A Heuristically Self-Organised Linguistic Attribute Deep Learning in Edge Computing For IoT Intelligence

Hongmei He

School of Aerospace, Transport and Manufacturing, Cranfield University, Cranfield, UK

Zhenhuan Zhu

Smart Perception Lab, Milton Keynes, UK

Abstract

With the development of Internet of Things (IoT), IoT intelligence becomes emerging technology. “Curse of Dimensionality” is the barrier of data fusion in edge devices for the success of IoT intelligence. Deep learning has attracted great attention recently, due to the successful applications in several areas, such as image processing and natural language processing. However, the success of deep learning benefits from GPU computing. A Linguistic Attribute Hierarchy (LAH), embedded with Linguistic Decision Trees (LDTs) can represent a new attribute deep learning. In contrast to the conventional deep learning, an LAH could overcome the shortcoming of missing interpretation by providing transparent information propagation through the rules, produced by LDTs in the LAH. Similar to the conventional deep learning, the computing complexity of optimising LAHs blocks the applications of LAHs.

In this paper, we propose a heuristic approach to constructing an LAH, embedded with LDTs for decision making or classification by utilising the distance correlations between attributes and between attributes and the goal variable. The set of attributes is divided to some attribute clusters, and then they are heuristically organised to form a linguistic attribute hierarchy. The proposed approach was validated with some benchmark decision making or classification problems from the UCI machine learning repository. The experimental results show that the proposed self-organisation algorithm can construct an effective and efficient linguistic attribute hierarchy. Such a self-organised linguistic attribute hierarchy embedded with LDTs can not only efficiently tackle ‘curse of dimensionality’ in a single LDT for data fusion with massive attributes, but also achieve better or comparable performance on decision making or classification, compared to the single LDT for the problem to be solved. The self-organisation algorithm is much efficient than the Genetic Algorithm in Wrapper for the optimisation of LAHs. This makes it feasible to embed the self-organisation algorithm in edge devices for IoT intelligence.

1Dr Hongmei He is with Cranfield University, Uk, email:h.he@cranfield.ac.uk or maryhhe@gmail.com
2Dr Zhenhuan Zhu is with Smart Perception Lab, UK, email:zhen-zhu@hotmail.com or zhenhuan2001@hotmail.com
1. Introduction

The Internet-of-Things (IoT) provides us with a large amount of sensor data. However, the data by themselves do not provide value unless we can turn them into actionable and/or contextualized information. Big data analysis allows us to gain new insights by batch-processing and off-line analysis. Currently, a microprocessor-based sensor node can support many channels (e.g. a Microchip processor can support up to 49 channel inputs [1]). Real-time sensor data analysis and decision-making is preferably automated on-board of IoT devices, which will make IoT intelligence towards reality.

Although the computing capability of a microprocessor has improved very much, the ‘curse of dimensionality’ is still a big challenge in data driven machine intelligence, as the computing complexity of designed model function increases as the increasing of input space. Blum and Rivest [2] have proved that training a 2-layer, 3-nodes and n inputs neural network is NP-Complete. Obviously, the big barrier of blocking the applications of deep-learning is the computing complexity, although it shows great attractive on solving complex nonlinear problems. With the strong capability of GPU, deep learning for 2-20 depth networks is successful (e.g. Google AlphaGo). To save the cost, an edge device of IoT systems may not need to equip with GPUs, if the machine intelligence algorithm inside the device is efficient enough. Hence, the performance improvement of computational intelligence is continuous work, especially for creating effective and efficient computing model in edge devices to implement IoT intelligence.

Linguistic decision tree (LDT), a probabilistic tree, has been well used for decision making and classification problems. Given an input space of n attributes, each of which can be described with limit labels. An LDT consists of a set of attribute nodes and a set of edges. An edge, linking from an attribute node, represents a label expression that describes the attribute node (see Section 3.2). However, the branch number of a decision tree exponentially increases as the number of input attributes increases. This shortcoming of an LDT greatly blocks its applications.

