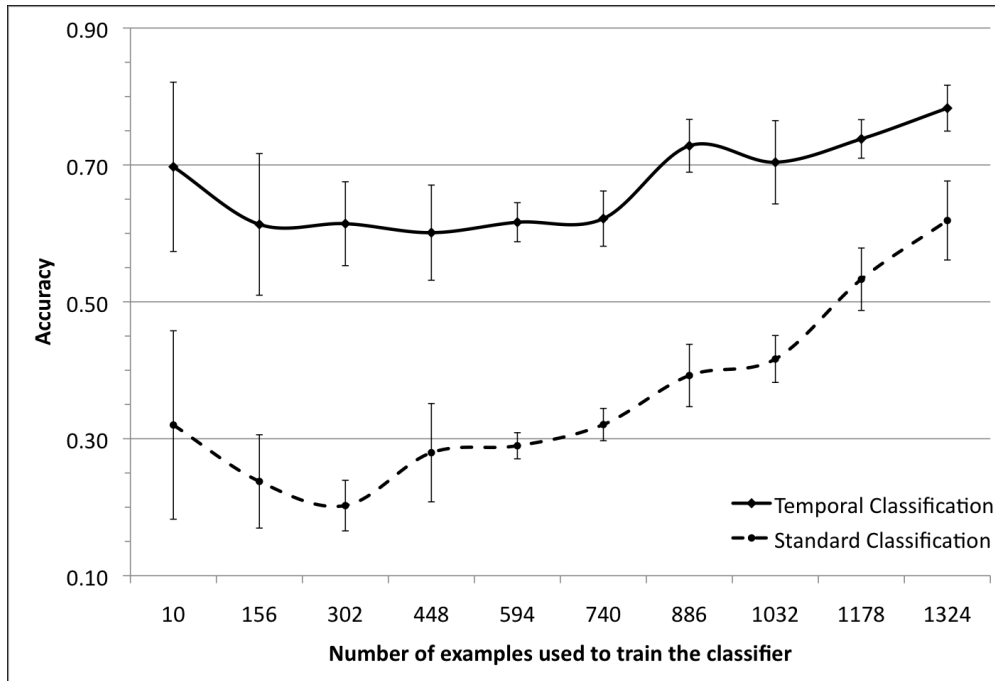


Fig. 7. Comparison of regular and temporal classification using 40 features and ID3. T-test  $p = 0.01\%$  and  $\delta = 31.1\%$

