# An improved algorithm for feature selection using fractal dimension

Haiqin Zhang[1] Chang-Shing Perng[2] Qingsheng Cai[1]
[1]Department of Computer Science, University of Science and Technology of China
[2]IBM Thomas J. Watson Research Center
E-mail: face@mail.ustc.edu.cn

**Abstract**

Dimensionality reduction is an important issue in data mining and machine learning. Traina[1] proposed a feature selection algorithm to select the most important attributes for a given set of n-dimensional vectors based on correlation fractal dimension. The author used a kind of multi-dimensional "quad-tree" structure to compute the fractal dimension. Inspired by his work, we propose a new and simpler algorithm to compute the fractal dimension, and design a novel and faster feature selection algorithm using correlation fractal dimension, whose time complexity is lower than that of Traina's. The main idea is when computing the fractal dimension of ($d$-1)-dimensional data, the intermediate generated results of the extended $d$-dimensional data is reused. It inherits the desirable properties described as in [1]. Also, Our algorithm does not require the grid sizes decrease by half as the original "quad-tree" algorithm. Experiments show our feature selection algorithm has a good efficiency over the test dataset.

**Key words:** Machine learning, Feature selection, Fractal dimension, Embedding dimension

## 1. Introduction

Many factors affect the success of data mining algorithms on a given task. The quality of the data is one such factor--if information in the data is irrelevant or redundant, or it's noisy and unreliable, then the process of knowledge discovery during training gets more difficult.

Feature subset selection is the process of identifying and removing as much irrelevant and redundant information as possible. It reduces the dimensionality of the data and may allow learning algorithms to operate faster and more effectively. In some cases, accuracy on future classification can be improved; in others, the result is a more compact, easily interpreted representation of the target concept.

Recent research has shown common machine learning algorithms to be adversely affected by irrelevant and redundant training information. The simple nearest neighbor algorithm is sensitive to irrelevant attributes—its sample complexity (number of training examples needed to reach a given accuracy level) grows exponentially with the number of irrelevant attributes [2,3]. Sample complexity for decision tree algorithms can grow exponentially on some concepts (such as parity) as well. The naïve Bayes classifier can be adversely affected by redundant attributes due to its assumption that attributes are independent given the class [4]. Decision tree algorithms such as C4.5 [5] can sometimes overfit training data, resulting in large trees. In many cases, removing irrelevant and redundant information can result in C4.5 producing smaller trees [6]. Therefore, the selection of a small number of highly predictive features is necessary in order to avoid overfitting the training data. Reducing the dimensionality of the data reduces the size of the hypothesis space

[1] This research was partially funded by National Natural Science Foundation of China under Grants 60075015 and 90104030.

and allows algorithms to operate faster and effectively.

Numerous selection methods have been studied, including genetic algorithms; sequential feature selection algorithms such as forwards, backwards and bidirectional sequential searches; and etc. Principal component analysis (PCA) is a useful tool for data reduction, and is well studied in statistics. The idea is to find linear combinations of the attributes, while either maximizing or minimizing the variability. But PCA is sensitive to noise and a few outliers can radically affect the results. So does singular value decomposition. Recent surveys on attribute selection using machine learning techniques are presented in [7,8].

In [1], Traina introduced a novel approach that can discover how many features are significant to characterize a dataset, and presented a fast, scalable algorithm to quickly select the most significant features of a dataset. The main idea is to use the correlation fractal dimension of the dataset to drop attributes which do not affect it. The correlation fractal dimension (CFD) is relatively unaffected by redundant attributes, and an algorithm was proposed that can compute CFD in linear time with respect to the count of objects. Based on that, a kind of backward-elimination algorithm was proposed to sequentially remove attributes that contribute minimally to CFD.

But the above fractal dimension reduction (FDR) approach is quite slow when we implement it under *Matlab* environment. The reason is that the FDR algorithm cannot take full advantage of the results of individual fractal dimension computation. Therefore, We propose a new approach for computing the CFD, which is not limited to the memory available. Based on fractal dimension computation, we propose a new feature selection algorithm that identifies how many attributes are sufficient for representing the dataset. The main idea is when computing CFD of ($d$-1)-dimensional data, the computing results of the extended $d$-dimensional data is reused. For example, if we remove any of the features from a $d$-dimensional data, we can get totally $d$ ($d$-1)-dimensional data. Since we already get the CFD results of $d$-dimensional data, we can reuse these results when computing the CFD of the $d$ ($d$-1)-dimensional data.