A hierarchical model could help overcome the “Curse of Dimensionality” in fuzzy rule-based learning [3]. Campello and Amaral [4] provided a cascade hierarchy of sub-models, using fuzzy relational equations [5], and unilaterally transformed the cascade models into the mathematically equivalent non-hierarchical one. Lawry and He[6] proposed a linguistic attribute hierarchy (LAH). It is a hierarchy of LDTs, each of which has a reduced input space, and represents different functions in the problem space. However, as the relationship between inputs and the output in the whole problem space could be strong non-linear and uncertain, different LAHs will have different performance for the problem to be solved. All of the research in [4, 6] neither investigated the performance of the proposed hierarchies, nor studied how the hierarchies can be constructed optimally.
He and Lawry [7, 8] investigated cascade LAHs, and developed a genetic wrapper algorithm (GWA) to optimise the cascade LAHs. It was shown that a cascade hierarchy can derive much less rules, compared to the single decision tree. However, the accuracy tends to drop as decision threshold increases. Later, they also investigated the optimisation of generic LAHs [9]. The optimisation of LAHs using the GWA is NP-complete, as the branch number of a decision tree is increasing exponentially as the input attribute number increases, and the convergency of GWA directly affects the speed of optimisation. Hence, the evolutionary algorithm takes very long time to optimise LAHs, even if the evolution takes a small number of iterations. When the attribute number is over 60, the time of evolution process on a PC is not acceptable.

Another issue of hierarchical approaches, including generic deep learning, is the lack of linguistic interpretability, especially neural networks have been viewed as a black box. This is because that intermediate variables, being arbitrarily introduced to interconnect the sub-models, do not present any meanings in a real system, and hence it is difficult to give a clear intuitive interpretation [4]. However, in an LAH, information propagation is completed via the linguistic rules extracted from LDTs. Hence, a transparent hierarchical decision making or classification can be provided.

In a distributed system, we may get partial information from different information resources, based on which, an initial estimate/decision could be done locally. Correspondingly, a collective decision is required. How do these information resources make contribution to the final collective decision? In an LAH, an intermediate variable is equivalent to the initial decision in terms of partial information [7, 8]. Hence, the structure of the hierarchy determines the process of decision or fusion process. He et. al. [10] further proposed an off-line optimisation of sensor arrangement for task-oriented sensor fusion, aligning with the structure of an self-organised wireless sensor network.

Mitra et al. [11] proposed a fuzzy knowledge-based network, whose structure can be automatically optimised by using linguistic rules of a fuzzy decision tree, thus to help in reducing the complexity of network training for a specific task. An LAH is equivalent to a forward neural network, in which, each neuron is the function of lower layer attributes, but represented by an LDT. The structure of an LAH is different to that of a classic multi-layer feed-forward neural network, in which, all input attributes are in the input layer. In an LAH, input attributes can feed a neuron (e.g. an LDT) at any layer, but for the simplicity of network structure, they cannot repeatedly feed more than one neuron.

The information propagation from bottom to top of an LAH through the LDTs in the LAH provides a new approach to attribute deep learning. In fact, a neural network or any other machine learning models can be used, instead of an LDT in the LAH. For example, a cascade of CMACs has been proposed in [12]. However, LAH embedded with linguistic decision trees could provide good interpretation for decision making or classification.

He et al. [13] used an LAH to interpret the process of semantic attribute deep learning for spam detection. The LAH was constructed manually in terms of the semantics of attributes. For IoT intelligence or edge computing, the onboard adaptive sensor fusion is a critical challenge, due to the limit of resources in edge devices (e.g. computing capacity, memory and even power supply). Hence, efficiently automatic constructing a
linguistic attribute hierarchy becomes necessary for this purpose.

In this research, a self-organised LAH (SOLAH), embedded with LDTs is proposed for the deep learning of attribute semantics in decision making or classification. A distance correlation-based clustering algorithm is proposed to decompose attributes to several clusters, and a linguistic attribute hierarchy is constructed with the produced clusters in terms of the average distance correlation of cluster members to the goal variable. The LDT, fed by a cluster of attributes with lower distance correlation, will be placed in the lower layer of the LAH. The preliminary experiments are conducted on the SMS spam database from UCI machine learning repository [14]. A set of databases will be used to validate the performance of the SOLAHs. The experimental results are compared with that of the single LDTs for different databases.

