Knowledge-Based Systems 15 (2002) 169–175

Knowledge-Based SYSTEMS

www.elsevier.com/locate/knosys

# Clustered linear regression

Bertan Ari[a], H. Altay Güvenir[b,*]

[a]*16021 NE 36th WAY, Redmond, WA 98052, USA*
[b]*Department of Computer Engineering, Bilkent University, Ankara 06533, Turkey*

Received 1 November 2000; accepted 2 February 2001

**Abstract**

Clustered linear regression (CLR) is a new machine learning algorithm that improves the accuracy of classical linear regression by partitioning training space into subspaces. CLR makes some assumptions about the domain and the data set. Firstly, target value is assumed to be a function of feature values. Second assumption is that there are some linear approximations for this function in each subspace. Finally, there are enough training instances to determine subspaces and their linear approximations successfully. Tests indicate that if these approximations hold, CLR outperforms all other well-known machine-learning algorithms. Partitioning may continue until linear approximation fits all the instances in the training set — that generally occurs when the number of instances in the subspace is less than or equal to the number of features plus one. In other case, each new subspace will have a better fitting linear approximation. However, this will cause *over fitting* and gives less accurate results for the test instances. The stopping situation can be determined as no significant decrease or an increase in relative error. CLR uses a small portion of the training instances to determine the number of subspaces. The necessity of high number of training instances makes this algorithm suitable for data mining applications. © 2002 Elsevier Science B.V. All rights reserved.

*Keywords*: Clustering Linear regression; Machine learning algorithm; Eager approach

## 1. Introduction

Approximating the values of continuous functions is called *regression* and it is one of the main research issues in machine learning, while approximating the values of functions that have categorical values is called as *classification*. In that respect, classification is a subcategory of regression. Some researchers emphasized this relation by describing regression as 'learning how to classify among continuous classes' [12].

For both these problems, we have also two types of solutions: *eager learning* and *lazy learning*. In eager learning, models are constructed according to the given training instances in training part. Such methods can obtain the interpretation of the underlying data. Constructing models in training leads long training times for eager learning methods. On the other hand, in lazy learning methods, all the work is done during testing, so they require much longer test times. Lazy learning methods do not construct models by using training data, so they cannot enable interpretation of training data. CLR, an extension of *linear regression*, is an eager learning approach.

Although most of the real life applications are classification problem, there are also very important regression problems such as problems involving time-series. Regression techniques can also be applicable to the classification problems. For example, neural networks are often applied to classification problems [14]

$$\mathrm{Est}(b) = \frac{\mathrm{Covariance}(x, y)}{\mathrm{Variance}(x)}$$

The traditional approach for regression problem is the classical *linear least-squares regression*. This old, yet effective, method has been widely used in real-world applications. However, this simple method has deficiency of linear methods in general. Advances in computational technology bring us the advantage of using new sophisticated non-linear regression algorithms. Among eager learning regression systems, CART [4], RETIS [9], M5 [12] and DART/HYESS [7] induces regression trees; FORS [3] uses inductive logic programming for regression and RULE [14] induces regression rules, projection pursuit regression [6], neural network models and MARS [5] produces mathematical models. Among lazy learning methods, locally weighted regression (LWR) [2] produces local parametric

* Corresponding author. Tel.: +90-312-290-1252; fax: +90-312-266-4126.

*E-mail addresses:* bari@microsoft.com (B. Ari), guvenir@cs.bilkent.edu.tr (H.A. Güvenir).

functions according to the query instances, and *k-NN* [1,10,11] algorithm is the most popular non-parametric instance-based approach for regression problems [13]. Regression by feature projections (RFP) method is an advanced *k-NN* method that uses feature projections based knowledge representation. This research uses the local weight and feature projection concepts and combines them with the traditional *k-NN* method. Using local weight with feature projection may cause losing the relation between the features; however, this new method eliminates the most common problems of regression, such as curse of dimensionality, dealing with missing feature values, robustness (dealing with noisy feature values), information loss because of disjoint partitioning of data, irrelevant features, computational complexity of test and training, missing local information at query positions and requirement for normalization.

CLR is an extension of *linear regression* algorithm. CLR approximates on the subspaces, and therefore, it can give accurate results for non-linear regression functions. Also, irrelevant features are eliminated easily. Robustness can be achieved by having large number of training instances. CLR can eliminate effects of noise as well.

