CHAPTER 3

KNOWLEDGE MANAGEMENT IN EDAPOLOGY USING SELF ORGANIZING MAPS AND LOCALIZED MATCHING MODEL FOR PLANT PREDICTION USING INCREMENTAL CLUSTERING

3.1 INTRODUCTION

The chapter presents two systems for knowledge management in Edaphology. The first system presents the knowledge management using Self Organizing Maps (SOM). The system is made of three modules consisting of dataset processing module, neuron training module and testing module. Dataset is collected from the Edaphologists and it comprises of soil characteristics and plant information. As the dataset is very large, it is reduced with the help of data type conversion process and normalization process and forms the operations in the dataset processing module. The SOM is created from the processed dataset in the neuron training module. In the testing module, user gives an input query and the plant names are outputted that are best matched to the query. In the second system, knowledge management is carried out using an incremental clustering technique. Here, at first an incremental DBSCAN (Density Based Spatial Clustering of Application with Noise) algorithm is given to a dynamic database in which the data is constantly updated. Subsequently, clustering of the soil data is carried out. Lastly, plant prediction is done with the use of the regression model. The systems are implemented in Java and evaluated under evaluation metrics. The evaluation metrics consist of the number of plants fetched for the user query, number of relevant plants in the retrieved list, time required for the process, memory utilized and the accuracy obtained for the
system. The experimental results have given good results indicating the effectiveness of the systems.

3.2 KNOWLEDGE MANAGEMENT IN EDAPHOLOGY USING SELF ORGANIZING MAPS

An effective knowledge management system in Edaphology is presented in this section and the system makes use of Self Organizing Map concept. Here, the SOM is constructed and from the built SOM, plants names retrieval is carried out for respective input query. The system mainly consists of three modules, namely dataset processing module, neuron training module and testing module. The system aims to help the Edaphologists in having the right plants for the soil attributes and this imparted knowledge will surely improve the productivity. The block diagram of the system using SOM is given in Figure 3.1.

3.2.1 Dataset Processing Module

The dataset comprises of various plant names and soil attributes obtained from the various Edaphologists and agriculturists and these are made used in evaluating the processed system. As the dataset is very vast and large, it needs to be reduced before neural training. If large data is given to the neural network, it will take a very long time to process and also increase the size of the system. Hence, it will reduce the effectiveness of the system and so the reduction of dataset is very essential. Reduction of dataset also results in minimizing the over-fitting tendency for the training data and also minimizes the correlation between features in order to make the training faster. Type conversion and normalization are used for decreasing the dataset size.
3.2.1.1 Type Conversion

The soil dataset collected from the Edaphologists consists of both the categorical data such as plant name and also numerical data. For categorical data, histogram density function is employed to change the string data to numeric valued data. It is also used to proficiently compute the mean and standard deviation. Here, reduction of data is carried out based on the idea of grouping similar data based on histogram sample value. And subsequently, the processes are carried out based on the group value rather than for each individual data which largely reduces the computation time and also reduces the
number of operations. The area of the histogram is given by the number of data in consideration. Normalization of the histogram is carried out using the relative frequencies which directly tells the count of data that is assigned for each category. Each column of the string variable data is taken to have the exclusive string values in a type conversion process. Here, let the exclusive string word be represented by $\omega_s$, frequency of exclusive string word be represented by $f_s$ and number of categorical string values be represented by $n_s$. The probability of exclusive string value $p_s$ is given by: 

$$p_s = \frac{f_s}{n_s}.$$ (3.1)

### 3.2.1.2 Normalization

After type conversion, the data is normalized to make sure the arithmetical scattering of the data values for every input and output is almost in the same interval. Apart from this, the data values ought to be scaled to the range of the input neurons which is carried out by normalization of each input in addition to other operations carried out in the network. Z-score based normalization procedure negates the relativism problems and also can be employed in case to assign points. The transformation of the data is done to have an average of zero and standard deviation of one. The zero-average scenario successfully eliminates aggregation distortions which results from changes in indicators’ means. Standard deviation of the indicator across the converted values forms the scaling factor. Hence, the indicator having large value would essentially result in better outcome for the composite indicator. Subsequently, the $z$-scores are transformed to an index, which means that all the scores for every ranking lies in the equal range. Let $S_{xy}$, $0 < y \leq V_{xs}$ be the representation of the values of the $x^{th}$ column where $V_{xs}$ is the total number of values in the $x^{th}$ column. Mean $M_x$ and standard deviation $D_x$ of the $x^{th}$ column can be given by:
\[ M_x = \frac{\sum_{i=1}^{V_{ss}} S_{xy}}{V_{ss}}, \quad \lambda_x = \frac{\sum_{i=1}^{V_{ss}} S_{xy}^2}{V_{ss}}, \quad D_x = \sqrt{\lambda_x - (M_x)^2} \] (3.2)

The normalized values \( O_{xy} \) is obtained by computing the difference between the mean of the column and the original value. The resulted value is divided by the standard deviation of the column. The normalized value can be obtained by the equation (3.3)

\[ O_{xy} = \frac{S_{xy} - M_x}{D_x} \] (3.3)

By the type conversion and normalization process, the input data set is effectively reduced which would result in improved network performance.