2. Existing work in Deep Learning

Recently, Deep Learning has been put a lot of attention by researchers. It uses multiple processing layers of computational models to learn representations of data with multiple levels of abstraction. Each successive layer is fed by the outputs of previous layer, forming a hierarchy of attributes from bottom to top. Deep learning uses the back-propagation algorithm to learn internal parameters that are used to compute the representation in each layer from the representation in the previous layer, thus to find the hidden structure or information in large data sets. Various deep learning architectures such as deep neural networks, convolutional deep neural networks, deep belief networks and recurrent neural networks have been developed. Deep learning has been successfully applied in the areas of speech recognition [15, 16, 24], image processing [17], object detection [18, 19], drug discovery [20] and genomics [21], etc. Especially, deep convolutional nets (ConvNets) have brought about breakthroughs in processing images [22], video [23], speech [24] and audio [25], whereas recurrent nets have shone light on sequential data such as text and speech [26].

ConvNets are constructed layer by layer for data processing. A classic application of a ConvNet is in image processing, utilising the properties of a colour image, consisting of the RGB channels of pixel values. Since the early 2000s, ConvNets have been successfully applied for the detection, segmentation and recognition of objects and regions based on images, for example, face recognition [27].

Druzhkov and Kustikova [28] did a survey on deep learning methods for image classification and object detection, covering autoencoders, restricted Boltzmann machines and convolutional neural networks. For example, Szegedy et al. [29] used deep neural networks to solve the problem of object detection for both classifying and precisely locating objects of various classes; Simonyan et al. [30] used deep fisher networks for the classification of large-scale images from ImageNet (http://image-net.org/); Krizhevsky et al. [31] trained a large deep convolutional neural network to classify the 1.3 million high-resolution images in the LSVRC-2010 ImageNet training set into the 1000 different classes; and He et al. [32] proposed an unsupervised feature learning framework, Deep Sparse Coding, which extends sparse coding to a multi-layer architecture for visual object recognition tasks.

Acoustic modeling is another area, where large, deep neural networks have been successfully applied. Like ImageNet, the massive quantities of existing transcribed
speech data provide rich resources for deep learning. Dahl [33] provided a brief review of the deep neural net approach to large vocabulary speech recognition (LVSR) in his thesis. Mohamed et al. [34] showed that hybrid acoustic models of pre-trained deep neural networks, instead of Gaussian mixture models (GMMs), could greatly improve the performance of a small-scale phone recognition. By using rectified linear units and dropout, Dahl et al. [35] further improve the model for a large vocabulary voice search task. A combination of a set of deep learning techniques has led to more than 1/3 error rate reduction over the conventional state-of-the-art GMM-HHM (Hidden Markov Model) framework on many real-world LVCSR tasks [36].

The qualitative properties of text data, very different to that of other modality data, provide critical challenges in use of machine learning. Recently Socher et al. [37] developed recursive deep models for semantic compositionality over a sentiment tree-bank, and improved the accuracy. Natural language understanding is another exploration of deep learning application, which could make a large impact over the next few years [26].

The property of compositional hierarchies of some signals is well exploited by deep neural networks through composing lower-level features to abstract the high-level one. For example, an image can be represented by a hierarchy from local edges, motifs, parts, to objects. Similarly, speech and text also have a hierarchy from sounds to phones, phonemes, syllables, words and sentences. In other words, this properties promote the capacities of deep neural networks. However, the interpretability of the decision making process is still an issue. Also, we cannot always explicitly see the semantics of higher-level features in other application domains.

With rapid progress and significant successes in a wide spectrum of applications, deep learning is being applied in many safety-critical environments. Ching et al. [47] forecasted that deep learning enabling changes at both bench and bedside with the potential to transform several areas of biology and medicine, although the limited amount of labelled data for training presents problems as well as legal and privacy constraints on work with sensitive health records. However, deep neural networks (DNNs) have been recently found vulnerable to well-designed input samples called adversarial examples, and adversarial perturbations are imperceptible to human but can easily fool DNNs in the testing/deploying stage [48]. The transparency of LAH allows the observation of the decision process. He et al. [13] have investigated the effect of different input attributes on different positions in a linguistic attribute hierarchy. This might indicate a linguistic attribute deep learning could provide good transparency to help people to defend adversarial perturbations to IoT Intelligence, which is out of the scope of this research.