## 2. Linear least-squares regression

Linear regression is the traditional approach for regression problems. There are two main classes of linear regression: *univariate linear regression* and *multivariate linear regression*.

### 2.1. Univariate linear regression

A set of data consisting of $n$ series of $x$ and $y$ values is given, where $x$ is the independent variable and $y$ is the dependent variable. In other words, there is only one unique feature that is represented by $x$, and the target is represented by $y$. Assume that there is a linear relation between variable $x$ and variable $y$:

$$y = bx + a$$

Here, $b$ is the slope of the line, while $a$ is the intercept at the $y$-axis. In reality, because of noise or mismatch between data and model, there is an error $\varepsilon$:

$$y_i = bx_i + a + \varepsilon_i$$

Finding a suitable model that minimizes the sum of squared errors for the given data set is the aim of *univariate linear regression*. Since *univariate linear regression* problem searches for a suitable model in the form of $y = bx + a$, a candidate slope, $b$ and intercept $a$ are chosen first. For each recorded $(x, y)$ pair, square of $(y - bx - a)$, which is equal to square of $e$, is added to the total error. The line having the smallest total error is the best-fit line and so is the best model for univariate linear regression. The value of $b$ can be estimated as follows:

$$b = \frac{\text{Covariance}(x, y)}{\text{Variance}(x)}$$

Note that if the variance of $x$ is zero, then we cannot estimate $b$. This occurs when the $x$ variable has the same value for all values of $y$. Once the value of $b$ is determined, the value of $a$ can be found easily.

### 2.2. Multivariate linear regression

Generally, the number of features in a data set is more than one. Finding a linear regression for data sets with more than one feature is called as *multivariate linear regression*. The model for multivariate linear regression can be represented in the index notation as follows:

$$y_i = a + b_1x_{1i} + b_2x_{2i} + b_3x_{3i}\ldots b_mx_{mi} + \varepsilon$$

By extending this representation, a system of $n$ equations and $m$ dependent variable can be shown as follows:

$$\begin{aligned}
y_1 &= a + b_1x_{11} + b_2x_{21} + b_3x_{31}\ldots b_mx_{m1} + \varepsilon_1 \\
y_2 &= a + b_1x_{12} + b_2x_{22} + b_3x_{32}\ldots b_mx_{m2} + \varepsilon_2 \\
&\vdots \\
y_i &= a + b_1x_{1i} + b_2x_{2i} + b_3x_{3i}\ldots b_mx_{mi} + \varepsilon_i \\
&\vdots \\
y_n &= a + b_1x_{1n} + b_2x_{2n} + b_3x_{3n}\ldots b_mx_{mn} + \varepsilon_n
\end{aligned}$$

We can estimate the values of unknown variables $b_m$, $\varepsilon_i$ and $a$ if we have sufficient training data. Since there are $n + (m + 1)$ unknowns and fewer equations than unknowns, there is no unique solution to this system of equations. The *least-squares* solution minimizes the sum of squares of the errors. Linear algebra was developed to facilitate the solution of systems of linear equations. Following the conventions of linear algebra, the multivariate linear regression model can be rewritten as

$$\begin{matrix} y_1 \\ y_2 \\ \vdots \\ y_i \\ \vdots \\ y_n \end{matrix} = \begin{bmatrix} 1x_{11} & x_{12} & x_{13} & \cdots & x_{1m} \\ 1x_{21} & x_{22} & x_{23} & \cdots & x_{2m} \\ \vdots & & & & \vdots \\ 1x_{i1} & x_{i2} & x_{i3} & \cdots & x_{im} \\ \vdots & & & & \vdots \\ 1x_{n1} & x_{n2} & x_{n3} & \cdots & x_{nm} \end{bmatrix} \times \begin{bmatrix} a \\ b_1 \\ b_2 \\ b_3 \\ \vdots \\ b_m \end{bmatrix} + \begin{bmatrix} \varepsilon_1 \\ \varepsilon_2 \\ \vdots \\ \varepsilon_i \\ \vdots \\ \varepsilon_n \end{bmatrix}$$

or more compactly:

$$\mathbf{y} = \mathbf{Xb} + \mathbf{e}$$

Then, we have a calculus problem to find a vector **b** that

Table 1
Common least-squares algorithms and their complexities

| LS algorithm | Complexity |
|---|---|
| Normal equations | $nm^2 + m^3/3$ |
| Householder orthogonalization | $2nm^2 - 2m^3/3$ |
| Modified gram Schmidt | $2nm^2$ |
| Givens orthogonalization | $3nm^2 - m^3$ |
| Householder bidiagonalization | $4nm^2 - 4m^3/2$ |
| R-bidiagonalization | $2nm^2 + 2m^3$ |
| Golub–Reinsch SVD | $4nm^2 + 8m^3$ |
| Chan SVD | $2nm^2 + 11m^3$ |

minimizes the inner product of the error vector:

$$\begin{aligned}\text{MinSSE} &= \mathbf{e}^\mathrm{T}\mathbf{e} = (\mathbf{y} - \mathbf{Xb})^\mathrm{T}(\mathbf{y} - \mathbf{Xb}) \\ &= \mathbf{y}^\mathrm{T}\mathbf{y} - \mathbf{y}^\mathrm{T}\mathbf{Xb} - \mathbf{b}^\mathrm{T}\mathbf{X}^\mathrm{T}\mathbf{y} + \mathbf{b}^\mathrm{T}\mathbf{X}^\mathrm{T}\mathbf{Xb}\end{aligned}$$

The vector of the partial derivatives of SSE with respect to the elements of **b** is

$$\frac{\partial \text{SSE}}{\partial \mathbf{b}} = -2\mathbf{X}^\mathrm{T}\mathbf{y} + 2\mathbf{X}^\mathrm{T}\mathbf{Xb}$$

The first order condition for a minimum requires that these first derivatives simultaneously be equal to zero. That is a necessary condition for the **b** that minimizes SSE is that

$$\mathbf{X}^\mathrm{T}\mathbf{Xb} = \mathbf{X}^\mathrm{T}\mathbf{y}$$

The solution to this equation is obtained by premultiplying both sides of this equality by a generalized inverse:

$$(\mathbf{X}^\mathrm{T}\mathbf{X})^{-1}\mathbf{X}^\mathrm{T}\mathbf{Xb} = (\mathbf{X}^\mathrm{T}\mathbf{X})^{-1}\mathbf{X}^\mathrm{T}\mathbf{y}, \qquad \mathbf{b} = (\mathbf{X}^\mathrm{T}\mathbf{X})^{-1}\mathbf{X}^\mathrm{T}\mathbf{y}$$

Evaluating $(\mathbf{X}^\mathrm{T}\mathbf{X})^{-1}$ is a very costly operation. It has a complexity of $O(n^2m^5)$ for a given $n \times m$ **X** matrix and evaluating $(\mathbf{X}^\mathrm{T}\mathbf{X})^{-1}\mathbf{X}^\mathrm{T}\mathbf{y}$ has a complexity of $O(m^3 + mn^2 + n^2m^5)$. There are better algorithms for solving the *least-square* problems. These algorithms and their complexity are presented in Table 1.

We need to choose an algorithm that minimizes the coefficients of the term $n$ and its powers because generally for a given data set, the number of instances is much larger than the number of features ($n \gg m$). This situation favors the *normal equations* method, and it is the method CLR uses to solve the *least-squares* problem.

## 3. Clustered linear regression

Clustered linear regression performs linear approximations on subspaces of training space. We assume that linear approximations in subspaces fit the non-linear regression function better. Using this approach, mathematical models of non-linear data sets can be found. This approach also eliminates the problems that are caused by correlated variables that frequently occur in real-world databases. An example of a data set with one feature is given in Fig. 1.

As shown in Fig. 1, the target is a function of only one feature, namely feature$_1$. This function is a non-linear regression function and does not fit any linear approximation in the whole space. In this example, we have local linear approximations that allow CLR to partition the space and find better linear approximations. If subspaces can be detected successfully, linear approximations of each subspace can easily be found by using the linear least-squares regression. However, sufficient number of training instances is needed to determine the number of subspaces, boundaries of subspaces and each subspace should contain enough training instances to apply the linear least-squares regression. We have three main assumptions to apply CLR successfully. These assumptions are as follows:

1. Target is a function of features.
2. This non-linear regression function of features has some local linear approximations.
3. There are enough training instances to determine the number of subspaces, boundaries of the subspaces and linear approximations of each subspace.