### 3.2.2 Neuron Training Module

Artificial Neural Networks finds its application in major areas and of late it has been the center of attractions for many researchers. It finds the employment in analysis, modeling and classification operations.

Self-Organizing Map (SOM) is one of the artificial neural networks which is employed in the system. Self Organizing Map is also known as the Kohonen Map. Self Organizing map (SOM) is a type of Artificial Neural Network (ANN) in which unsupervised learning technique is employed to generate a two-dimensional illustration using the map for training samples. Neighborhood functions are used to uphold the properties of the input data which makes SOM dissimilar from various other artificial neural networks.

SOM comprises of elements known as neurons or rather called nodes. There is a weight vector and a position in the map linked with each individual node. The input data vector and the weight vector have the same dimension.
Normally, nodes are arranged in two-dimensional grid of hexagonal or rectangular shape. The method for assigning a vector from the data space to a node in the map is by calculating the distance and finding the node which is nearest to the data space vector by the distance metric. In a SOM, unsupervised learning mechanism is normally used in the training process. In the training process, the map is created using input samples and the process is known as vector quantization. Normally, there exist three diverse kinds of network initializations, namely random initialization, initialization using initial samples and linear initialization. Figure 3.2 shows the structure of the SOM.

![Figure 3.2 Structure of Self-Organizing Maps](image)

Random initialization is employed in the proposed system where random values are picked and allocated to the vector. Random initialization has found the application in the system as the input data is hidden or little known at the initialization time. Here, the input vector is represented by $v_k$, for $0 < k < d_i$,
where $d_v$ is the vector input dimension. Training of the SOM is carried out in a repetitive process and it comprises of sketching sample vectors from the input data set and in the process, it acts as if it is doing the teaching to the SOM. Let the weight vector be symbolized by $\vec{w}$. The training process comprises of picking the vector element using the similarity measure and subsequently, updating the vector values in the neighborhood of the vector element. The procedure is repeated for $n_t$ times and in each training stage, one sample vector is taken arbitrarily from the input data set. The vector in consideration is given to other element units present in the network and the similarity measure is computed between the input data sample and all the vectors. For similarity measure computation, navigate through every node present and compute the minimum distance between the weight vector and input vector. Let us consider the $k^{th}$ node, minimum distance $\Delta_k$ between the weight vector $\vec{w}$ and input vector $\vec{u}$ is computed as:

$$\Delta(\vec{w}, \vec{u}) = \sqrt{\sum_{i=1}^{d_v} (w_i - u_i)^2}$$

(3.4)

The vector with highest similarity with the input sample is taken as the best-matching unit or element. The similarity is computed based on the distance measure. Once the best-matching unit or element is found out, updation of elements in the SOM are carried out. In the updation process, the best-matching unit is updated to the slight nearer sample vector and is given by:

$$\vec{w}_i(\tau + 1) = \vec{w}_i(\tau) + \gamma(\vec{v}, \tau)\beta(\tau)(T(\tau) - \vec{w}_i(\tau))$$

(3.5)

Where, $\tau$ represents the present iteration, $\vec{w}_i(\tau)$ gives the weight vector, $T(\tau)$ represents the target input, $\gamma(\vec{v}, \tau)$ is the neighborhood function and $\beta(\tau)$ is the learning rate with respect to time. The best-matching unit’s neighbors are also likewise updated. The updation process results in expanding the BMU and its
neighbors in the direction of the sample vector. When the training of the SOM is completed, for each node, a respective plant is set depending on the values.

### 3.2.3 Testing Module

In the testing module, a test query is given as an input by the user which will have certain soil attributes and the system outputs the corresponding plant name that matches the query.