The remainder of the paper is organized as follows. Section 2 introduces the concepts needed to understand our proposed method. Section 3 gives the fractal dimension algorithm developed by us, and follows by a simple comparison with Traina's algorithm. In Section 4, we present the new proposed feature selection algorithm, and show it has a lower time complexity over Traina's. Section 5 discusses the experiments and evaluation of our techniques. Section 6 concludes the paper with a simple summary.

## 2. Concepts

In multi-dimensional space, a dataset is represented with columns as attributes (features) and rows as different data objects. We are interested in those datasets with numerical attributes since they are very common in real datasets. For ease of understanding, we will give some simple definitions (more detail in [1]).

Definition1- The embedding dimension $E$ of a dataset is the dimension of its address space. In other words, it is the number of attributes of the dataset.

Definition2 – The intrinsic dimension $D$ of a dataset is the dimension of the spatial object represented by the dataset, regardless of the space where it is embedded.

Fractal dimension has been a useful tool for the analysis of spatial access methods [9]. A fractal dataset is known by its characteristic of being self-similar. Fractal datasets are characterized

by their fractal dimensions. By embedding the data set in an *E*-dimensional grid whose cells have sides of size *r*, we can compute the frequency of data points falling into the *i*-th cell, thus compute $D_q$, the generalized fractal dimension.

Definition3 – The fractal dimension $D_q$ of a dataset is defined as:

$$D_q \equiv \frac{1}{q-1}\frac{\partial \log \sum_i C_{r,i}^q}{\partial \log r}$$

Here, *r* is the grid size, $C_{r,i}$ is the number of objects in the *i*-th cell under grid size *r*. There are three kind of fractal dimension widely used: Hausdorff fractal dimension ($q$=0); Information fractal dimension($\lim_{q->1} D_q$); Correlation fractal dimension($q$=2). $D_2$ measures the probability that two points chosen at random will be within a certain distance of each other. Changes in the correlation dimension mean changes in the distribution of points in the data set. Here we use $D_2$ as the intrinsic dimension of a dataset, for the approach we proposed is to identify the correlated attributes and discard those uncorrelated. The sum of occupancy is defined as $S(r) = \sum_i C_{r,i}^2$. The fractal dimension of a dataset is the derivative of *log*(*S*(*r*)) with respect to the logarithm of the radius. As we assume self-similar datasets, we expect the derivative results in a constant value. Thus, we can obtain the fractal dimension $D_2$ of a dataset plotting *S*(*r*) for different values of the radius *r*, and calculating the slope of the resulting line. For convenience, we will use CFD or *D* to represent $D_2$ in the following sections.

Notice here all algorithms proposed in this paper are implemented under *Matlab* 6.0, which is optimized for matrix operations, and where a dataset is regarded as a matrix. We also use some optimized functions to implement our algorithms.

## 3. Computation of fractal dimension

This section presents two algorithms to compute the fractal dimension *D* of any given set of points in any *E*-dimensional space. The first one uses box-counting approach proposed by Traina[1]. Based on it, we propose a simpler dividing and counting algorithm.

*Algorithm 1 Compute fractal dimension D of a dataset A (box-count approach)*
*input: normalized dataset A (N rows, with E dimensions/attributes each)*
*output: fractal dimension D*
*Begin*
*For each desirable grid-size r*
  *$C_i$ is the element count in the i-th grid.*
  *Compute the sum of occupancies S(r) = sum($C_i$^2)*
*end;*
*Print the values of log(r) and log(S(r)) generating a plot;*
*Return the slope of the linear part of the plot as the fractal dimension D of the dataset A.*
*End*

Traina[1] use a kind of multi-level grid structure to store the object count in different grids under different level(grid size). The structure is easily composed when considering each level has

a radius the half of the size of the previous level, that is, the grid sizes are sequenced as ($r$=1, 1/2, 1/4, 1/8, etc.). Each level of the structure corresponds to a different radius, so the depth of the structure is equal to the number of points in the resulting plot. The structure is created in main memory, so the depth of the structure is limited by the amount of the main memory available.

We propose a simpler algorithm for computing fractal dimension. It is not based on any hierarchy structure, so its efficiency will not be affected by the amount of main memory available. Also, it does not require the grid sizes decrease by half. Here we first give an algorithm to compute the sum of occupancies $S(r)$ for a given grid size $r$.