These assumptions are essential for CLR. If there is no functional relation between target and features, there is no possibility of a linear approximation that fits some region of non-linear regression function. The second assumption indicates the necessity of local linear approximations. If we

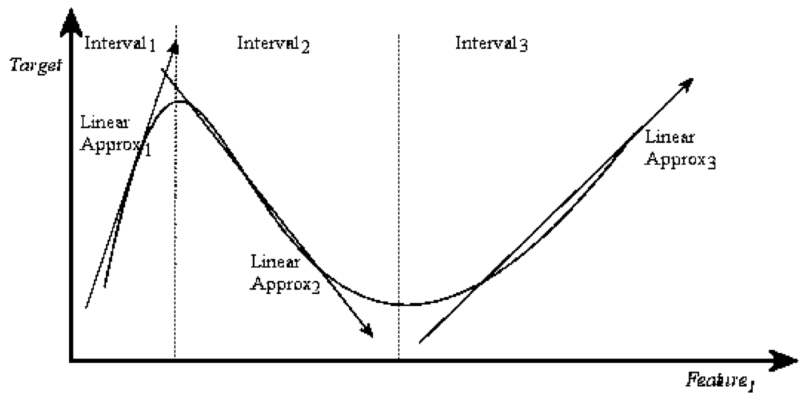

Fig. 1. An application of CLR for a training space with one feature.

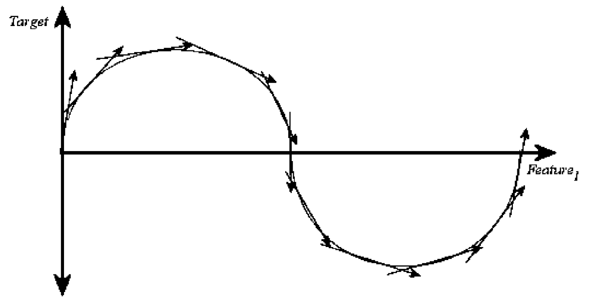

Fig. 2. Some non-linear functions may not have good local linear approximations.

have a non-linear regression function such as in Fig. 2, CLR cannot perform accurately. This can be a sinusoidal function that has no local linear approximation.

CLR algorithm has three main parts:

- finding the number of subspaces,
- determining boundaries of subspaces,
- applying linear regression to all subspaces.

### 3.1. The CLR algorithm

Subspaces are defined by simple rules on each feature. A rule for a feature determines an interval on that feature. Rules determine intervals on a feature such that they are mutually exclusive. Subspaces are composed of all possible combinations of the intervals and hence the rules.

Determining partitions suffices to determine all subspaces. So, three main operations of CLR algorithm can be stated as:

- finding number of intervals on each feature,
- determining the boundaries of intervals,
- applying linear regression to all subspaces.

#### 3.1.1. Finding number of intervals

There is no certain heuristic for determining the number of intervals. Increasing the number of intervals increases the number of subspaces, and therefore, better linear approximations can be found. However, increasing the number of subspaces causes *over fitting* (losing generality) and, as a result, decreases the accuracy on unseen cases.

Since there is no certain heuristic for determining number of intervals, CLR simply increases the number of intervals on each feature until the relative error does not decrease. CLR maintains a vector **v**, whose each element corresponds a feature. In each step, CLR increases only one element of **v**. The increase that decreases the relative error mostly in a member of **v** is chosen in that step. When the algorithm stops, **v** holds the number of intervals for each feature.

It is obvious that trying to increase the number of intervals of a feature is useless after it causes an increase in the relative error because it shows that *over fitting* on that feature already started. Once over fitting starts, certainly increasing the number of subspaces so increasing the number of intervals on a feature, increases the effects of over fitting. Some number of useless tries can be avoided by 'banning' a feature on which increasing number intervals already caused over fitting.

For determining intervals, some test instances are randomly selected from the train data set. In our experiments, the size of the test data for determining the intervals is the 10% of the train data set. Tests on different data sets indicate that this ratio is the optimum for sufficiently large data sets, and changing this ratio does not affect the accuracy much. Note that the test data set used in the evaluation of a particular interval is different from the test data set used in the evaluation of the CLR algorithm on a data set.