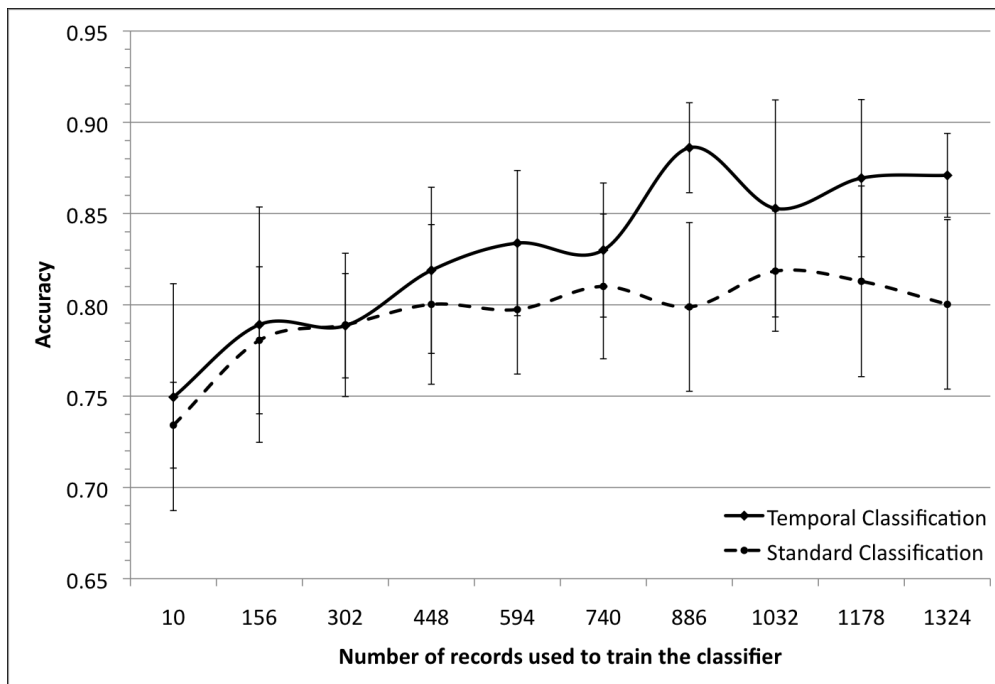


Fig. 8. Comparison of regular and temporal classification using 40 features and SVMs. T-test  $p = 0.04\%$  and  $\delta = 4.93\%$

The experiments show that adding the historical data significantly improves the temperature predictions using both the ID3 and the SVM algorithms. Moreover, the SVM algorithm performed better than the original ID3 algorithm, although the ID3-Interval algorithm (not shown) gave some improvements.

## 5.2 Experiments with Reduced TCEQ Data

Databases with a large number of features often include many noisy variables that do not contribute to the classification. The TCEQ database also appears to include many noisy variables because the SVM placed small weights on them. Since we normalized the data, the relative magnitudes of the SVM weights correspond to the relative importance of the features. In particular, the following numerical features had the highest weights:

- 25. **asr**: average solar radiation
- 35. **wind-SN<sub>700</sub>**: south-north wind speed component at 700 hPa
- 40. **precip**: precipitation

How accurate classification can be obtained using only these three selected features? These features have some interesting characteristics that make them better than other features. For example, wind-SN<sub>700</sub>, the south-north wind speed component, is intuitively more important than wind-EW<sub>700</sub>, the east-west wind speed component, in determining the temperature in Houston, Texas. In addition, the precipitation can stay in the ground for some time and affect the temperature a longer period than most of the other features. Hence our hypothesis was that these three features can already give an accurate classification.

To test this hypothesis, we conducted another set of experiments by applying the experimental procedure described in Section 5.1 to the reduced three-feature TCEQ database. The results of these experiments are shown in Figures 9 and 10. The accuracies of the classifiers based on only three features were surprisingly similar to the accuracies of the classifiers based on all forty features. In this experiment the temporal classification was again more accurate than the traditional classification.