![Block diagram of the testing module](image)

**Figure 3.3 Block diagram of the testing module**

Initially, the input query is type converted and normalized and subsequently matched with the trained SOM. The system responds to the processed input query by returning the plant names which is related to input query and the process is carried out by the trained SOM. The similarity between the target plant and the input vector is calculated based on the cosine similarity measure. In case of any unknown query, the system outputs the plant which is nearest to the target plant depending on the similarities. This means that when the input data query is not exactly corresponding to the best matching unit, then the input query is matched for subsequent resemblance with the neighbors. The block diagram of the testing module is given in Figure 3.3. The system outputs the plant names best match to the input user query.
3.3 EXPERIMENTAL RESULTS

This section presents the results and discussions of the system for SOM based knowledge retrieval in Edaphology. Initially, SOM is trained and then the test query is given as input to get the plant names. Here, the number of plants retrieved for each query, time of computation and the memory used are measured.

3.3.1 Experimental set up and Dataset Description

The system is implemented in JAVA on a system having 4 GB RAM and 2.10 GHz Intel i-5 processor. Initially, the domain knowledge collected from Edaphologists is modelled into a knowledge base, which acts as the input dataset. The input dataset is type converted and normalized before training and testing processes.

3.3.2 Screen Shots of the System

In this section, the screen shots of the system are given. Figure 3.4 shows the home screen, Figure 3.5 shows the dataset connection, Figure 3.6 displays the details, Figure 3.7 shows the type converted data, Figure 3.8 shows the normalized data, Figure 3.9 shows the SOM values formed. Figure 3.12 - Figure 3.15 shows the output obtained for different input queries.
Figure 3.4 Home window

Figure 3.5 Dataset Connection
Figure 3.6 Showing the plant and soil details of the dataset

<table>
<thead>
<tr>
<th>Plant Origin Dataset</th>
<th>Type of Plant</th>
<th>Score</th>
<th>Score Type</th>
<th>Soil Type</th>
<th>Soil Characteristics</th>
<th>Plant Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Dark green leaves</td>
<td>Fungus</td>
<td>9.06</td>
<td>0.02</td>
<td>Clay</td>
<td>Moderately medium</td>
<td>Leaves are elongated and shiny.</td>
</tr>
<tr>
<td>Light green leaves</td>
<td>Bacteria</td>
<td>9.54</td>
<td>0.05</td>
<td>Loam</td>
<td>Moderately medium</td>
<td>Leaves are broad and flat.</td>
</tr>
</tbody>
</table>

Figure 3.7 Showing the type converted data

<table>
<thead>
<tr>
<th>Plant Origin Dataset</th>
<th>Converted Score</th>
<th>Converted Type</th>
<th>Converted Score Type</th>
<th>Converted Soil Type</th>
<th>Converted Soil Characteristics</th>
<th>Converted Plant Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Dark green leaves</td>
<td>9.06</td>
<td>Fungus</td>
<td>0.02</td>
<td>Clay</td>
<td>Moderately medium</td>
<td>Leaves are elongated and shiny.</td>
</tr>
<tr>
<td>Light green leaves</td>
<td>9.54</td>
<td>Bacteria</td>
<td>0.05</td>
<td>Loam</td>
<td>Moderately medium</td>
<td>Leaves are broad and flat.</td>
</tr>
</tbody>
</table>
Figure 3.10 Results obtained for test case input query 1

Figure 3.11 Results obtained for test case input query 2
Figure 3.12 Results obtained for test case input query 3
Figure 3.13 Results obtained for test case input query 4
For the queries mentioned in the Figures 3.10 to 3.13, the number of
plants retrieved is given in the Table 3.1.

Table 3.1 Number of Plants retrieved

<table>
<thead>
<tr>
<th>Input query</th>
<th>Number of plants Retrieved</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>15</td>
</tr>
<tr>
<td>2</td>
<td>20</td>
</tr>
<tr>
<td>3</td>
<td>21</td>
</tr>
<tr>
<td>4</td>
<td>18</td>
</tr>
</tbody>
</table>

3.4 LOCALIZED MATCHING MODEL FOR PLANT PREDICTION USING INCREMENTAL CLUSTERING

Incremental DBSCAN is employed in the system for performing
knowledge management. It also performs the grouping of data into clusters. As
incremental clustering is used, there is no need of prior knowledge of the
number of clusters in the database. The prediction of the plant from the local
subset is carried out using the linear regression equation. The system is
composed of three modules namely: Preprocessing module, clustering module
and plant prediction module. In preprocessing module, operations such as type
conversion and normalization are carried out. Incremental DBSCAN is used in
the clustering module to effectively perform the clustering task and in plant
prediction module; linear regression is employed to successfully predict the
plant names in accordance with the user input query having different levels of
soil attribute values. The block diagram of the system is given in Figure 3.14.
3.4.1 Preprocessing Module