GPU has become the necessary hardware facility for the research on deep learning due to its complexity. Justus et al. [46] analysed various factors that influence the complexity of deep learning, and divided these factors to three categorises: (1) Layer features, such as activate function, optimiser of weights (e.g. Gradient Descent) and number of training samples; (2) layer specific features, including four subcategories: i. Multi-Layer Perception features (e.g. number of inputs, number of neurons and number of layers), ii. Convolutional features (e.g. matrix size, kernel size, stride size, input padding, input depth and output depth, etc.); iii. Pooling features (e.g. matrix size, stride size, input padding), iv. Recurrent features (e.g. Recurrence type,
bi-directionality); (3) Hardware Features (e.g. GPU technology, GPU count, GPU memory, GPU memory bandwidth, GPU clock speed, GPU cord count, GPU peak performance, Card Connectivity). The first two categories of factors determine the performance of the function represented by a deep learning model, while the GPU feature directly affects the efficiency of the computing. It is hard to express the polynomial relationship between the computing cost and all the various factors, or use a big O(*) to represent the computing complexity.

3. Linguistic attribute deep learning with a LAH

3.1. A linguistic attribute hierarchy (LAH)

The process of aggregation of evidence in multi-attribute decision making or classification based on attributes \( x_1, ..., x_n \) can be represented with a functional mapping \( y = f(x_1, ..., x_n) \), where \( y \) is the goal variable. However, this mapping is often dynamic and uncertain, and it is difficult to find a mathematical equation to precisely describe the mapping function. An attribute hierarchy represents the function \( f \) with a hierarchy of sub-functions, each of which represents a new intermediate attribute. The set of original attributes \( \{x_1, ..., x_n\} \) is categorized into \( m \) clusters \( s_1, ..., s_m \). When a cluster of attributes is used to make initial decision, a new intermediate attribute is produced by the function of the clustering attributes. One or more intermediate attributes can be combined with another cluster of attributes to make further decision in next level, or multiple intermediate attributes can be directly used to make decision in next level. As there are \( m \) attribute clusters, at least \( m \) subfunctions are produced. Namely, \( z_i = G_i(s_i) \) for \( i = 1, ..., m \) and \( n - 1 \geq m \), as the maximal level of hierarchy with \( n \) input attributes is the cascade hierarchy with \( n - 1 \) subfunctions. The mapping function \( f \) is represented by a new function \( F \) of the intermediate attributes \( z_1, ..., z_\tau \) and/or a cluster \( s_i \) of input attributes. The intermediate attributes can be represented by subfunctions of \( G_1...G_\tau \), fed by lower level of attribute set, which could include intermediate attributes and/or another cluster \( s_j \) of input attributes. Hence, \( y = f(x_1, ..., x_n) = F(z_1, ..., z_\tau, s_i) = F(G_1(S_1),...,G_\tau(S_\tau), s_i) \).

A linguistic decision tree (LDT) can explicitly model both the uncertainty and vagueness in a mapping system, and linguistic rules extracted from the LDT often implies our knowledge of aggregation. When an LAH uses LDTs to represent the functional mappings between parent and child attribute nodes, the semantic information will be transparently propagated through the hierarchy from bottom to top. Assume the goal variable \( y \in \Omega_y \), and a set \( \mathcal{L}_y \) of labels can be used to describe the goal variable. An introduced intermediate attribute \( z \) will represent an approximate of the goal variable \( y \). Namely, \( z \in \Omega_y \), and the label set \( \mathcal{L}_y \) can be used to describe \( z \). Hence, an LAH can present a hierarchical decision making, and it can provide transparent linguistic interpretation, which helps perform semantic attribute deep learning.

For examples, assume five attributes are fed to an LAH. Fig. 1(a) shows a cascade LAH, embedded with 4 LDTs, through which, the decision information is cascaded from the bottom model \( LDT_1 \) to the top model \( LDT_4 \); Fig. 1(b) illustrates a general LAH, embedded with 3 LDTs, through which, the final decision is made by \( LDT_3 \), based on the two intermediate attributes from \( LDT_1 \) and \( LDT_2 \) in the bottom of the
LAH, and the additional attribute $x_5$. The intermediate attributes $z_1$ and $z_2$ represent an approximate of the goal variable $y$, $z_1$ is decided by $x_1$ and $x_2$, and $z_2$ is decided by $x_3$ and $x_4$.