#### 3.1.2. Determining the intervals

Determining the number of intervals and their boundaries is a critical part of the CLR algorithm. Interval boundaries should be carefully selected so that the total error can be minimized. We developed a new algorithm for this purpose, called LRClustering.

The LRClustering algorithm tries to find the suitable interval boundaries so the total *mean absolute distance* (MAD) is minimized. After forming subspaces by using all combinations of intervals on each feature, linear regressions of these subspaces can be found and evaluated by the *linear least-squares regression* method.

Interval boundaries are determined by the training

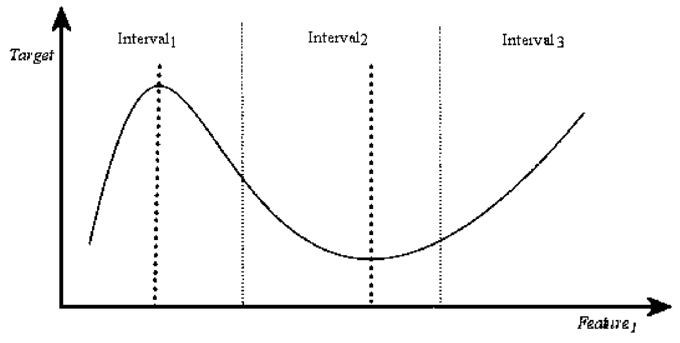

Fig. 3. Determining intervals on one feature.

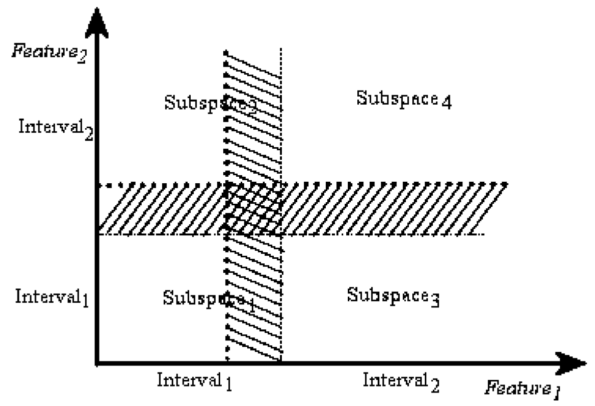

Fig. 4. An example of partitioning with two features.

instances. *Extreme* instances of an interval determine the boundaries of this partition. Extreme instances of an interval on a feature are the instances whose feature values are either maximum or minimum. So, a dense distribution of training instances is necessary to determine the boundaries sharply.

For the sake of simplicity, let us consider an example with only one feature. Let us assume that this feature has a functional relation between target such as in Fig. 3 and by having three intervals on this feature, this functional relation can be optimally approximated. All training instances lie on a regression curve and they are distributed equally. Initially, two boundaries on this feature are determined by distributing instances to three intervals equally. Linear regression and MADs of these three intervals are evaluated by the *least-square linear regression* method. In Fig. 3, bold dotted lines indicate the boundaries that minimize MAD of the system and so the boundaries of the final intervals and thin straight lines represent the initial boundaries.

Interval$_1$ and Interval$_2$ are hybrid intervals as they contain some parts of the different final intervals, which have totally different linear approximations. Linear regression of these intervals cannot be as accurate as Interval$_3$, which contains only one final interval. We call the intervals that contain only parts of one final partition such as Interval$_3$ as *deprive intervals*. There is at least one *deprive interval* for each

Table 2
Data sets and their relevant properties

| Dataset | Original name | Instances | Features |
|---|---|---|---|
| AL | Ailerons | 7154 | 40 |
| CA | Computer activity | 8192 | 21 |
| EV | Elevators | 8752 | 18 |
| FD | Fried | 40,768 | 10 |
| HO | House_16h | 22,784 | 16 |
| KI | Kinematics | 8192 | 8 |
| PL | Plastic | 1650 | 2 |
| PN | 2D planes | 40,768 | 10 |
| PT | Pole telecom | 9065 | 48 |
| | Artificial | 10,000 | 10 |

feature if the number of intervals is more than the actual intervals.