The dataset used in the system consists of soil characteristics and plant details collected from various Edaphologists and agriculturists. The data set comprises of lots of soil features and every feature has its own range of values. To standardize the values to a particular range, numerical conversion process of type conversion and normalization are employed. The dataset has a lot of attributes and the range of variance of the particular attribute is also high. Hence, the data values are confined to particular range through numerical value conversion. This would decrease the data size and also the computation time.
In type Conversion, the conversion of the string values in the dataset to numerical values using lookup tables is carried out. Here, Hash table is used for the storage of unique values and its corresponding bit locations. Unique string values are found out by reading each column of the variable from the dataset. In normalization, the aim is to make sure that the input and output almost lies in the same range. The normalized input must be comparable with the scaled values. Normalization results in having reduced range of variables. Min-max normalization technique is employed to match the variable to the respective target plant. The min-max based normalization is given by the formula:

\[ V_{nor} = \left\{ \left[ \frac{M_c - M_{mi}}{M_{ma} - M_c} \right] \cdot (V_{ma} - V_{mi}) + V_{mi} \right\} \]

(3.6)

Where, \( V_{nor} \) is the normalized value, \( M_c \) is the column value, \( M_{mi} \) is minimum column value, \( M_{ma} \) is maximum column value, \( V_{mi} \) is minimum value to be converted, \( V_{ma} \) is maximum value to be converted.

### 3.4.2 Clustering Module

Clustering of large data, groups the similar data together and forms a beneficial method to find out information from a vast database. There are many clustering techniques and in this system, an incremental DBSCAN algorithm is employed which can be used for dynamic datasets. The algorithm also possesses the benefit that the number of clusters need not be known in the initial stage. It also negates the rerunning process when updating the database.

DBSCAN which stands for Density-Based Spatial Clustering of Applications with Noise is one of the data clustering algorithms which finds applications in diverse fields. It is essentially a density-based clustering algorithm which found opening from the projected density distribution of matching nodes. DBSCAN has many advantages which are described below.
Here in DBSCAN, the number of clusters need not be known earlier. Another advantage is that it can discover arbitrarily shaped clusters. It can even differentiate a cluster totally enclosed by a dissimilar cluster. The attributes required by this algorithm are eps (epsilon) and the minimum number of points essential to build a cluster. Here, it initiates with an arbitrary initial point that has not been visited. Subsequently, the neighborhood of the point is taken into consideration and the algorithm traverses the evaluation procedure recursively. If it consists of sufficient number of points, a new cluster is formed. Otherwise, the respective point is branded as noise. In the case where the cluster is fully expanded, then the algorithm continues to repeat through the residual unvisited points in the dataset. Figure 3.15 gives the clustering of a set of objects based on the DBSCAN where the radius is taken as 1 cm and minimum number of points is taken as 5.

![Figure 3.15 Clustering of a set of objects based on the DBSCAN](image)

The flow of the algorithm is described below.

a. Initially, read the first record and let it be represented as $R_1$

Subsequently set $R_1$ as $C_1$, where $C_1$ is the cluster 1.

b. Read all other records $R_k$, where $0 < k < m$ and $m$ is the total number of records.
c. Compute the distance \( d \) between \( \text{Re}_k \) and \( C_m \). The distance is given by the formula: 
\[
d_{km} = \sum_{k,m=1}^{x} |\text{Re}_k - C_m|^2
\]

d. After the distances are computed, sorting is carried out in the ascending order of \( d \).
e. Check for the condition that if the distance \( d_{km} \) is less than the threshold set \( T_h \). If the condition is true \((d_{km} < T_h)\), add the data to the minimum distance cluster and updation of cluster center is made.
f. If the condition is not met, a new cluster is formed.
g. Read the next record and go to step c.
h. Continue this process until all records are read.

Hence, the input large dataset is clustered into various groups with the help of incremental DBSCAN clustering. The pseudo code of the algorithm is given in Figure 3.16.

1. Read \( \text{Re}_1 \). Set \( \text{Re}_1 \) as \( C_1 \).

2. Read \( \text{Re}_1 \) to \( \text{Re}_k \)

3. Compute distance: 
\[
d_{km} = \sum_{k,m=1}^{x} |\text{Re}_k - C_m|^2
\]

4. Sort in the ascending order of \( d \).

5. If \( (d_{km} < T_h) \)
   
   Add the data to the minimum-distance cluster and update cluster center.

   Else

   New cluster is formed.

6. Read the next record and go to step 3 till all records are read.

Figure 3.16 The pseudo code of the incremental DBSCAN clustering
3.4.3 Plant Prediction module

In this module, the user inputs a query and the system outputs the corresponding plant names. The user query consists of soil attributes which are processed by type conversion and normalization. It is subsequently applied to incremental DBSCAN clustering process. The system generates plant names most matching to the input user query. The output plant names are produced with the help of the regression model based on soil features.

