# A density based approach to classification

Hui Wang*
School of Computing and Mathematics
University of Ulster
Belfast, BT37 0QB, UK
H.Wang@ulst.ac.uk

David Bell
School of Computer Science
Queen's University Belfast
Belfast BT7 1NN, UK
da.bell@qub.ac.uk

Ivo Düntsch
Computer Science Department
Brock University
St Catharines, Ontario, L2S 3A1, Canada
duentsch@cosc.brocku.ca

## ABSTRACT

This paper presents a novel method for classification, which is density based and makes use of the models built by the lattice machine (LM) [5, 7]. Density is a natural concept to use in clustering and the LM is a relatively new method for supervised learning developed in recent years. The LM approximates data resulting in, as a model of data, a set of hyper tuples that are equilabelled, supported and maximal. The method presented in this paper uses the LM model of data to classify new data with a view to maximising the density of the model. In order for the method to have wide applicability a measure of density is introduced for hyper tuples and relations.

Experiments were carried out with both public and proprietary data. Experimental results show that our method outperforms the classification method in the LM literature and it is comparable to the C5.0 classification algorithm. It is also shown that our method works quite well in an application - stock market data mining.

## Keywords

classification, lattice machine, density, hyper tuple, hyper relation.

## 1. INTRODUCTION

The lattice machine (LM) is a general algebraic framework for supervised learning [5, 7]. It aims to find as a model of data a hyper relation, called the *LM-model*, such that each hyper tuple in the hyper relation is equilabelled, supported, and maximal. Being equilabelled means the model is consistent with data; being maximal means the model has generalisation capability; and being supported means the model does not generalise beyond the information given in the data.

*Corresponding author.


SAC 2003, Melbourne, Florida, USA


This coincides with the *least general generalisation* principle [3] in inductive logic programming.

When data come from Euclidean space, the model is a set of hyper rectangles consistently, tightly and maximally approximating the data. This approach is different from decision tree induction, which aims to partition the given data. Figure 1 shows a 2D dataset and the models targeted by decision tree induction and the lattice machine respectively.

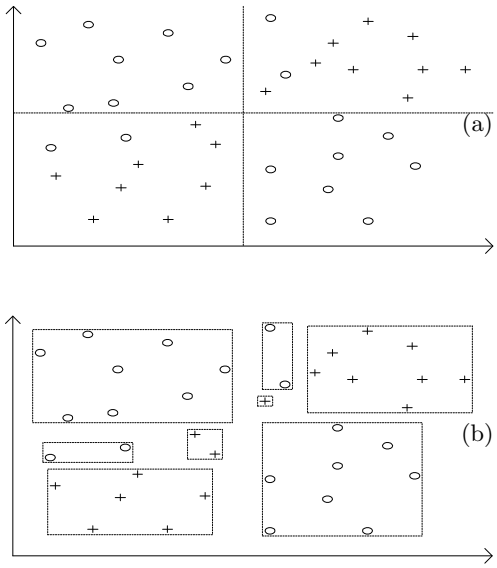


**Figure 1:** ***(a) Partition of data – decision tree approach. (b) Approximation of data - lattice machine approach. Each rectangle is a hyper tuple, which is equilabelled and supported. The class-separating margin between the rectangles is clearly higher than that in the partition approach used in decision tree induction.***

When such a model is obtained, classification can be done by the C2 algorithm [7]. C2 distinguishes between two types of data: those that are covered by at least one hyper tuple, and those that are not. Here we call the former *primary data* and the latter *boundary data*. Classification is done based on two measures – $C^1$ and $C^2$. For details the reader

is invited to consult [7]. Primary data $t$ is put in the same class as a hyper tuple which covers $t$, and boundary data are classified by the two measures. Figure 2 illustrates the C2 algorithm.

