Temporal Data Classification
using Linear Classifiers

Peter Revesz and Thomas Triplet

University of Nebraska - Lincoln, Lincoln NE 68588, USA
revesz@cse.unl.edu, ttriplet@cse.unl.edu

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 is tested on weather forecasting using the meteorological database from the Texas Commission on Environmental Quality.

1   Introduction

Data classifiers, such as support vector machines or SVMs [24], decision trees [15], 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, we can classify a set of people according to their current heart condition. However, people's heart condition can change over time. Therefore, the more interesting thing would be to classify people using the current information according to whether they are likely to develop serious heart condition sometime in the future. Classifications where the current data is used for predicting the future is also used with classifiers.

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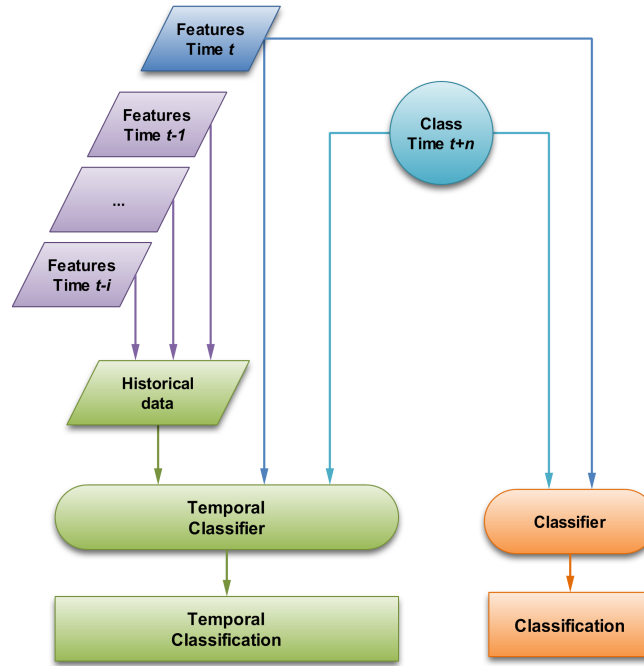
Fig. 1. Comparison of the standard and the temporal classification methods.

of spatio-temporal data and classification algorithms. Qin and Obradovic [14] are interested in incrementally maintaining an SVM classifier when new data is added to a database. Therefore, [14] 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 [22] classify temporal data using probabilistic induction. Our earlier work [19] 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 classfication method that instead of probabilistic induction (as in [22]) 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 Comission 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 in time.

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 form 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 and discusses the results. Finally, Section 6 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 $f(\overrightarrow{x})$ can be expressed as follows:

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

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

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

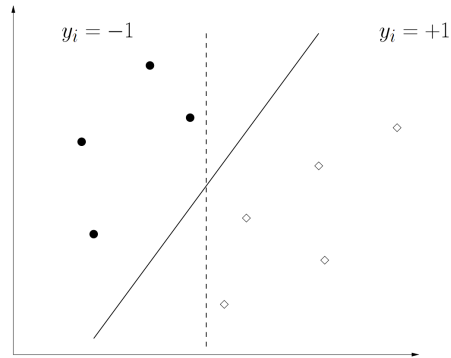$$f(\{Antibody\_A, Antibody\_B\}) = w_1 Antibody\_A + w_2 Antibody\_B + b$$

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 can be found that classifies all examples without error. 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.

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 $f(\{Antibody\_A, Antibody\_B\})$ as follows:

- If $f(\{Antibody\_A, Antibody\_B\}) \geq 0$ then the patient has *Disease*.
- If $f(\{Antibody\_A, Antibody\_B\}) < 0$ 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 $\overrightarrow{x_i} \in \mathbb{R}^n$ with $i \in [1..l]$ be a set of l training examples. Each training example $\overrightarrow{x_i}$ can be represented as a point in the n-dimensional feature space.

Support Vector Machines (SVMs) [24] 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:

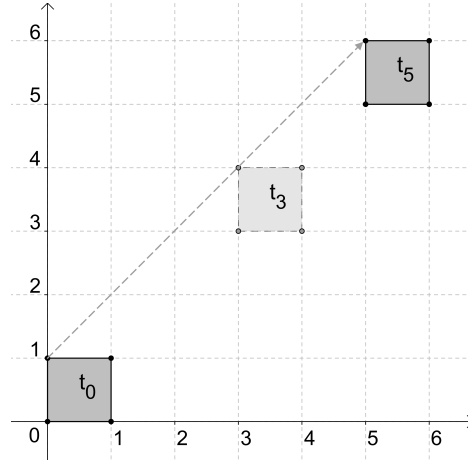- Each internal node tests an attribute.

Fig. 3. A moving square.

- Each branch corresponds to the value of the attribute.
- Each leaf assigns a classification.