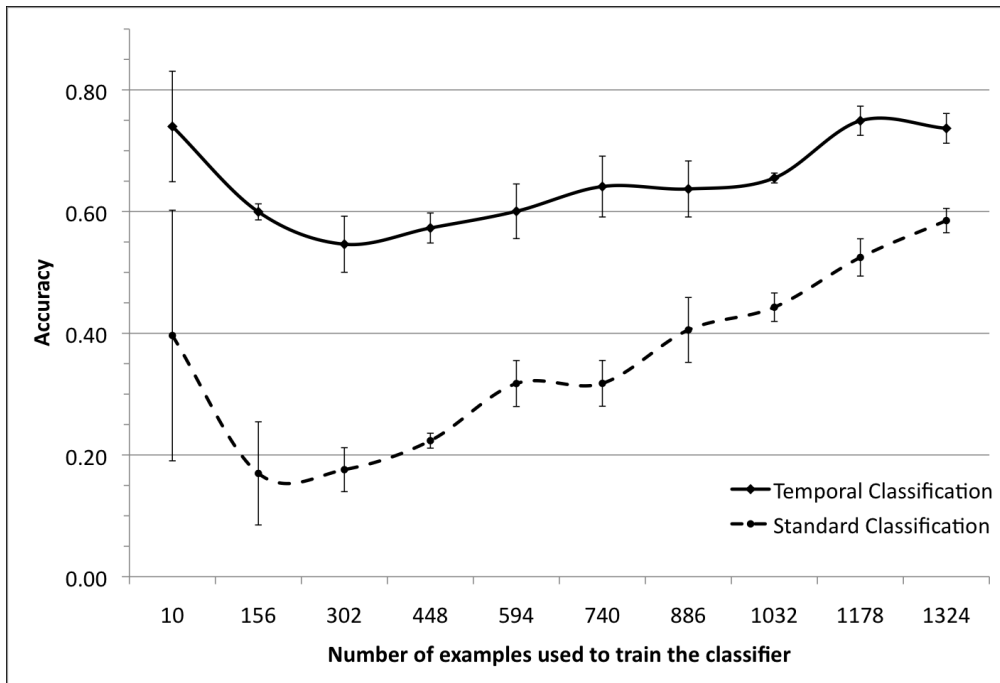


Fig. 9. Comparison of regular and temporal classification using 3 features and ID3. T-test  $p = 0.01\%$  and  $\delta = 29.17\%$

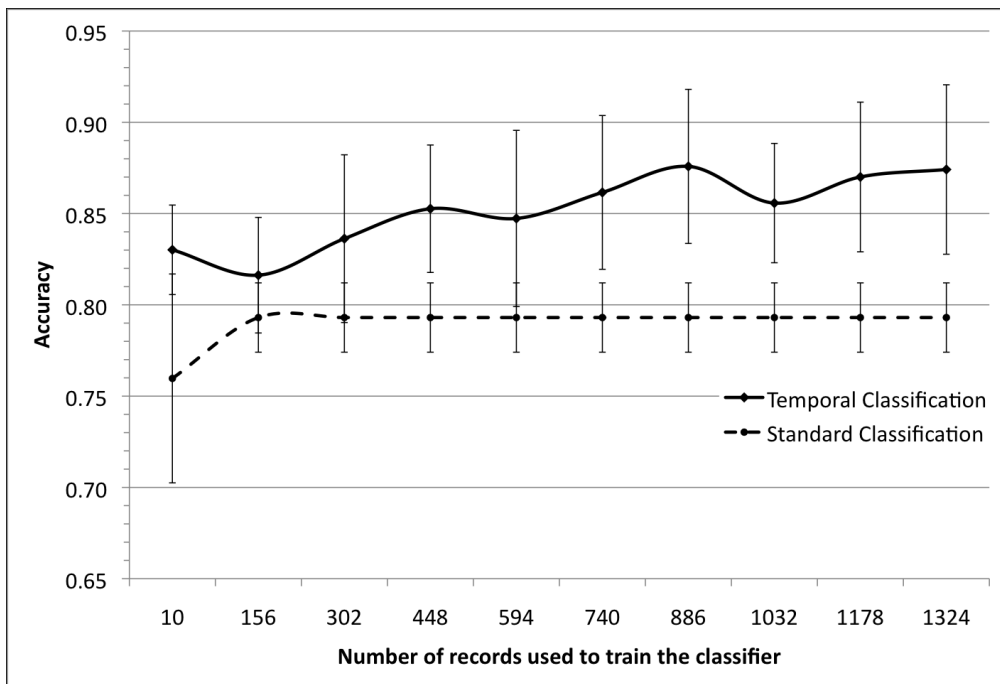


Fig. 10. Comparison of regular and temporal classification using 3 features and SVMs. T-test  $p = 0.01\%$  and  $\delta = 6.02\%$

## 6 Comparison of Temporal Classification and IDW Interpolation

### 6.1 Experiments with Influenza Temporal Data

Seasonal influenza is a major public health concern causing millions of respiratory illnesses and 250,000 to 500,000 deaths worldwide each year. If a new strain of influenza virus emerges, a pandemic could ensue with the potential to cause millions of deaths. The ongoing H1N1 (“swine flu”) pandemic is responsible for the hospitalization of nearly 1,480,000 people worldwide, causing the death of over 25,000 patients<sup>1</sup>. In the United-States alone the *Centers for Disease Control and Prevention* estimates nearly 45,000 hospitalizations and the death of 10,900 of patients<sup>2</sup>. These statistics highlight the need for early disease prediction and vaccinations, which can greatly reduce the number of people affected.