Figure 1: Two LAHs with five input attributes

3.2. A Linguistic decision tree based on Label Semantics

An LDT [38, 39] is a probabilistic tree under the framework of label semantics [40]. Label semantics introduces two fundamental and interrelated measures: Appropriateness measure ($\mu_L(x)$) and mass assignment ($\nu_x$), where, $\mu_L(x)$ quantifies how a label $L$ is appropriate to describe $x$ based on agent’s knowledge of the current labeling conventions, shown in the collected samples or evidences; $\nu_x$ quantifies how a particular subset of labels is appropriate to describe $x$. The particular subset of all and only the labels that are appropriate to describe $x$ is called focal set, denoted as $F$. Given a set of labels $L = \{s, m, l\}$, the focal set can be $F = \{\{s\}, \{s, m\}, \{m\}, \{m, l\}, \{l\}\}$.

An LDT consists of a node set $V$ and an edge set $E$. A node $v \in V$ is associated to an attribute, and an edge $e \in E$ from a node is a focal set, which is appropriate to describe the attribute, indicated by the node. A path from the top node to a leaf of the LDT is called a branch, denoted as $B$. The branch length ($l$) is the number of nodes on the branch $B$. Each branch provides a conjunction of focal sets: $F_1 \land \ldots \land F_l$, with the inference of a set of mass assignments conditional to the branch $B$, denoted as $\nu_y(F|B)$, for each focal set $F \in F_y$, where, $F_y$ is a set of focal sets that are appropriate to describe different values distributed in the domain of the goal variable $y$. The semantics of an LDT is described in Definition 3.1.

Definition 3.1 (Semantics of an LDT). The rule derived from a branch $B_i$ in an LDT is presented as:

$$F_{i_1} \land \ldots \land F_{i_l} \rightarrow F : \nu_y(F|B_i),$$ (1)

where, $F_{i_k}$ is the $k$-th focal set on branch $B_i$, and $F_{i_k} \in F_{x_{i_k}}$. Given attribute values $\vec{x} = (x_1, \ldots, x_n)$. The mass assignment $\nu_y$ can be obtained using Jeffrey’s rule [41].

$$\nu_y(F) = \sum_{i=1}^{\beta} \mu_{B_i}(\vec{x})\nu_y(F|B_i), F \in F_y$$ (2)
where $\beta$ is the branch number of the LDT, and $\nu_y(F|B_i)$ is equivalent to the conditional probability $p(F|B_i)$. $x_{ij}$ denotes the $j$-th attribute on the $i$-th branch. Assume a focal set $F_{ij}$ is appropriate to describe attribute $x_{ij}$ on the branch $B_i$. The appropriateness measure of $\vec{x}$ on branch $B_i$ is the product of all mass assignments on focal sets that are appropriate to describe the corresponding nodes on the branch, as expressed in Formula (3).

$$\mu_{B_i}(\vec{x}) = \prod_{j=1}^{l} \nu_{x_{ij}}(F_{ij}).$$

(3)

4. The semantics of a linguistic attribute hierarchy

When an LAH is represented a hierarchy of LDTs, the information is propagated through the LDTs from information sources (e.g. sensors) in the bottom to the decision variable on the top. Therefore, the rules can be derived as conditional expressions through Formula (1) in the label semantics framework. The output variable (either an intermediate attribute or a goal variable) of each LDT in the LAH can be calculated with Formula (2). This provides an approach to quantifying the degree of our belief how each focal set $F \in F_y$ is appropriate to describe the goal, given partial input attributes and/or the previous results of decision or classification (intermediate attributes) in lower level.