*Algorithm 2: Divide_count(A,r,count) (divide and count approach on computing S(r))*
*input: normalized dataset A (N rows, with E dimensions/attributes each) and grid-size r;*
*count is a vector that stores the object count in N grids, initially each with a count of 1.*
*output: sum of occupancy S(r):*
*begin*
*data=floor(A./r); i=1; s=N;*
*data=sortrows(data); //sorts the rows of the matrix data in ascending order as a group*
*while (i<=s-1)*
*if data(i)==data(i+1), //the two objects are the same*
*Increase the count in the i-th grid with that of (i+1)-th grid;*
*Delete the (i+1)-th grid: delete the (i+1)-th object from data and count;*
*s=s-1;*
*Else i=i+1;*
*End;*
$C_i$*=count(i); S(r)=sum(*$C_i$*^2);*
*End*

The smartness of this algorithm is that we used the property of equal grid sizes for all attributes, divided them by the radius, and sorted the results according to each attribute; If any two data object should fall in a grid, the dividing results by $r$ must be the same, and after "*sortrows*", they must be the neighboring elements, so it makes very easy to accumulate the data count in $i$-th grid. Finally, $s$ is the unique available grid count under grid size $r$.

For each given $r$, we can get the sum of occupancy by Algorithm 2 with different parameters. Then the fractal dimension is the derivative of $log(S(r))$ with respect to $log(r)$. If the grid sizes are inversely sequenced as (1/2,1/4,1/8,etc), we can get a more tighter algorithm to compute CFD.

*Algorithm 3 Compute fractal dimension D of a dataset A (divide-count approach)*
*begin*
*For each grid size r from min to max*
*If r is the minimal radius, [count,data]=divide_count(data,r,count)*
*Else [count,data]=divide_count(data,2,count);*
$C_i$*=count(i); S(r)=sum(*$C_i$*^2);*
*end*
*Print the values of log(r) and log(S(r)) generating a plot;*
*Return the slope of the linear part of the plot as the fractal dimension D of the dataset A.*

*End*

When computing $S(r)$ for next radius, we reuse the intermediate counting results of previous step. Each step the size of "*data*" is decreased by being divided and counted. Therefore, for larger grid size, the efficiency is improved a lot compared with directly calling "*divide_count*" each time with the original data and a different $r$. We do not need to generate any hierarchy structure to store the counting results.

We know from [1] that Algorithm 1 is linear on the number of points in the dataset. The computational complexity is $O(N*E*R)$, where $N$ is the number of objects in the dataset, $E$ is the embedding dimension, and $R$ is the number of points used to plot the $S(r)$ function.

For Algorithm 3, since the grid sizes are inversely sequenced as (1/2,1/4,1/8,etc), we can assume that by continuous dividing operations, each step the object count in the remaining data set is decreased by 1/2. So the time complexity of Algorithm 3 is $\sum_{r=1}^{R} E * N / 2^{r-1} \approx O(E * N)$ roughly, which is independent of $R$, the number of different grid sizes. This shows that the algorithm is only scalable to the size of dataset and its embedding dimension.

## 4. Fractal dimension reduction

The goal of feature selection is to select a subset of relevant features from the original feature set, that is, we need to discard a subset of unrelevant features. The number of features to be discarded depends on the fractal dimension of the dataset.

Observation 1 – The fractal dimension of a dataset cannot be greater than its embedding dimension. There are $\lceil D \rceil$ attributes which cannot be determined from the others. Since $D \leqslant E$, there are at least $E-\lceil D \rceil$ attributes which can be correlated with the others [1].

According to this observation, we can sequentially remove those attributes that have minimal effect on the fractal dimension of the dataset. Then, the attributes left are those unrelevant to each other. The attributes removed are relevant or dependant on those attributes left. Based on this idea, Traina proposed Algorithm 4 for fractal dimension reduction[1].

*Algorithm 4 - Fractal dimensionality reduction (FDR1) algorithm*
*input: dataset A, E is the embedding dimension*
*output: list of attributes in the reverse order of their importance*
*Begin*
*1- Compute the fractal dimension D of the whole dataset;*
*2- Initially set all attributes of the dataset as the significant ones,*
*and the whole fractal dimension as the current D;*
*3- While there are significant attributes do:*
*4- For every significant attribute i, compute the fractal dimensions $pD_i$*
*using all significant attributes excluding attribute i;*
*5- Sort the fractal dimensions $pD_i$ obtained in step 4 and select the*
*attribute a which leads to the minimum difference (current D - $pD_i$ );*

*6- Set the pDi obtained removing attribute a as the current D;*
*7- Output attribute a and remove it from the set of important attributes;*
*end*