Ginsberg et al. [9] showed that the increased use of certain flu-related search terms are good indicators of flu activity. In particular, they established a strong correlation between the number of *influenza-like illness* (ILI) searches and the data collected by the *Centers for Disease Control and Prevention*. Google Flu Trends is a novel service from Google that utilizes this correlation and uses aggregated Google search data to estimate current flu activity around the world in near real-time. The data set provides weekly ILI measurements since 2003 for each U.S. state as well as national average measurements. The raw values provided by Flu Trends correspond to the number of ILI cases per 100,000 physician visits. The historical nature of this data set makes it particularly interesting for the evaluation of the temporal classification method. The database describes two variables:

- (1) **t**, the date the measurement was recorded,
- (2) **ili**, the number of ILI cases per 100,000 physician visits.

Unlike interpolation methods, SVMs and decision trees are not designed to estimate a value in the future, but assign a label to records. The IDW interpolation and temporal classification methods thus cannot be compared directly. Hence, we derived a new variable *alert* by applying a fixed threshold to *ili* values. The binary *alert* variable determines whether the current flu activity is within normal levels given the ILI value or whether an alert should be raised. The threshold was arbitrarily chosen after analyzing the distribution of ILI values. The interpolated value could be used to predict one of the two possible alert states and could therefore be compared against the temporal classification methods using the following procedure:

---

<sup>1</sup> Statistics from <http://www.flucount.org/> as of 2010/05/10.

<sup>2</sup> U.S. CDC statistics as of 2010/05/10

- (1) Normalize the dataset.
- (2) Randomly select 10% of the records from the dataset as a testing set.
- (3) Build an IDW, SVM or ID3-Interval classifier using the remaining data.
- (4) Analyze the accuracy of the classifier using the testing set.

As mentioned earlier, the prediction of disease outbreaks is critical to raise an *alert* state so that appropriate measures to prevent the disease from spreading may be taken. On the other hand, the alert should be temporary and raised only when necessary because maintaining an alert state indefinitely would be too expensive. The analysis of the overall accuracies of the classifiers is therefore not satisfying, and the classifier should be evaluated in terms of *sensitivity* and *specificity*. The sensitivity – or true positive rate – is the probability that an alert is predicted when there is actually an alert state. The specificity – or false positive rate – is the probability that the alert state will not be raised when it is unnecessary.

In order to distinguish the two alert states, the predictions of the two methods were analyzed using a Receiver Operating Characteristic (ROC) curve analysis [16, 29]. A ROC curve represents the sensitivity as a function of the specificity. The ROC curve analysis is specially useful when the distribution of the classes to predict is unbalanced as it is the case with the flu dataset. The overall performance of a specific classifier is defined as the *Area Under the Curve (AUC)*. This simple value is representative of both the sensitivity and the specificity of the classifier. The AUC normally ranges from 0.5, which corresponds to a classifier that randomly classifies items to 1, which corresponds to a perfect classifier.

For testing the three methods, we used the data of the past six weeks to predict the alert state two weeks ahead of time. Hence, we made the following transformation:

$$\text{FLU}^{0,0} \implies \text{FLU}^{6,2}$$

as described in Section 4. For the IDW interpolation method the power 0.3 was empirically chosen to maximize the performance. Figure 11 shows the ROC curves for the IDW interpolation algorithm and the temporal classification method applied to SVMs and decision trees.

As a control, we also ran experiments on  $\text{FLU}^{0,2}$ . For SVM the AUC was 0.95. Hence the difference in the AUCs between  $\text{FLU}^{0,2}$  and  $\text{FLU}^{6,2}$  do not seem significant because both predictions methods have a high accuracy.