ID3 [15] 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).

2.3 Constraint Databases

Constraint databases [13, 17] form an extension of relational databases [7] where the database can contain variables that are usually constrained by linear or polynomial equations.

Example 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 northeast 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, 16, 23]. Constraint database systems include CCUBE [4], DEDALE [9], IRIS [3], and MLPQ [18].

Constraint databases, which were initiated by Kanellakis et al. [12], have many applications ranging from spatial databases [21, 6] through moving objects [10, 2] to epidemiology [20]. However, only Geist [8] and Johnson et al. [11] applied them to classsification problems. In particular, both Geist [8] and Johnson et al. [11] discussed the representation of decision trees by constraint databases.

3 Representation and Querying of Linear Classifiers

This section describes the representation of linear classifiers in constraint databases [13, 17], 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 a 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 *svmlib*[5] library to build a linear classification using a SVM. That would result in a linear classifier, which 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 4 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.
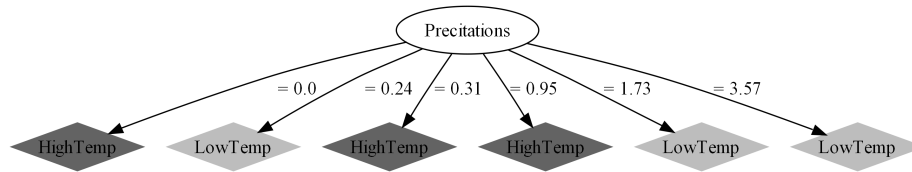

Fig. 4. Decision Tree for the prediction of the temperature using the *weather* dataset.

A straightforward translation from the ID3 decision tree in Figure 4 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 to the intervals $\{(-\infty, 12], (12, 17], (17, +\infty)\}$.

Figure 5, which shows a modified decision tree, based on the above heuristic. Translating that modified decision tree yields the following constraint relation:
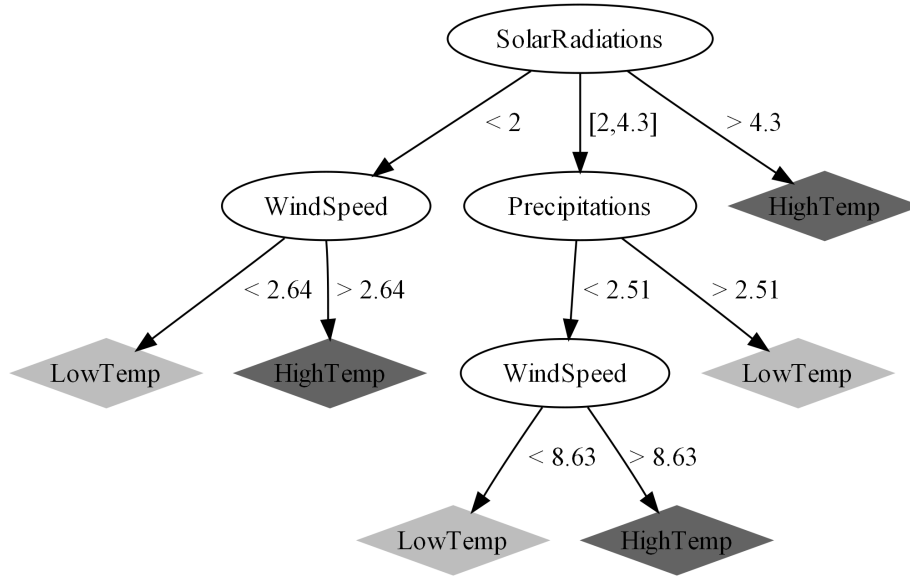
Fig. 5. Decision Tree for the prediction of the temperature using the *weather* dataset.

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 independant 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 just 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 consequtive 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. Although this seems a simple idea, it was not tried yet for decision trees or SVMs.

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 $\Longrightarrow$, can be described as:

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

where for any relation the first superscript is the days of historical data and the second superscript is the days predicted in the future.

5 Experimental Results and Discussion

5.1 Experiments with 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 SVM-Lib [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 fourty 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. **precp:** 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