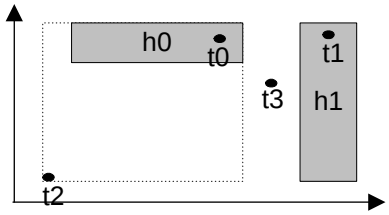


**Figure 2:** ***Illustration of the C2 algorithm:*** $h_0$ ***and*** $h_1$ ***represent two hyper tuples in a model with different class labels.*** $t_0$ ***is classified with*** $h_0$ ***and*** $t_1$ ***with*** $h_1$ ***since they are within the respective hyper tuples.*** $t_2$ ***is classified with*** $h_0$ ***since*** $h_0 + t_2$ ***does not overlap*** $h_1$***.*** $t_3$ ***can be classified with both*** $h_0$ ***and*** $h_1$ ***since neither*** $t_3 + h_0$ ***nor*** $t_3 + h_1$ ***overlaps the other hyper tuple. The exact classification depends on the order of the hyper tuples in the model.***

We implemented the C2 algorithm and carried out experiments on some benchmarking data, and the result is shown in Table 1. From this table we can clearly see that C2 performed extremely well on primary data. Since the overall performance is the weighted average of those of primary and boundary data, it is clear that C2 performed poorly on boundary data. So if we want to improve the performance of the C2 algorithm we should concentrate on boundary data. This led us to the development of a different approach to classification, which is reported here.

| Dataset | Prediction success (%) | | | | %PP |
|---|---|---|---|---|---|
| | C4.5 | C5.0 | LM/C2 | Prim. | |
| Annealing | 91.8 | 96.6 | 93.6 | 98.0 | 92.5 |
| Australian | 85.2 | 90.6 | 83.5 | 95.1 | 87.2 |
| Auto | 72.2 | 70.7 | 76.1 | 86.8 | 56.1 |
| Diabetes | 72.9 | 72.7 | 71.7 | 71.0 | 66.8 |
| German | 70.5 | 71.7 | 72.5 | 72.6 | 65.4 |
| Glass | 81.3 | 80.4 | 82.7 | 87.6 | 79.4 |
| Heart | 77.1 | 77.0 | 77.0 | 82.1 | 61.5 |
| Iris | 94.0 | 94.7 | 92.7 | 97.6 | 82.7 |
| Sonar | 69.4 | 71.6 | 69.7 | 81.4 | 59.2 |
| TTT | 86.2 | 86.2 | 83.5 | 96.2 | 94.9 |
| Vote | 95.1 | 96.5 | 94.2 | 98.5 | 89.7 |
| Wine | 94.3 | 94.3 | 94.4 | 98.9 | 53.9 |
| Average | 82.5 | 83.6 | 82.6 | 88.8 | 74.1 |

**Table 1:** ***Prediction success of C4.5, C5.0 and LM/C2 on all data, and LM/C2 on primary data. The "PP" column is the percentage of primary data.***

In our search for a method to improve the LM we observed that the notion of density has been successfully used for clustering [4, 2, 8] and wondered if the problem could be solved by a density approach. This effort has resulted in a novel classification method – *LM/Dens*. *LM/Dens* starts with an LM-model (a hyper relation) of data from the lattice machine, and classifies new data with a view to maximising the density of the hyper relation.

In the rest of the paper we first of all present a brief review of the lattice machine. We then present our definition of density for hyper relations, which is expected to accommodate different situations. The *LM/Dens* method is then presented. Experimental results using both public and proprietary data are then presented, and the paper concludes with a summary.

## 2. THE LATTICE MACHINE

Let $\mathbf{D}$ be a relation with a schema $\Omega = \{x_1, \cdots, x_T\}$ and domains $V_x$ of $x \in \Omega$. Let $d$ be an onto mapping, $d : \mathbf{D} \twoheadrightarrow V_d = \{d_0, \ldots, d_{\mathbf{K}}\}$.

$\mathbf{D}$ is the generic dataset, $x \in \Omega$ is an attribute, and $d$ is called a *labelling* of $\mathbf{D}$; the value $d(t)$ is called the *label of* $t$. $d$ is usually referred to as the *decision attribute*. The pair $\langle \mathbf{D}, d \rangle$ is referred to as a *decision system*.

The mapping $d$ induces a partition $\mathcal{P}_d$ of $\mathbf{D}$ with the classes $\{\mathbf{D}_0, \ldots, \mathbf{D}_{\mathbf{K}}\}$, where $t \in \mathbf{D}_i \iff d(t) = d_i$.