## 6.2 Experiments with Influenza Spatio-Temporal Data

In addition to the national average measurements, the Flu Trends data provides weakly ILI measurements for each U.S. state. This can be extended to a

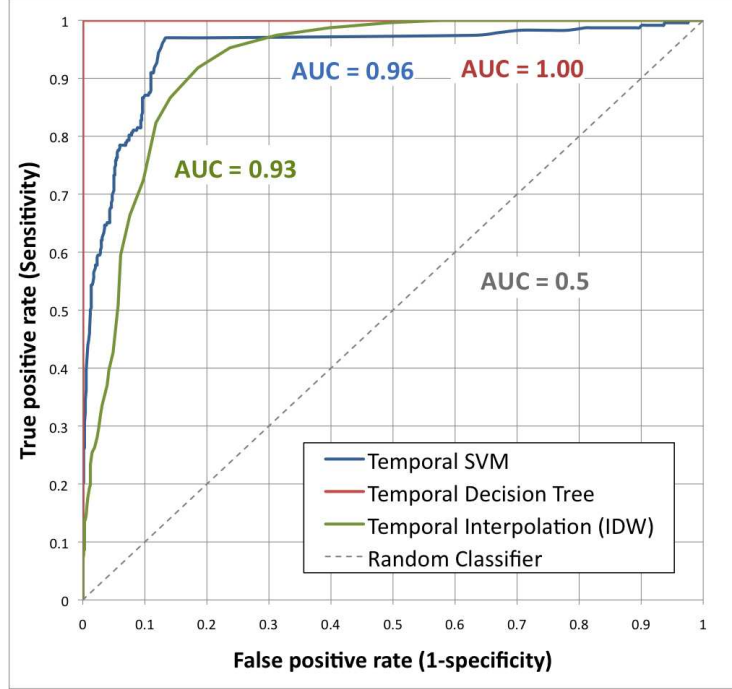


Fig. 11. Experimental results for temporal data for national flu trends showing the ROC analysis of temporal interpolation using SVMs, Decision Trees, and IDW.

spatio-temporal data set if for each state we add the geographical coordinates (latitude and longitude) of its capital city of the state. In our experiments, the database included the following four variables:

- (1)  $\mathbf{t}$ , the date the measurement was recorded,
- (2)  $\mathbf{x}$ , the longitude of the state capital city,
- (3)  $\mathbf{y}$ , the latitude of the state capital city,
- (4)  $\mathbf{ili}$ , the number of ILI cases per 100,000 physician visits.

We compared the proposed temporal classification methods using both SVMs and decision trees against the IDW spatio-temporal interpolation method on  $FLU^{6,2}$  (see Section 4) using the same procedure as in Section 6.1. The resulting ROC analysis is show in Figure 12.

### 6.3 Discussion

The results of the experiments when using temporal data only (Figure 11) show that the temporal classification method applied to decision trees was able to perfectly predict the alert state. Overall, when applied to SVMs, the temporal classification also improves the IDW interpolation because  $AUC_{SVM}$  is greater than  $AUC_{IDW}$ . In the details, the ROC analysis reveals that the IDW interpolation has a higher sensitivity. In other words, this algorithm is