For more complex systems that have more than one feature, we need to focus on subspaces. As stated above, subspaces are formed by the combination of intervals from each feature. The subspaces that are formed only by *deprive intervals* are called *deprive subspaces*. Deprive subspaces can be easily detected by their high accuracy (low error rate) according to other subspaces. Deprive subspaces play an important role in determining boundaries of partitions.

Initially, boundaries are selected randomly. Their final values are determined by the help of MAD of all subspaces. Let us assume a boundary of an interval as given in Fig. 4. A boundary must be moved to its correct location. We first determine which direction and how much we move. Deprive intervals are used to determine direction. If a boundary is not in its correct location, both neighboring partitions will have high MAD values. But, if one of these intervals is deprive, it will have a low MAD. It is obvious that the boundary between a deprive interval and its neighboring interval with high MAD should move to the direction of the interval with high MAD because this interval contains some part of the final interval of deprive interval. We apply the same approach to subspaces. While the LRClustering algorithm is processing a boundary, it sums the MAD of the subspaces ‘upper’ and ‘lower’ from the boundary. If MAD of the upper subspace is higher than that of lower,

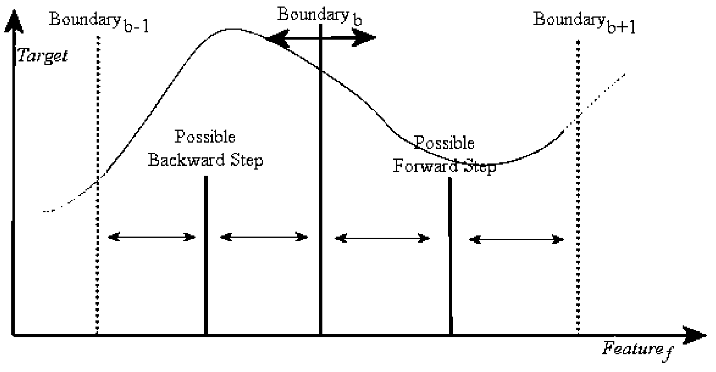

Fig. 5. A step of searching optimal position of boundary *b*. Training instances are placed on the curve, so they are sorted on their values of feature *f*. Possible next steps are half of the distances between boundary *b* and its neighbors.

Table 3
Comparison of the results of the algorithms

| Dataset | RE | | | |
|---|---|---|---|---|
| | CLR | MARS | RSBF | KNNR |
| AL | 0.432 | **0.412** | 0.439 | 0.443 |
| CA | 0.191 | 0.213 | 0.591 | **0.189** |
| EV | 0.45 | **0.443** | 0.461 | 0.606 |
| FD | **0.205** | 0.243 | 1.77 | 0.352 |
| HO | 0.71 | 0.758 | 0.816 | **0.678** |
| Kl | 0.438 | 0.707 | 0.789 | **0.403** |
| PL | 0.42 | **0.41** | 0.442 | 0.480 |
| PN | 0.254 | **0.22** | 0.567 | 0.289 |
| PT | 0.343 | 0.449 | 0.956 | **0.131** |
| Average | **0.383** | 0.428 | 0.759 | 0.397 |

the boundary moves to upper direction or vice versa. In Fig. 4, we have a domain with two features. Bold dotted lines represent optimum boundaries and thin straight lines show the current location of the boundaries.

In this example, $\text{Interval}_2$ of $\text{feature}_1$ and $\text{Interval}_1$ of $\text{feature}_2$ are deprive intervals and so $\text{Subspace}_3$ is a deprive subspace. Shaded regions are the regions, which do not belong to current subspace and cause increase in MAD of these subspaces. Note that $\text{Subspace}_3$, a deprive subspace, has no shaded region. This is why deprive subspaces have lower MADs. Let us assume that $\text{MAD}_i$ represents MAD of $\text{Subspace}_i$. By assuming that shaded regions cause increases in MAD, following inequalities can easily be retrieved from Fig. 4

$$\text{MAD}_1 + \text{MAD}_3 < \text{MAD}_2 + \text{MAD}_4 \tag{1}$$

$$\text{MAD}_3 + \text{MAD}_4 < \text{MAD}_1 + \text{MAD}_2 \tag{2}$$

Note that the deprive subspace, $\text{Subspace}_3$, is always in the small side of the inequality. Eq. (1) determines the direction of the boundary on $\text{feature}_1$ and Eq. (2) determines the direction of the boundary on $\text{feature}_2$. By using this approach, we have an idea about in which direction a boundary advances in the next step. Length of this step is equal to the number of instances we advance. Whatever the difference between the feature values of two instances is unimportant.