Based on algorithm 2 in Section.4 for computing fractal dimension, we propose a new feature selection algorithm. By observation, we find that given the grid distribution of *d*-dimensional dataset, after removing any of the features from *d*-dimensional dataset, we can get the gird distribution of those (*d*-1)-dimensional dataset by combining the grid distribution of the original *d*-dimensional dataset. The main idea is when computing CFD of (*d*-1)-dimensional data, the intermediate computed results of the extended *d*-dimensional data is reused. For example, if we remove any of the features from a *d*-dimensional data, we can get totally *d* (*d*-1)-dimensional data. Since we already get the dividing and counting results of *d*-dimensional data, we can reuse these results when computing the CFD of the *d* (*d*-1)-dimensional data.

*Algorithm 5 - Fractal dimensionality reduction (FDR2) algorithm (Count reusing approach):*
*Input&Output: data both as the original dataset and reduced dataset.*
*begin*
*While there are significant attributes do*
*Data1=data; //with N1 object*
*count is a unit vector that stores the object count in N1 grids, initially each with a count of 1;*
*For each grid size r from min to max*
*If r is the minimal grid size, [count ,data1]=divide_count(data1,count,r)*
*else [count ,data1]=divide_count(data1,count,2);*
*S(r)=sum(count^2);*
*For each attribute j in data1*
*Data2 is the remaining set after removing j from data1;*
*count2=divide_count(data2,count,1); //directly counting the repetitive elements*
*S2(r,j)=sum(count2.^2); //S2 is the set of partial sum of occupancies*
*end;*
*End;*
*If the first unsignificant attribute, // compute the fractal dimension of the original dataset*
*FD1= the slope of the linear part of log(S(r)) versus log( r)*
*For each missing feature i, // compute the partial fractal dimension of the remaining dataset*
*FD2(i)=the scope the linear part of log(S2(r,i)) versus log( r);*
*end*
*Get min_i, so that FD2(min_i) has the minimal difference with FD1;*
*Data = the remaining set after deleting feature min_i from data;*
*End;*
*End;*

The reduced feature set lies in the final reduced dataset. The stopping condition, can be deleting at most *E*-upper(FD1) features or minimal difference satisfying some threshold.

The time complexity of Algorithm 4 is $O(N*E^2*M*R)$, where *M* is the number of features removed. he time complexity of Algorithm 5 is $O(N*E^2*M)$ which is invariant with different *R*.

Both algorithms are linear on *N* (number of objects) and quadratic on the embedding dimension of the dataset.

## 5. Experiments

The proposed two algorithms are implemented under *Matlab*6.0 environment. Traina implemented his hierarchy grid structure under Object C++ environment. Since we can't do that in *Matlab*, we can only do experiments with our proposed techniques.

We evaluate our methods on two datasets. Surprisingly, we find that some randomly generated datasets have self-similar property, a 2-dimensional randomly generated dataset with 8000 points has fractal dimension 1.96. We use such a dataset as a BASE for producing other datasets. The datasets to be evaluated are built over the BASE and are added some more attributes. Respectively, we use 5000,8000,12000, 14000 of points for each dataset to test our algorithms. The two datasets are:

DATASET1 - The original 2D points of the BASE dataset (x, y) became 5D points ($a=x$, $b=y$, $c=a+b$, $d=a^2+b^2$, $e=a^2-b^2$). The three latest coordinates included in this dataset are strongly correlated with the two first coordinates. Thus, the fractal dimensions of the new datasets (1.86 for size 8000; 1.87 for 10000) are close to the fractal dimension of the original BASE dataset (1.96).

DATASET2 – Three more attributes are added to DATASET1 ($f$=F($a$,$b$), $g$=*random1*, $h$= *random2*). As the three new coordinates include random noise to the dataset, the fractal dimension of DATASET2 becomes 3.85(for size 10000), basically the dimensionality of the DATASET1 (1.87) plus the dimensionality of a square in 2D (2.0). The sixth variable ('$f$') is non-linearly depending on the others, and has not much influence on the fractal dimension of DATASET2.