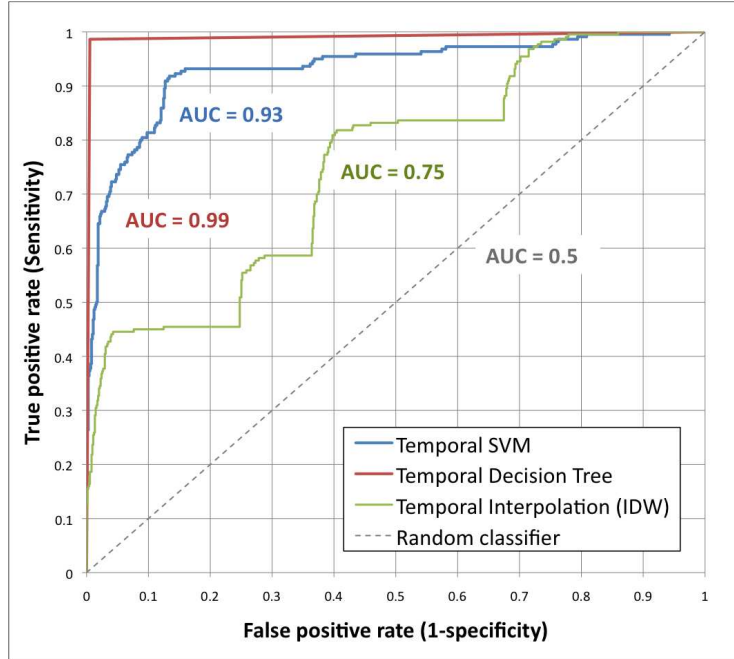


Fig. 12. Experimental results for spatio-temporal data for the U.S. states showing the ROC analysis for the spatio-temporal interpolation using SVMs, Decision Trees, and IDW.

more suitable when one needs to accurately predict when there is a risk of influenza pandemic. On the other hand, the temporal classification with SVM has a higher specificity, which means the temporal SVM classifier predicts better when the flu activity is within acceptable levels.

When considering spatio-temporal data (Figure 12), the feature space has a higher dimension. In this case, we first notice that all three classifiers have a lower AUC. However, although the performance of the two temporal classifiers is only slightly lower, the performance (AUC) of IDW interpolation has significantly diminished from 0.92 to 0.75. This result confirms the intuition that decision trees and SVMs handle better highly-dimensional feature spaces. As a result, when considering the spatio-temporal data set, our method performed significantly better than the IDW interpolation algorithm did. This result may be explained by the local variability of the flu activity in each state. If the activity varies frequently, unlike both temporal classifiers, the interpolation algorithm, which is based on the few past values only, will be unable to capture those variations and accurately predict the future values. When using the temporal data only at a national scale, the local variations are smoothed and the interpolation performs comparably to the temporal classification.

## 7 Conclusions

We proposed a new temporal data classification method. This method seems applicable in general for temporal phenomena that exhibit major trends that develop gradually over time also contain significant fluctuations in the measured values between adjacent time instances. Weather is a good example of such a temporal phenomenon because of clear warming or cooling trends over weeks or months occurring simultaneously with significant daily fluctuations.

The experiments on the TCEQ database show two major results: (1) significant accuracy improvements are obtained by using histories, and (2) no accuracy improvement is obtained by using more than three features. The experiments on an influenza data set show that our new temporal classifiers based on decision tree and SVMs are more accurate than the traditional IDW interpolation method.

A natural question is whether these experimental results also hold for other databases. Another question is whether non-linear temporal classifiers would be even more accurate than the linear temporal classifiers based on decision trees and SVMs.

## References

- [1] S. Abiteboul, R. Hull, and V. Vianu. *Foundations of Databases*. Addison-Wesley, 1995.
- [2] S. Anderson and P. Revesz. Efficient maxcount and threshold operators of moving objects. *Geoinformatica*, 13, 2009.
- [3] B. Bishop, F. Fischer, U. Keller, N. Steinmetz, C. Fuchs, and M. Pressnig. *Integrated Rule Inference System*, 2008. Software available at: [www.iris-reasoner.org](http://www.iris-reasoner.org).
- [4] A. Brodsky, V. Segal, J. Chen, and P. Exarkhopoulo. The CCUBE constraint object-oriented database system. *Constraints*, 2(3-4):245-77, 1997.
- [5] C. C. Chang and C. J. Lin. *LIBSVM: A library for support vector machines*, 2001. Software available at: [www.csie.ntu.edu.tw/~cjlin/libsvm](http://www.csie.ntu.edu.tw/~cjlin/libsvm).
- [6] J. Chomicki, S. Haesevoets, B. Kuijpers, and P. Revesz. Classes of spatiotemporal objects and their closure properties. *Annals of Mathematics and Artificial Intelligence*, 39(4):431-461, 2003.
- [7] E. F. Codd. A relational model for large shared data banks. *Communications of the ACM*, 13(6):377-87, 1970.
- [8] I. Geist. A framework for data mining and KDD. In *Proc. ACM Symposium on Applied Computing*, pages 508-13. ACM Press, 2002.
- [9] Jeremy Ginsberg, Matthew H Mohebbi, Rajan S Patel, Lynnette Bram-