Now, we use the LAH in Fig. (b) as an example to demonstrate the upwards information propagation through an LAH. In Fig. (b), two LDTs ($LDT_1$, $LDT_2$) are located in the bottom of the LAH. $y = f(z_1, z_2, x_5) = f(g_1(x_1, x_2), g_2(x_3, x_4), x_5)$. The mappings $g_1$, $g_2$, and $f$ are represented in the form of linguistic decision trees (i.e. $LDT_1$, $LDT_2$ and $LDT_3$). $LDT_1$ provides the function of quantifying our belief of $z_1$ on goal labels in terms of input attributes $x_1$ and $x_2$. $LDT_2$ presents the function of quantifying our belief of $z_2$ on goal labels in terms of input attributes $x_3$ and $x_4$, and $LDT_3$ offers the function of quantifying our final belief of the goal variable $y$ on its labels in terms of lower level beliefs of $z_1$ and $z_2$, as well as another input attribute $x_5$ from information source (e.g. sensors). Assume the input attributes are clustered to $\kappa$ subsets $s_1, ... s_\kappa$, which can feed the intermediate attributes or/and the goal variable in an LAH. As an intermediate attribute $z_i \in \Omega_y$ is the approximate of the goal variable $y$, it can be estimated by a decision making model (e.g. LDT), fed with $s_i$. Hence, the decision making model can be trained with the samples of $(S_i, y)$. Given the values of all input attributes $\vec{x} = (x_1, ..., x_n)$, the mass assignments of all intermediate attributes and the goal variable can be estimated with Formula (2) through all the LDTs in the LAH. Namely, the decision information is propagated through the bottom LDTs to the top LDT in the LAH. The semantics of an LAH is defined as Definition 4.1.

**Definition 4.1 (Semantics of an LAH).** The semantics of an LAH is the synthetisation of rules extracted from the branches in the LAH, which are allocated by the given sample, $\vec{x} = (x_1, ..., x_n)$. Assume $k$ outputs (i.e. $k$ intermediate attributes) of $LDT(t_1)$ ... $LDT(t_k)$ are the inputs of $LDT(t_i)$, the rule will be $B^{t_1} \land ... \land B^{t_k} \rightarrow B^{t_i}$, where $B^{t_i}$
... $B^{k}$ can be derived in the form of Formula (1), by an LDT, either directly based on information sources (i.e. the input attributes), or based on the intermediate attributes from the lower level of LDTs.

For the instance of Fig. 1(b), given all the input attribute values, $\vec{x} = \{x_1, x_2, x_3, x_4, x_5\}$ in all samples. For generality, we use $\ell$ to denote the label expression, associated to an edge in an LDT. For the special case, the edges of an LDT are associated to a focal set, $\ell$ represents a focal set $F$. Hence, the semantics of each decision tree can be described as:

For $LDT_1(x_1, x_2)$, as it is fed with two attributes, the maximum branch length is 2. Hence, the rule corresponding to the branch $B^1_i$ can be:

$$\ell^1_{i_1} \land \ell^1_{i_2} \rightarrow F : \nu_y(F|B^1_i), F \in F_y.$$  

Similarly, for $LDT_2(x_3, x_4)$, the rule corresponding to the branch $B^2_j$ can be:

$$\ell^2_{j_1} \land \ell^2_{j_2} \rightarrow F : \nu_y(F|B^2_j), F \in F_y.$$  

For $LDT_3(z_1, z_2, x_5)$, the maximum branch length is 3. Hence, the rule corresponding to the branch $B^3_k$ can be:

$$\ell^3_{k_1} \land \ell^3_{k_2} \land \ell^3_{k_3} \rightarrow F : \nu_y(F|B^3_k), F \in F_y.$$  

The synthetic semantics of the LAH is:

$$(\ell^1_{i_1} \land \ell^1_{i_2}) \lor (\ell^2_{j_1} \land \ell^2_{j_2}) \rightarrow (\ell^3_{k_1} \land \ell^3_{k_2} \land \ell^3_{k_3})$$

$$\rightarrow F : \nu_y(F|B^3_k), F \in F_y.$$  

5. Construction of Linguistic Attribute Hierarchy

The basic idea for the construction of a Linguistic Attribute Hierarchy includes two steps: (1) Decomposition of attributes with a distance correlation based clustering algorithm; (2) Self-organisation of a linguistic attribute hierarchy in terms of the distance correlation between clustered attributes and the goal variable.