Let $\mathcal{L} \stackrel{\text{def}}{=} \prod_{x \in \Omega} 2^{V_x}$. Then $t \in \mathcal{L}$ is a vector $\langle t(x) \rangle_{x \in \Omega}$, where $t(x) \subseteq V_x$ are sets of values [1]. The elements of $\mathcal{L}$ are called *hyper tuples*; the elements $t$ of $\mathcal{L}$ with $|t(x)| = 1$ for all $x \in \Omega$ are called *simple tuples*. Any set of hyper tuples is called a *hyper relation*, and any set of simple tuples is called a *simple relation*

$\mathcal{L}$ is a lattice under the ordering

$$t \leq s \iff t(x) \subseteq s(x)$$

with the sum and product operations, and the maximal element (i.e., 1) given by

$$\begin{aligned} t + s &= \langle t(x) \cup s(x) \rangle_{x \in \Omega}, \\ t \times s &= \langle t(x) \cap s(x) \rangle_{x \in \Omega}, \\ 1 &= \langle V_x \rangle_{x \in \Omega}. \end{aligned}$$

$\mathcal{L}$ is called *domain lattice* for $\mathbf{D}$.

For $h \in \text{Ł}$ we write $\downarrow h$ for $\{x \in \mathcal{L} : x \leq h\}$.

$d$ can then be extended over all of $\mathcal{L}$ by setting

$$d(r) = \begin{cases} d_q, & \text{if } r \text{ is equilabelled and there} \\ & \text{is } t \in \mathbf{D} \text{ such that } t \leq r \\ & \text{and } d(t) = d_q; \\ \mathit{unknown}, & \text{otherwise.} \end{cases} \quad (1)$$

The dataset $\mathbf{D}$ can be naturally embedded into $\mathcal{L}$ by assigning

$$t \mapsto \langle \{t(x_1)\}, \{t(x_2)\}, \cdots, \{t(x_T)\} \rangle.$$

and we shall identify $\mathbf{D}$ with the result of this embedding. Thus we have $\mathbf{D} \subseteq \mathcal{L}$.

The lattice machine aims to find, as a model of data, a hyper relation such that each hyper tuple is *equilabelled*, *maximal* and *supported*.

An element $h \in \mathcal{L}$ is called *equilabelled* with respect to $\mathbf{D}_q$, if $\emptyset \neq \downarrow h \cap \mathbf{D}$, $\downarrow h \cap \mathbf{D} \subseteq \mathbf{D}_q$. In other words, $h$ is equilabelled if $\downarrow h$ intersects $\mathbf{D}$, and every element in this intersection is labelled $d_q$ for some $q \leq \mathbf{K}$.

$h$ is called *maximal* if $h$ is equilabelled and there is no equilabelled $s \in \mathcal{L}$ such that $h \lneq s$. $h$ is called *supported* by $\mathbf{D}$ if there is $X \subseteq \mathbf{D}$ such that $h = \sum_{x \in X} x$.

[1]Note that if $t \in \mathcal{L}$ and $x \in \Omega$, then $t(x)$ is the projection of $t$ to its $x$–th component. In practical terms, $t(x)$ can be treated as a set if $x$ is a categorical attribute, and it can be treated as an interval if $x$ is numerical.

It has been shown [6] that the model exists and is unique. The lattice machine is an approximation approach to classification – approximating the data by hyper tuples (regions, if data are numerical). This is in contrast to the partition approach adopted by decision tree induction, which aims to partition the data.

The approximation approach has a bigger class-separating margin than the partition approach, as can be seen in Figure 1. Coincidentally the support vector machines aim to find a class-separating hyper plane with maximal margin.

The LM algorithm [7] is able to find a hyper relation satisfying all three conditions (equilabelled, maximal, supported). However the LM algorithm is slow and hence is unsuitable for large datasets. An efficient algorithm, *CaseExtract*, is proposed for modelling [5], and the model found is equilabelled and supported, but not guaranteed to be maximal.

For classification the C2 algorithm is proposed [5, 7]. The lattice machine, coupled with the CaseExtract and C2 algorithms, achieved classification accuracy comparable to that of C4.5 [5].

## 3. DENSITY OF HYPER TUPLES

In this section we present a measure of density for hyper tuples and hyper relations. In the sequel we shall use the same notation as in Section 2.

Consider a hyper tuple $h \in \mathcal{L}$. In order to measure the density of the hyper tuple we need to know the coverage (i.e., number of data points covered by it) and the volume of the hyper tuple. Since attributes have different domains and scales and they may be numerical or categorical, they need to be normalised to a single scale. Since $h$ can be a general hyper tuple as well as a simple tuple, we need the density measure to apply uniformly to simple tuples and hyper tuples. We note that in a simple tuple each field is a single value, and so we must give a magnitude to a single value in order for a simple tuple to have a volume. To this end we need to *quantize* all attributes. Based on this understanding we introduce the following definitions.