- mer, Mark S Smolinski, and Larry Brilliant. Detecting influenza epidemics using search engine query data. *Nature*, 457(7232):1012–4, Feb 2009.
- [10] S. Grumbach, P. Rigaux, and L. Segoufin. The DEDALE system for complex spatial queries. In *Proc. ACM SIGMOD International Conference on Management of Data*, pages 213–24, 1998.
- [11] R. Güting and M. Schneider. *Moving Objects Databases*. Morgan Kaufmann, 2005.
- [12] T. Johnson, L. V. Lakshmanan, and R. T. Ng. The 3W model and algebra for unified data mining. pages 21–32, 2000.
- [13] P. C. Kanellakis, G. M. Kuper, and P. Revesz. Constraint query languages. *Journal of Computer and System Sciences*, 51(1):26–52, 1995.
- [14] G. M. Kuper, L. Libkin, and J. Paredaens, editors. *Constraint Databases*. Springer-Verlag, 2000.
- [15] L. Li and P. Revesz. Interpolation methods for spatiotemporal geographic data. *Computers, Environment, and Urban Systems*, 28(3):201–27, 2004.
- [16] CE Metz. Seminars in nuclear medicine; basic principles of roc analysis. 8(4):283–298, 1978.
- [17] Y. Qin and Z. Obradovic. Efficient learning from massive spatial-temporal data through selective support vector propagation. In *17th European Conference on Artificial Intelligence*, pages 526–530, 2006.
- [18] J.R. Quinlan. Induction of decision trees. *Machine Learning*, 1(1):81–106, 1986.
- [19] R. Ramakrishnan. *Database Management Systems*. McGraw-Hill, 1998.
- [20] P. Revesz. *Introduction to Constraint Databases*. Springer-Verlag, 2002.
- [21] P. Revesz, R. Chen, P. Kanjamala, Y. Li, Y. Liu, and Y. Wang. The MLPQ/GIS constraint database system. In *Proc. ACM SIGMOD International Conference on Management of Data*, 2000.
- [22] P. Revesz and T. Triplet. Reclassification of linearly classified data using constraint databases. In *12th East European Conference on Advances of Databases and Information Systems*, pages 231–245, 2008.
- [23] P. Revesz and S. Wu. Spatiotemporal reasoning about epidemiological data. *Artificial Intelligence in Medicine*, 38(2):157–70, 2006.
- [24] P. Rigaux, M. Scholl, and A. Voisard. *Introduction to Spatial Databases: Applications to GIS*. Morgan Kaufmann, 2000.
- [25] D. A. Shepard. A two-dimensional interpolation function for irregularly spaced data. In *Proc. 23rd ACM National Conference*, pages 517–524, 1968.
- [26] V. S. Tseng and C.-H. Lee. Effective temporal data classification by integrating sequential pattern mining and probabilistic induction. *Expert Systems with Applications*, 36(5):9524–9532, 2009.
- [27] J. D. Ullman. *Principles of Database and Knowledge-Base Systems*. Computer Science Press, 1989.
- [28] V. Vapnik. *The Nature of Statistical Learning Theory*. Springer-Verlag, 1995.
- [29] MH Zweig and G Campbell. Clin chem; receiver-operating characteristic

(roc) plots: a fundamental evaluation tool in clinical medicine. 39(4):561–577, 1993.