Regression model forms a part of statistics and is used mainly for finding out the relationships among the variables in consideration. There are many regression models that use different methods for making the various analyses, mainly focusing on the relationship among dependent variables or independent variables. Usually regression models assist in finding out the dependency of a variable change with respect to another independent variable taking all other independent variables as constant. Regression can also find out the dependent variable’s conditional expectation when the independent variables are given. Emphasis is also made on quantile or location parameter of the conditional distribution. Here, the regression function is formed by the estimated target’s independent variable function. Variation characterization of the dependent variable about the regression function is also analyzed in regression model. Regression models and analysis finds its major application in the fields of prediction and forecasting. It is also employed to find out the independent variables which are associated with the dependent variable and also to search all relationship ties within the variables of the system. Sometimes, regression can also be employed to infer causal relationships between the variables but this can give wrong relationships.

There has been a large improvement in the regression analysis and a lot of methods are found in the regression area. Regression models like linear
regression and ordinary least squares regression have finite count of unknown parameters and hence are parametric. In nonparametric regression, regression function may be infinite-dimensional and also confine to a specified set of functions. Regression model performance is largely dependent on the data generating process and also takes into account the regression method employed. Often the data-generating processes are not really known and assumptions are made in the case. But the case can be tested if large data is available. Though assumptions are used in the model, regression model finds its great application in data prediction.

In regression, data is matched to a regression function and the application of regression results in having a smoothened data. Here, the best regression line to match the attributes is found out. The regression can be of types of linear, nonlinear, and generalized linear. Regression mainly finds its application in forecasting of data. In many cases, nonlinear problems are converted to linear problems. Linear regression aims to model the relationship between a scalar dependent variable and other variables denoted. In simple linear regression, there will be only one explanatory variable. When more than one other variable is in consideration, it is known as multiple linear regressions.

Linear predictor functions and unknown model parameters are made use in the linear regression models. Linear regression can also be modeled in such a way that the median of the conditional distribution of scalar dependent variable can be stated as a linear function of other explanatory variable. Linear regression mainly concentrates on the conditional probability distribution rather than joint probability distribution. Linear regression is extensively employed in many practical applications as the model linearly associates unknown attributes easily than other models.
Equation for the linear regression line for prediction is given by:

\[ R = xG_1 + yG_2 + z \]  \hspace{1cm} (3.7)

Where, \( R \) is the linear regression value, \( G_1 \) is the vector length, \( G_2 \) is the angle between two vectors, \( x, y \) are the slopes, \( z \) is the y intercept. Regression line obtained for sample data is given in Figure 3.17.

![Figure 3.17 Regression line for sample data](image)

3.4.4 Experimental Results

This section presents the experimental results of the system. The input data are preprocessed with the help of type conversion, normalization and clustering. After that, the plants are found with the help of linear regression.

3.4.4.1 Experimental Set up And Dataset Description

The system is implemented in JAVA and the system requirement is 4 GB RAM and 2.10 GHz Intel i-5 processor. Initially, the information collected from domain experts, act as an input data set. Before training and clustering processes, the input dataset is type converted and normalized.
3.4.4.2 Screen shots of the system

The screen shots of the system are presented in this section.

Figure 3.18 Home screen of the system

Figure 3.19 Type conversions of Sample 10 records
Figure 3.20 Normalized Min-Max Values 0.0 to 1.0

Figure 3.21 clustered plants
Figure 3.22 Linear regressions

The following figures show the matched plants for the different types of soil characteristics.
Figure 3.23 Type 1 soil characters

Figure 3.24 Type 2 soil characters
Figure 3.25 Type 3 soil characters

Figure 3.26 Type 4 soil characters
Figure 3.27 Type 5 soil characters

For the soil characters mentioned in the Figures 3.23 to 3.27, the number of plants retrieved is given in the Table 3.2.

Table 3.2 Number of Plants retrieved

<table>
<thead>
<tr>
<th>Soil characters</th>
<th>Number of plants</th>
</tr>
</thead>
<tbody>
<tr>
<td>Type 1</td>
<td>2</td>
</tr>
<tr>
<td>Type 2</td>
<td>3</td>
</tr>
<tr>
<td>Type 3</td>
<td>3</td>
</tr>
<tr>
<td>Type 4</td>
<td>2</td>
</tr>
<tr>
<td>Type 5</td>
<td>2</td>
</tr>
</tbody>
</table>