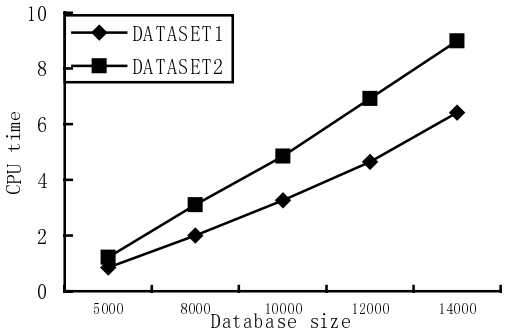


Fig.1 CPU time of Algorithm 3

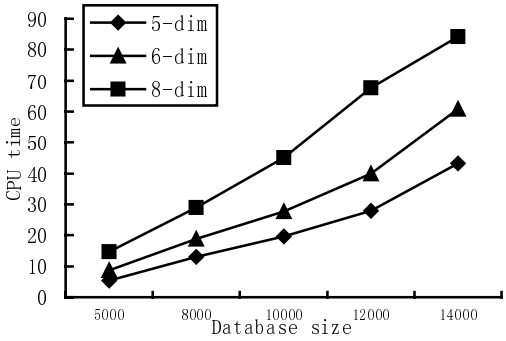


Fig.2 CPU time of Algorithm 5

Figure 1 presents the plot of CPU time of Algorithm 3 for computing CFD of DATASET1 and DATASET2. Figure 2 presents the CPU time of Algorithm 5 for fractal feature selection of three datasets which are produced by selecting the first 5,6,8 attributes from DATASET2. From these plots, we can see that the execution times of the two algorithms are linear on the number of points in the dataset, and Algorithm 5 is nearly square with the dimension of the dataset.

Table.1 presents a snapshot during the execution of Algorithm 5 for DATASET1 (size 8000). Each step, the algorithm selects an attribute that has the minimal significance. The first data row contains the CFD of the whole dataset and the rough elapsed time for computing it (The time is rough because Algorithm 5 computes the CFD sequentially and crossly). Other data rows

sequentially present the attributes dropped, the CFD of the remaining dataset and rough elapsed time each step. Since the embedding dimension is 5, and the intrinsic dimension is 1.8557, according Observation 1, the count of the attributes that can be removed is at most 3. It can be seen from the table that the elapsed time decreased as more attributes are dropped.

| Attribute dropped | CFD | CPU elapsed time |
|---|---|---|
| | 1.8557 | 4.98 |
| *b* | 1.8423 | 3.38 |
| *d* | 1.8277 | 2.55 |
| *c* | 1.8507 | 2.00 |

Table.1 The comparison of attribute dropped, CFD of the remaining set and CPU time elapsed

## 6. Conclusions

In this paper we introduce a novel technique that can discover how many attributes are significant to characterize a dataset. We also present a fast, scalable algorithm to quickly select the most significant attributes of a dataset.

The main contributions of this paper are: (1) propose a new algorithm to compute CFD; (2) Based on the algorithm, design a new algorithm for FDR, whose time complexity is lower than the original algorithm [1]. It can detect the hidden correlations such as nonlinear and non-polynomial correlations existing in the dataset, spotting how many attributes strongly affect the behavior of the dataset regarding index and retrieval operations. We show by experiments that the algorithms are linear with the size of dataset, so they are scalable to large datasets.

**References**

[1] C Traina. Jr., A. Traina, L. Wu, and C. Faloutsos. Fast feature selection using the fractal dimension. In XV Brazilian Symposium on Databases (SBBD), 2000.

[2] P. Langley and S. Sage. Scaling to domains with irrelevant features. In R. Greiner, editor, Computational Learning Theory and Natural Learning Systems, volume 4. MIT Press, 1994.

[3] D.W. Aha, D. Kibler, and M. K. Albert. Instance based learning algorithms. Machine Learning, 6:37–66, 1991.

[4] P. Langley and S. Sage. Induction of selective Bayesian classifiers. In Proceedings of the Tenth Conference on Uncertainty in Artificial Intelligence, Seattle, W.A, 1994. Morgan Kaufmann.

[5] J.R. Quinlan. C4.5: Programs for machine learning. Morgan Kaufmann, Los Altos, California, 1993.

[6] R. Kohavi and G. John. Wrappers for feature subset selection. Artificial Intelligence, special issue on relevance, 97(1–2):273–324, 1996.

[7] A. L. Blum and P. Langley. Selection of relevant features and examples in machine learning. Artificial Inteligence, vol. 97, pp. 245-271, 1997.

[8] Mark A.Hall. Correlation-based feature selection for machine learning. PH.D Thesis, 1999.

[9] A. Belussi and C. Faloutsos. Estimating the selectivity of spatial queries using the correlation fractal dimension. In 21th Intl. Conf. on Very Large Data Bases (VLDB), Zurich, Switzerland, 1995.