5.1. Attribute decomposition based on their distance correlation

5.1.1. Distance correlation

Distance correlation can be used to statistically measure the dependence between two random variables or vectors, which could have different dimensions. It is zero if and only if the random variables are statistically independent. Assume $(x_i, y_i), i = 1, 2, ..., n$ be a set of samples. The $n$ by $n$ distance matrices $(a_{i,j})$ and $(b_{i,j})$ present the all pairwise distances for $x$ and $y$ in all the samples, respectively.

$$a_{i,j} = \| x_i - x_j \|; \quad b_{i,j} = \| y_i - y_j \|.$$  

(4)
where $||\cdot||$ denotes Euclidean norm. Then take all doubly centered distances:

$$A_{i,j} = a_{i,j} - \bar{a}_i - \bar{a}_j + \bar{a}_., B_{i,j} = b_{i,j} - \bar{b}_i - \bar{b}_j + \bar{b}_.,$$

(5)

where, $\bar{a}_i$ is the $i$-th row mean, $\bar{a}_j$ is the $j$-th column mean, and $\bar{a}_.$ is the grand mean of the distance matrix for the $x$ samples. The notation is similar for the $b$ values on the $y$ samples. The squared sample distance covariance (a scalar) is simply the arithmetic average of the products $A_{i,j}$ and $B_{i,j}$:

$$dCov^2_n(x, y) = \frac{1}{n^2} \sum_{i=1}^{n} \sum_{j=1}^{n} A_{i,j} B_{i,j}.$$  

(6)

The sample distance variance is the square root of

$$dVar^2_n(x) = dCov^2_n(x, x) = \frac{1}{n^2} \sum_{i,j} A_{i,j}^2,$$

(7)

The distance correlation of two random variables can be calculated through dividing their distance covariance by the product of their distance standard deviations, as Formula (8).

$$dCorr(x, y) = \frac{dCov(X, Y)}{\sqrt{dVar(x) \cdot dVar(y)}}.$$  

(8)

Distance correlation has the properties:

1. $0 \leq dCorr_n(x, y) \leq 1$, $0 \leq dCorr(x, y) \leq 1$;
2. The distance correlation matrix is symmetric (i.e. $dCorr(i, j) = dCorr(j, i)$), and $dCorr(i, i)=1$.
3. $dCorr(x, y) = 0$, if and only if $x$ and $y$ are independent;
4. $dCorr_n(x, y) = 1$ and $dCorr(x, y) = 1$ indicates that the linear subspaces spanned by $x$ and $y$ samples respectively almost surely have an identical dimension [42].

5.1.2. Clustering attributes based on distance correlation

As we do not need to consider the distance correlation between an attribute and itself, we set the diagonal of the distance correlation matrix to zeros for the convenience of computing. First, we find the maximum value in the distance correlation matrix $dCorr$. The maximum value is denoted as $d_{max}$, the column of the maximum value in the matrix is denoted as $i_{max}$. Here we set a value $\alpha$ as the range of distance correlation difference in a cluster. Namely, the distance correlation of attributes in a cluster, correlating to attribute $i_{max}$ will be in $(d_{max} - \alpha, d_{max}]$. A feasible approach to setting the value of $\alpha$ is:

$$(max(dCorr) - min(dCorr))/k,$$  

(9)
Usually, \( \min(dCorr) \) is not a zero, and \( k \) is the preset number of clusters. Secondly, all relevant columns and rows, where the identified cluster members in the cluster are located, are set to zeros. The process is repeated with starting to find next attribute with largest distance correlation in the unvisited attributes for the next cluster, until all elements of \( dCorr \) are zeros. Algorithm 1 provides the pseudo-code of the clustering algorithm, where the produced clusters are saved in \( S \).