$$\text{TCEQ}^{0,2} \Longrightarrow \text{TCEQ}^{15,2}$$
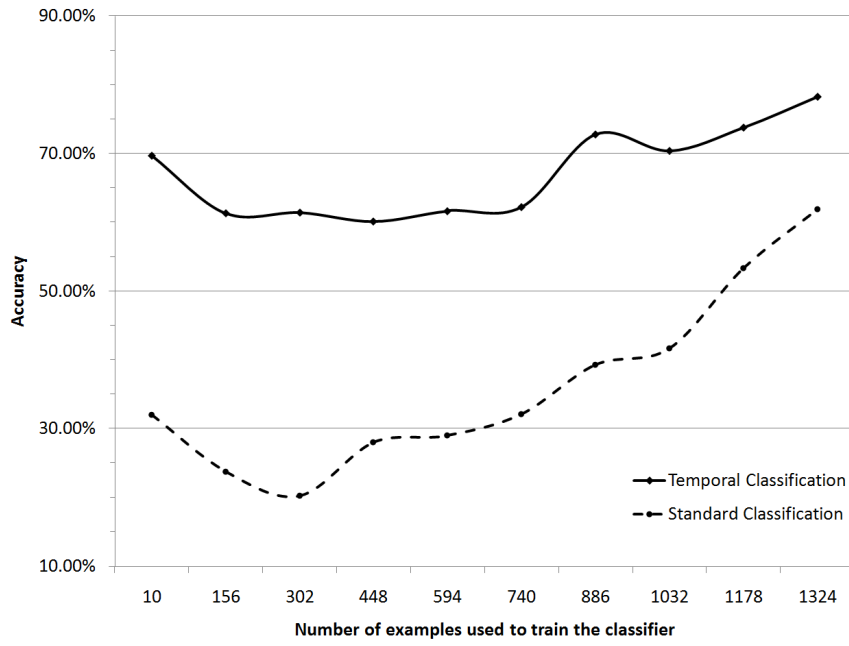
as described in Section 4.

Fig. 6. Comparison of regular and temporal classification using 40 features and ID3.
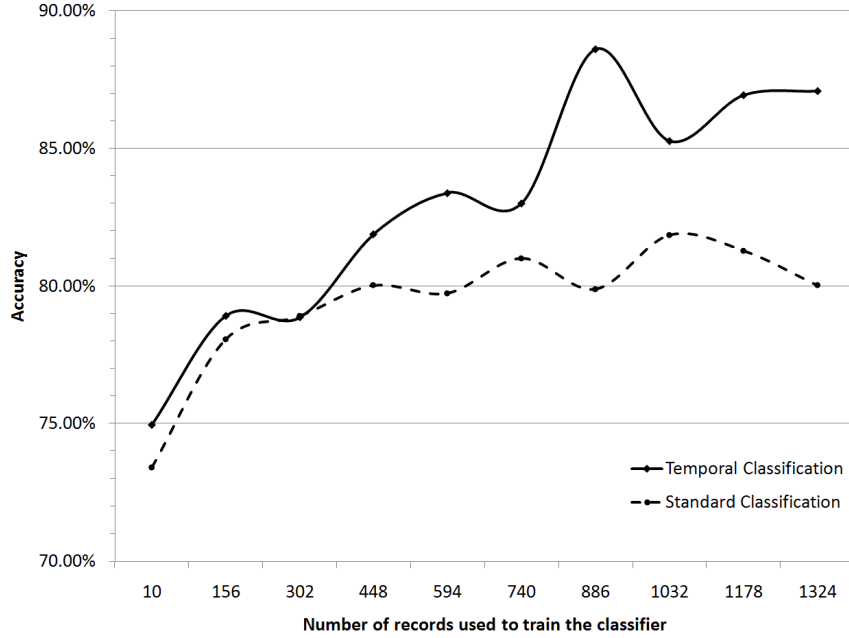

Fig. 7. Comparison of regular and temporal classification using 40 features and SVMs.

Figure 6 reports the average results of repeating the above procedure twelve times for n equal to $5, 15, 25, \ldots, 95$ using the original ID3 algorithm. Similarly, Figure 7 reports the average results using SVMs.

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$_{700}$**: south-north wind speed component at 700 hPa
40. **precp:** 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$_{700}$, the south-north wind speed component, is intuitively more important than wind-EW$_{700}$, 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 8 and 9. The accuracies of the classifiers based on only three features were surprisingly similar to the accuracies of the classifiers based on all fourty features. In this experiment the temporal classification was again more accurate than the traditional classification.

6   Conclusions

Our 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.