Length of the next step depends on the search algorithm used. If it is a sequential search, we advance one instance to the preferred direction and update subspaces and MAD of the system to test if there is an improvement. Testing a new situation is very costly. Even it is done linearly; a better approach is necessary. The most efficient search algorithm is the binary search. The LRClustering algorithm uses binary search to find the optimum positions of boundaries. Binary search requires a sorted list, so all instances are sorted on their feature values for each feature. This preprocessing is done initially only once. Preprocessing spends negligible time, but using binary search decreases the number of testing new situation dramatically and so the training time. Fig. 5 shows the search approach.

We have already explained two main parts of CLR. The last part of the algorithm is finding linear approximation in a given subspace. The LRClustering algorithm uses the *linear least-squared regression* for this purpose.

## 4. Evaluation of the CLR algorithm

CLR is an algorithm based on linear regression, so most of the advantages and disadvantages of linear regression are carried to CLR. For example, curse of dimensionality is one of the biggest problems of linear regression since it increases the complexity. If the number of features increases, accuracy will decrease because possibility and effects of interaction between features will increase.

One of the problems of linear regression is finding a general linear approximation for the whole training space. Generally, different parts of training subspace have different local linear approximations. This idea led us to develop the CLR algorithm. CLR can successfully detect such local linear approximations and handle cases of correlated

Table 4
The training and test times of the algorithm. All results are given in ms scale

| Dataset | CLR | | MARS | | RSBF | | KNNR | |
|---|---|---|---|---|---|---|---|---|
| | Train | Test | Train | Test | Train | Test | Train | Test |
| AL | $1.70\times10^6$ | 51.3 | $2.67\times10^4$ | 7.4 | $3.74\times10^4$ | 39.1 | 115.2 | $4.91\times10^4$ |
| CA | $8.60\times10^5$ | 53.4 | $1.99\times10^4$ | 8 | $1.73\times10^4$ | 47.2 | 71.8 | $4.29\times10^4$ |
| EV | $5.03\times10^5$ | 55.9 | $1.80\times10^4$ | 8.2 | $1.10\times10^4$ | 40.7 | 65.9 | $4.35\times10^4$ |
| FD | $1.71\times10^6$ | 361 | $6.31\times10^4$ | 42.7 | $1.77\times10^4$ | 236.9 | 179.2 | $6.69\times10^5$ |
| HO | $2.36\times10^6$ | 191.1 | $4.86\times10^4$ | 22.2 | $2.53\times10^4$ | 148.2 | 154.5 | $2.75\times10^5$ |
| Kl | $1.90\times10^5$ | 49.9 | $9.36\times10^3$ | 8.9 | $2.18\times10^3$ | 35.6 | 26.1 | $2.51\times10^4$ |
| PL | $2.28\times10^2$ | 5.2 | $3.14\times10^2$ | 0.2 | $7.86\times10^1$ | 8.4 | 0.3 | $5.61\times10^2$ |
| PN | $1.98\times10^6$ | 363.4 | $6.18\times10^4$ | 38.9 | $1.49\times10^4$ | 183.9 | 177.3 | $6.64\times10^5$ |
| PT | $3.56\times10^6$ | 33 | $9.62\times10^3$ | 6.3 | $3.31\times10^4$ | 26.1 | 101.9 | $2.48\times10^4$ |
| Average | $1.43\times10^6$ | 129.36 | $2.86\times10^4$ | 15.9 | $1.77\times10^4$ | 85.12 | 99.13 | $1.99\times10^5$ |

Table 5
Test results on artificial data set

| | CLR | MARS | RSBF |
|---|---|---|---|
| Artificial | 0.029 | 0.868 | 0.972 |

features and relations of higher order. This is the main advantage of CLR on linear regression.