First of all we need a (non-quantized) measure of magnitude for any subset of an attribute domain.

DEFINITION 1. *Let $h \in \mathcal{L}$ be a hyper tuple, and $x \in \Omega$ be an attribute. The* magnitude *of $h(x)$ is defined as*

$$\mathrm{mag}'(h(x)) = \begin{cases} \max(h(x)) - \min(h(x)), & \\ \qquad\qquad \textit{if } x \textit{ is numerical and } |h(x)| > 1 \\ |h(x)|, \qquad \textit{otherwise} \end{cases}$$

Note that $h(x)$ is the projection of $h$ onto attribute $x$, $\min(h(x))$ is the minimal value in $h(x)$ while $\max(h(x))$ is the maximal value.

We do not assume knowledge of significance of individual attributes, so we have to treat all attributes equally. This is another reason for normalising the attributes to a single scale. Normalisation can be achieved as follows.

DEFINITION 2. *Let $\lambda \in \mathbb{R}^+$ be a number chosen by the user. For an attribute $x \in \Omega$, the* normalisation coefficient *is $s(x) \stackrel{\text{def}}{=} \lambda / \mathrm{mag}'(V_x)$, where $V_x$ is the domain of attribute $x$.*

The parameter $\lambda$ relates to the granularity of attributes in the normalisation process so it is called the *granularity coefficient* of normalisation. The larger the $\lambda$ the finer the granularity.

DEFINITION 3. *Given a granularity coefficient $\lambda$, the* measurement of a unit *of attribute $x \in \Omega$ is $u(x) \stackrel{\text{def}}{=} \mathrm{mag}'(V_x)/\lambda = 1/s(x)$.*

With a unit measurement for every attribute, a hyper tuple $h$ can be *quantized* attribute by attribute as follows: if $\mathrm{mag}'(h(x))$ is less than $u(x)$ the $x$-dimension of $h$ should be treated as a unit; otherwise the $x$-dimension of $t$ is treated as $\mathrm{mag}'(h(x))/u(x)$ units. This leads to the following definition.

DEFINITION 4. *The* quantized magnitude *of $h(x)$ is defined as*

$$\mathrm{mag}(h(x)) = \begin{cases} 1, \textit{ if } \frac{\mathrm{mag}'(h(x))}{u(x)} < 1 \\ \frac{\mathrm{mag}'(h(x))}{u(x)}, \textit{ otherwise.} \end{cases}$$

In the sequel whenever we talk about magnitude we refer to quantized magnitude unless otherwise indicated.

DEFINITION 5. *The* volume *of $h$ is defined as*

$$\mathrm{vol}(h) = \prod_{x \in \Omega} \mathrm{mag}(h(x))$$

DEFINITION 6. *The* coverage *of $h$ is* $\mathrm{cov}(h) \stackrel{\text{def}}{=} \{t \in \mathbf{D} : t \leq h\}$.

DEFINITION 7. *The* density *of $h$ is defined as*

$$\mathrm{den}(h) = \frac{|\mathrm{cov}(h)|}{\mathrm{vol}(h)}$$

*The* density *of hyper relation $H$,* $\mathrm{den}(H)$*, is then the average density of the hyper tuples in $H$.*

To illustrate the above definitions, we first of all consider the simple relation in Table 2. For any tuple $t$ in this table, if $\lambda > 1$ then $\mathrm{mag}(t(x)) = 1$ for $x \in \Omega$. As a result $\mathrm{vol}(t) = 1$ by definition. Since a simple tuple covers only itself, i.e., $\mathrm{cov}(t) = \{t\}$, we have $\mathrm{den}(t) = 1$. This is to say that every simple tuple has a unit density, and so is any simple relation.

Now we consider the hyper relation in Table 3. This is the model of the data in Table 2 built by the LM algorithm[6]. If the granularity coefficient is fixed at 4, the normalisation coefficients are $s(A_1) = 2/3$ and $s(A_2) = 4/9$, the volumes are $\mathrm{vol}(h_0) = 8/3$ and $\mathrm{vol}(h_1) = 32/9$, and the coverage values are $\mathrm{cov}(h_0) = \{t_0, t_2, t_4, t_6\}$ and $\mathrm{cov}(h_1) = \{t_1, t_3, t_5, t_7\}$. So the densities of hyper tuples are: $\mathrm{den}(h_0) = 4/\mathrm{vol}(h_0) = 3/2 = 1.5$ and $\mathrm{den}(h_1) = 9/8 = 1.125$. As a result the density of this hyper relation is 1.313.