An interesting question is whether these results also hold for other databases. There are some other remaining questions. For example, would non-linear temporal classifiers also be better than regular non-linear classifiers? In the future, we plan to experiment with other data sets and use non-linear classifiers in addition to SVMs and decision trees.
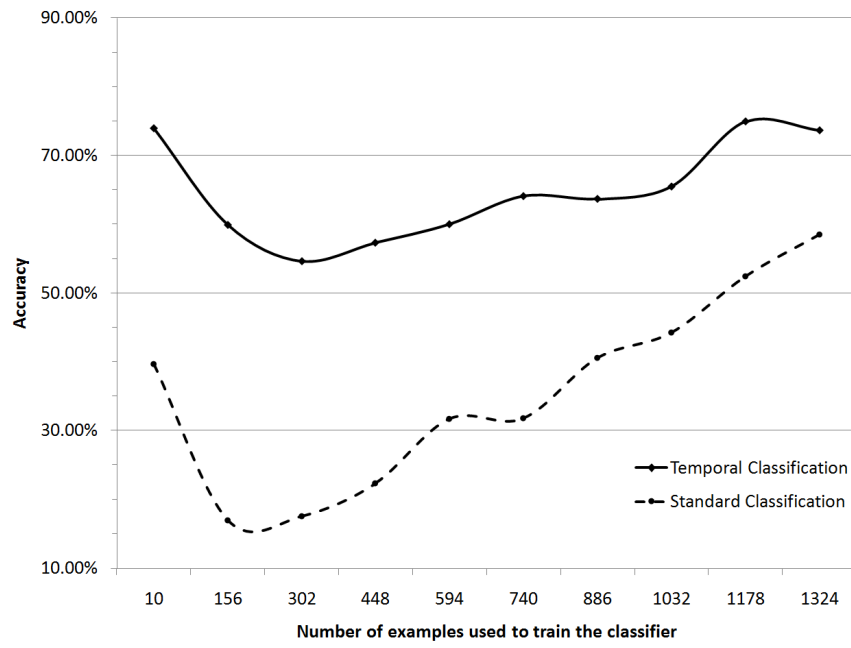
Fig. 8. Comparison of regular and temporal classification using 3 features and ID3.
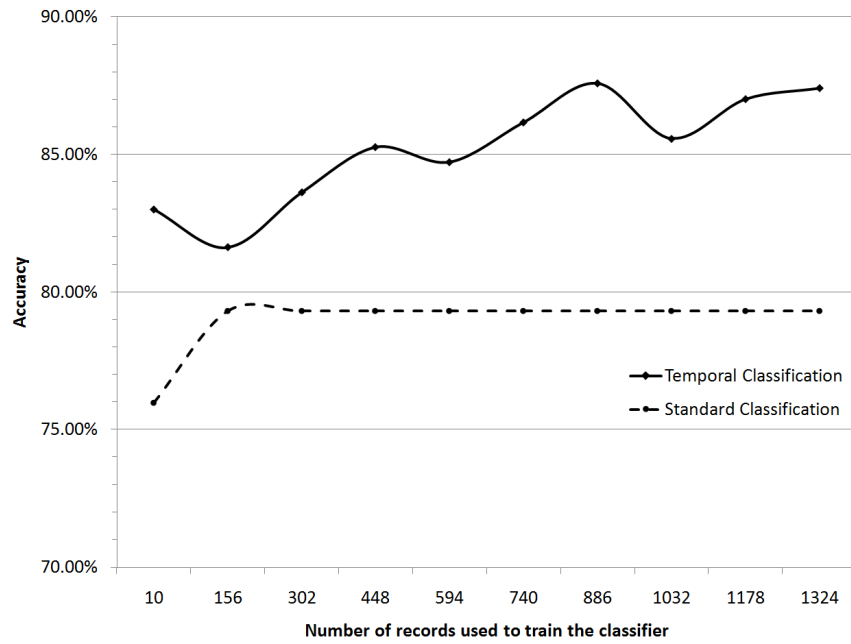

Fig. 9. Comparison of regular and temporal classification using 3 features and SVMs.

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.

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.

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. 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.

10. R. Güting and M. Schneider. *Moving Objects Databases*. Morgan Kaufmann, 2005.

11. T. Johnson, L. V. Lakshmanan, and R. T. Ng. The 3W model and algebra for unified data mining. pages 21–32, 2000.

12. P. C. Kanellakis, G. M. Kuper, and P. Revesz. Constraint query languages. *Journal of Computer and System Sciences*, 51(1):26–52, 1995.

13. G. M. Kuper, L. Libkin, and J. Paredaens, editors. *Constraint Databases*. Springer-Verlag, 2000.

14. 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.

15. J. Quinlan. Induction of decision trees. *Machine Learning*, 1(1):81–106, 1986.

16. R. Ramakrishnan. *Database Management Systems*. McGraw-Hill, 1998.

17. P. Revesz. *Introduction to Constraint Databases*. Springer-Verlag, 2002.

18. 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.

19. 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.

20. P. Revesz and S. Wu. Spatiotemporal reasoning about epidemiological data. *Artificial Intelligence in Medicine*, 38(2):157–70, 2006.

21. P. Rigaux, M. Scholl, and A. Voisard. *Introduction to Spatial Databases: Applications to GIS*. Morgan Kaufmann, 2000.

22. 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.

23. J. D. Ullman. *Principles of Database and Knowledge-Base Systems*. Computer Science Press, 1989.

24. V. Vapnik. *The Nature of Statistical Learning Theory*. Springer-Verlag, 1995.