Linear regression is a fast algorithm due to its simplicity. CLR is a complex algorithm because determining number of intervals and optimal positions of their boundaries is required. However, CLR is an eager learning approach, and so its test time is very short. Since training is done only once, its slowness is not a problem as the other lazy learning approaches.

### *4.1. Data sets and results*

Results of CLR are measured by evaluating relative error for each data set and using tenfold cross-validation. CLR works on data sets that have only linear features. Initially, we present the data sets and their properties in Table 2.

These data sets are the largest data sets available in Ref. [8]. Since CLR requires large number of training instances, we tested CLR on these data sets. An artificial data set is also generated by a random instance generator such that three main assumptions of CLR hold. By using this data set, we can check if CLR can find the mathematical model of a data set, which holds the given assumptions.

The relative error (RE) results are presented in Tables 3 and 4. Since these data sets are very large, their sizes are above the limits of the most of the implementations of the machine-learning algorithms. We could test these data sets only on MARS, RSBF, KNNR and CLR.

The tests show that although CLR has a high-time complexity and so long training time, its accuracy is higher than MARS and RSBF. Since the training is performed only once, while testing is done for many times, the low test time requirements make CLR suitable for applications where high accuracy is crucial and sufficiently large number of instances is available.

Tests on artificial data set show that only CLR has the capability of determining the mathematical model of such a system. This feature of CLR can be useful for mathematicians or scientist to get a mathematical model for a given system. The results of the tests on artificial data set are presented in Table 5.

## 5. Discussion

We presented a new algorithm that improves the accuracy of linear regression by clustering training spaces of data sets to improve the accuracy of local linear regressions.

It is assumed that there are linear approximations that fit the training data locally. By applying this approach, we obtain accuracy rates higher than simple multivariate linear regression. New algorithm keeps the advantages and disadvantages of linear regression. Curse of dimensionality is still a big problem in CLR. Large number of features increases the requirements for number of training instance and training time.

CLR can make good linear approximations only on large data sets. Determining boundaries exactly requires a dense data set. Finding linear regression of each subspace accurately also requires high training numbers because number of subspaces is exponentially related to the number of features. So, in small data sets, generally CLR gives inaccurate results. This constraint makes the CLR algorithm suitable for data mining applications.

## References

[1] D. Aha, D. Kibler, M. Albert, Instance-based learning algorithms, Machine Learning 6 (1991) 37–66.
[2] G.C. Atkenson, A.W. Moore, S. Schaal, Locally Weighted Learning, http://www.cs.gatech.edu/fac/Chris.Atkenson/local-learning/, October 12, 1996.
[3] I. Bratko, A. Karalic, First order regression, Machine Learning 26 (1997) 147–176.
[4] L. Breiman, J.H. Friedman, R.A. Olshen, C.J. Stone, Classification and Regression Trees, Wadsworth, Belmont, CA, 1984.
[5] J.H. Friedman, Multivariate adaptive regression splines, The Annals of Statistics 19 (1) (1991) 1–141.
[6] J.H. Friedman, W. Stuetzle, Projection pursuit regression, Journal of American Statistical Association 76 (1981) 817–823.
[7] J.H. Friedman, Local Learning Based on Recursive Covering, ftp://stat.stanford.edu/pub/friedman/dart.ps.Z, 1996.
[8] H. Altay Guvenir, I. Uysal, Bilkent University Function Approximation Repository, http://funapp.cs.bilkent.edu.tr, 2000.
[9] A. Karalic, Employing linear regression in regression tree leaves, Proceedings of ECAI'92, Austria 1992 pp. 440–441.
[10] D. Kibler, D.W. Aha, M.K. Albert, Instance-based prediction of real-valued attributes, Computational Intelligence 5 (1989) 51–57.
[11] T.M. Mitchell, Machine Learning, McGraw Hill, New York, 1997.
[12] J.R. Quinlan, Learning with continuos classes, in: Adams, Sterling (Eds.), Proceedings AI'92, Singapore, 1992, pp. 343–348.
[13] I. Uysal, H.A. Guvenir, Regression on feature projections, Knowledge-based Systems 13 (4) (2000) 207–214.
[14] S.M. Weiss, N. Indurkhya, Rule-based machine learning methods for functional prediction, Journal of Artificial Intelligence Research 3 (1995) 383–403.