The notion of density is used in the literature for many different purposes. The most relevant use is in clustering [4, 2, 8], but in these studies density was defined mainly for numerical attributes and it was not quantized. Our definition of density applies to both numerical and categorical attributes and, since normalised and quantized, can be used to compare among hyper tuples and among hyper relations.

## 4. CLASSIFICATION VIA MAXIMISING DENSITY

| | $A_1$ | $A_2$ | $d$ |
|---|---|---|---|
| $t_0$ | $a$ | 2 | 1 |
| $t_1$ | $f$ | 10 | 2 |
| $t_2$ | $c$ | 4 | 1 |
| $t_3$ | $f$ | 9 | 2 |
| $t_4$ | $c$ | 3 | 1 |
| $t_5$ | $e$ | 7 | 2 |
| $t_6$ | $b$ | 1 | 1 |
| $t_7$ | $d$ | 6 | 2 |

**Table 2:** ***A relation on the scheme*** $\{A_1, A_2, d\}$ ***as a dataset, where attribute*** $A_1$ ***is categorical,*** $A_2$ ***is numerical, and*** $d$ ***is decision attribute. The values in the*** den***() column are the densities of the hyper tuples.***

| | $A_1$ | $A_2$ | $d$ | den() |
|---|---|---|---|---|
| $h_0$ | $\{a, b, c\}$ | $\{1, 2, 3, 4\}$ | 1 | 1.500 |
| $h_1$ | $\{d, e, f\}$ | $\{6, 7, 9, 10\}$ | 2 | 1.125 |

**Table 3:** ***The hyper relation obtained by the lattice machine as a model of the data in Table 2.***

Having a notion of density as defined above we now present our classification method. Our philosophy for classification is *classify data into tuples to maximally increase the density of hyper tuples.*

We assume that we have an LM-model built by the CaseExtract algorithm [5]. Our objective is to design a classification algorithm that works with this model and performs well on boundary data.

Let $t$ be a data (simple) tuple to be classified and $H = \{h_0, h_1, \cdots, h_n\}$ be an LM-model, where $h_i$ are equilabelled and supported hyper tuples. Our algorithm is called *Dens*, and it goes as follows.

- If $t$ is primary, i.e., there is $h_i$ such that $t \leq h_i$, then label $t$ by $d(h_i)$. Note that $d$ is the labelling function and it has been extended to $\mathcal{L}$ in Eq.1.
- Otherwise ($t$ is boundary), i.e., there is no $h_i$ such that $t \leq h_i$,
  1. Calculate $h'_i \stackrel{\text{def}}{=} t + h_i$ for $i = 0, \cdots, n$;
  2. Calculate $\text{den}(h'_i)$ for $i = 0, \cdots, n$;
  3. Label $t$ by $d(h'_m)$ such that for $j \neq m$
$$(\text{den}(h'_m) - \text{den}(h_m))/\,\text{den}(h_m) > (\text{den}(h'_j) - \text{den}(h_j))/\,\text{den}(h_j).$$

The *Dens* algorithm classifies primary data in the same way as the C2 algorithm. It classifies boundary data by the hyper tuple that carries the maximal rate of increase in density.

It should be noted that $h_m$ is the only hyper tuple in the model affected by the classification, and either $\text{den}(h'_m) > \text{den}(h_m)$ or $\text{den}(h'_m) < \text{den}(h_m)$ is true. In the former case the classification leads to the highest rate of increase in density for the affected hyper tuple while in the latter case it leads to the lowest rate of decrease in density for the affected hyper tuple.

It should also be noted that $(\text{den}(h'_m) - \text{den}(h_m))/\,\text{den}(h_m) > (\text{den}(h'_j) - \text{den}(h_j))/\,\text{den}(h_j)$ for $j \neq m$ does not means that $\text{den}(h'_m) > \text{den}(h_j)$, where $H'_i = \{h_0, \cdots, h_{i-1}, h'_i, h_{i+1}, \cdots, h_n\}$. As a result it is not guaranteed that $\text{den}(H'_m) > \text{den}(H'_j)$ for $j \neq m$. In other words absorbing a new data tuple into an existing hyper tuple leads to a new hyper relation. The new hyper relation may not have the highest density, but the affected hyper tuple has the highest rate of increase (or equivalently, lowest decrease) in density.

It is clear that the complexity of the *Dens* algorithm is at worst $O(n)$, where $n$ is the number of hyper tuples in the model. It is worth noting that $n$ is expected to be only a fraction of $|\mathbf{D}|$ – the size of dataset. In the experiment reported in [6], $n$ is between 1.5% and 18% of the size of data.

Our method is similar in spirit to the nearest neighbour (NN) method since the notion of volume measures, in a sense, the "distance" between a simple tuple and a hyper tuple. Our method goes beyond "distance" to take into account the coverage of hyper tuples. Furthermore our method works for both numerical attributes and categorical attributes in a uniform way, while the NN method can be problematic with categorical attributes for which an ordering does not exist.

# 5. EVALUATION

We implemented the *Dens* algorithm together with the *CaseExtract* algorithm [5] - a variation of the LM algorithm [6, 7]. We refer to this implementation as *LM/Dens*. We experimented with *LM/Dens* using some public datasets. The datasets are described in Table 4, which are available from UC Irvine Machine Learning Repository [2]. The experimental results are shown in Tables 5, along with C5.0 results on the same datasets [3]. It is clear from this table that *LM/Dens* outperforms *LM/C2* and it is comparable to C5.0. Furthermore *LM/Dens* performs significantly better for primary data than for the whole set of data.

The $\lambda$ parameter affects classification accuracy since it determines the granularity at which we quantize data to calculate volume and density. Figure 3 shows that the larger the $\lambda$ the better the classification accuracy, but this saturates at some points.

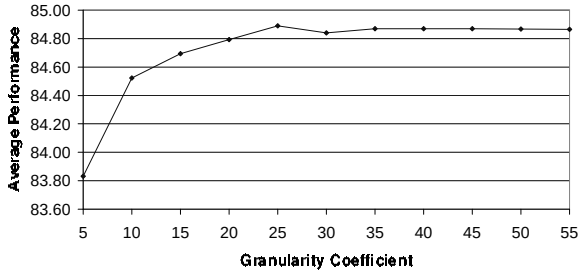


**Figure 3:** ***Average performance of all datasets vs granularity coefficient, showing the effect of*** $\lambda$ ***on the prediction accuracy in classification.***

We also incorporated *LM/Dens* into an experimental stock market data mining system - *StockMiner* - to select stocks

[2] `http://www.ics.uci.edu/~mlearn/MLRepository.html`

[3] We used the C5.0 module in the Clementine data mining package. Details on Clementine can be found at `http://www.spss.com/clementine`.

for investment. In this system standard technical indicators (see [1] for example) are calculated for individual stocks based on historical prices of stocks, and classification labels (*buy, hold* and *sell*) are determined for stocks with respect to changes in price at a later time. Thus we get a decision system. In our experiment we used 10 indicators including "Relative Strength" and "Moving Average". *LM/Dens* is then used to build an LM-model, which is then used to classify stocks. The stocks classified as *buy* or *sell* are then selected for investment purposes. Since primary data has significantly higher success rate in classification (see Table 5), only those selected stocks that are deemed *primary* by *LM/Dens* are presented to the user.

The result of an evaluation on FTSE100 is shown in Table 6, which shows that, on average, the selected stocks outperformed the index (FTSE100) by over 10% in 20 calendar days. This is a remarkable performance in terms of stock investment.

| Datasets | #Feature | #Example | #Class |
|---|---|---|---|
| Annealing | 38 | 798 | 6 |
| Australian | 14 | 690 | 2 |
| Auto | 25 | 205 | 6 |
| Diabetes | 8 | 768 | 2 |
| German | 20 | 1000 | 2 |
| Glass | 9 | 214 | 6 |
| Heart | 13 | 270 | 2 |
| Iris | 4 | 150 | 3 |
| Sonar | 60 | 208 | 2 |
| TTT | 9 | 958 | 2 |
| Vote | 18 | 232 | 2 |
| Wine | 13 | 178 | 3 |

**Table 4:** ***General information about the datasets.***

| Dataset | Prediction success (%) | | | | %PP |
|---|---|---|---|---|---|
| | C5.0 | LM/C2 | LM/Dens | Prim. | |
| Annealing | 96.6 | 93.6 | 96.1 | 98.0 | 92.5 |
| Australian | 90.6 | 83.5 | 92.7 | 95.1 | 87.2 |
| Auto | 70.7 | 76.1 | 73.2 | 86.8 | 56.1 |
| Diabetes | 72.7 | 71.7 | 69.9 | 71.0 | 66.8 |
| German | 71.7 | 72.5 | 69.3 | 72.6 | 65.4 |
| Glass | 80.4 | 82.7 | 83.6 | 87.6 | 79.4 |
| Heart | 77.0 | 77.0 | 78.1 | 82.1 | 61.5 |
| Iris | 94.7 | 92.7 | 95.3 | 97.6 | 82.7 |
| Sonar | 71.6 | 69.7 | 73.6 | 81.4 | 59.2 |
| TTT | 86.2 | 83.5 | 95.6 | 96.2 | 94.9 |
| Vote | 96.5 | 94.2 | 95.7 | 98.5 | 89.7 |
| Wine | 94.3 | 94.4 | 95.5 | 98.9 | 53.9 |
| Average | 83.6 | 82.6 | 84.9 | 88.8 | 74.1 |

**Table 5:** ***Prediction success of C5.0, LM/C2 and LM/Dens on all data, and LM/Dens on primary data. The validation method used is 5 fold cross validation in all cases. Note that the granularity coefficient $\lambda$ was set to*** 25 ***in the experiment, and the "PP" column is the percentage of primary data.***

## 6. CONCLUSION

The *LM/Dens* classification method reported here was developed to improve the classification accuracy of the lattice machine, and to demonstrate the value of density for classification.

Given an LM-model, *LM/Dens* classifies new data with a view to maximising the rate of increase in density. Experiment shows that the method does indeed improve the classification accuracy of the lattice machine, and it is comparable in prediction accuracy to C5.0 - the state of the art decision tree induction algorithm. This provides evidence that density is useful in classification. Application of *LM/Dens* to stock market data mining was also successful.

| RID | Group | AvgChgSel% | AvgChgInd% | Count |
|---|---|---|---|---|
| 24 | ftse100 | 2.63 | -0.04 | 23 |
| 22 | ftse100 | 6.27 | -23.90 | 28 |
| 25 | ftse100 | 9.73 | -9.05 | 12 |
| 23 | ftse100 | 10.56 | -2.69 | 45 |
| 28 | ftse100 | 16.98 | 0.79 | 7 |
| Average | ftse100 | 9.23 | -6.98 | 23 |

**Table 6:** ***20 calendar day average performance of the selected stocks between 20000101 and 20001117. Here RID is the ID of hyper tuple in the model, Group is the collection of stocks, AvgChgSel is the average percentage change in price of the selected stocks, AvgChgInd is the average percentage change of FTSE100 index in the same period, and Count is the number of selected stocks with*** buy ***class label.***

## 7. REFERENCES


[1] S. B. Achelis. *Technical Analysis From A to Z.* McGraw-Hill Professional Publishing, 2000.

[2] M. Ester, H. P. Kriegel, J. Sander, and X. Xu. A density-based algorithm for discovering clusters in large spatial databases with noise. In *Proc. 2nd Int. Conf. on Knowledge Discovery and Data Mining*, pages 226–231. AAAI Press, 1996.

[3] S. Muggleton and L. D. Raedt. Inductive logic programming: Theory and methods. *Journal of Logic Programming*, 19/20:629–679, 1994.

[4] E. Schikuta. Grid clustering: an efficient hierarchical clustering method for very large data sets. In *Proc. 13th Int. Conf. on Pattern Recognition*, volume 2, pages 101–105. IEEE Computer Society Press, 1996.

[5] H. Wang, W. Dubitzky, I. Düntsch, and D. Bell. A lattice machine approach to automated casebase design: Marrying lazy and eager learning. In *Proc. IJCAI99*, pages 254–259, Stockholm, Sweden, 1999.

[6] H. Wang, I. Düntsch, and D. Bell. Data reduction based on hyper relations. In *Proceedings of KDD98, New York*, pages 349–353, 1998.

[7] H. Wang, I. Düntsch, and G. Gediga. Classificatory filtering in decision systems. *International Journal of Approximate Reasoning*, 23:111–136, 2000.

[8] W. Wang, J. Yang, and R. Muntz. STING: A statistical information grid approach to spatial data mining. In *Proc. 23rd Int. Conf. on Very Large Databases*, pages 186–195. Morgan Kaufmann, 1997.