**Algorithm 1 Distance Correlation Clustering \((dCorr,k)\)**

1. Initialise(\( S \));
2. \( t=0 \);
3. while \((dCorr \neq [0])\) do
4. \( t = t+1; \)
5. \( \alpha = (\max(dCorr) - \min(dCorr))/k; \)
6. \( [d_{max}, i_{max}] = \max(dCorr); \)
7. \( T \leftarrow i_{max}; \)
8. \( T \leftarrow \text{find}_i(d_{max} > dCorr_{t,i_{max}} \geq d_{max} - \alpha)); \)
9. \( S_t \leftarrow T; \)
10. clearCols(\( dCorr, T; \))
11. clearRows(\( dCorr, T; \))
12. end while

### 5.2. Self-organisation of linguistic attribute hierarchy

Once attributes are clustered based on their distance correlation, we can construct a linguistic attribute hierarchy in terms of the distance correlation between clustered attributes and the goal variable. If an attribute has stronger distance correlation with decision variable, this indicates the attribute has stronger linear correlation to the goal variable, namely, the mapping function between the attribute and the goal could be more linear. In other words, it is easier to make decision in terms of the attribute. Our preliminary experimental results show that attributes, which have stronger distance correlation to the goal variable, should be fed to the LDT in higher layer of the linguistic attribute hierarchy. The average of distance correlation between attributes in a cluster and the goal variable can be calculated with Formula (10):

\[
\overline{dCorr}\left(s\right) = \frac{\sum_{x \in s} dCorr(x, y)}{t}, t = |s|.
\] (10)

where, \( x \) is the attributes in cluster \( s \), \( y \) is the goal variable, and \( t \) is the size of cluster (i.e. number of attributes in cluster \( s \)).

The basic idea of constructing an LAH based on the produced clusters is that:

1. calculating average distance correlation for each produced cluster;
2. Sort all average distance correlations between clusters and decision variable;
3. The LDT fed by the cluster with lower average distance correlation will be at lower level of the LAH;
4. The LDTs fed by the clusters that have similar average values of distance correlation will be at the same level of the LAH. A threshold \( \theta \) is set for the assessment.
of clusters at the same level. The outputs of all LDTs at the same level will be the partial inputs of the next LDT in the next level of LAH;

(5) The LDT fed by the cluster with the highest average distance correlation will be on the top of the LAH.

Assume $d\text{Corr}(s_i)$ represents the average distance correlation of cluster $s_i$ to the goal variable, and $\theta$ represents the maximum difference of distance correlation values of clusters at the same level to the goal variable. Namely, if

$$|d\text{Corr}(s_i) - d\text{Corr}(s_j)| < \theta,$$

then the LDTs fed with $s_i$ and $s_j$ will be at the same level in the LAH.

Algorithm 2 provides the pseudo-code of the self-organisation of LAH, where $d\text{Corr}_{XY}$ is the vector of distance correlation of all attributes to the goal variable, $S$ is the set of clusters produced by the distance correlation-based clustering algorithm above, $R$ is the set of all the average distance correlations for cluster set $S$ with $K$ clusters to the goal variable, $Z$ is the index of an intermediate attribute, which starts from $n+1$, the input attribute number, $I$ is used to save the indices of intermediate attributes and the corresponding clusters at the same level, $H$ is used to save the hierarchy to be constructed. For all clusters, if $R_i - R_a < \theta$, save the intermediate attribute index $Z$ to $I$, and append $Z$ and the cluster $S_t_i$ to the hierarchy $H$, else, start a new level, append current $Z$, other intermediate attributes saved in $I$ and corresponding cluster $S_t_i$ to the hierarchy $H$, and save current $Z$ to $I$. If more than one LDTs at the same level in the LAH, and there is no further cluster to be constructed, then the outputs of those LDTs will be the inputs of the top LDT in the LAH. If the LDT constructed with the last cluster does not share the same level with other LDTs, then it will be the top LDT of LAH.

### 5.3. Training of a linguistic attribute hierarchy

In 2009, He and Lawry [7, 8] first time proposed using the domain and labels of the goal variable to describe intermediate attributes. This made the hierarchy training became possible, and the hierarchical decision making became meaningful. In [9], He and Lawry developed a non-recursively post-order traversal algorithm to train a given LAH, where all LDTs are trained with LID3 in a bottom-up way. Intermediate attribute values are estimated with the trained LDTs, and then they are input to next level LDTs. For example, in Fig. 1(b), $LDT_1$, $LDT_2$ and $LDT_3$ are trained in turn, where, the intermediate variables $z_1$ and $z_2$ use the corresponding $y$ values in all training points, as the goal values of $LDT_1$ and $LDT_2$ respectively. After $LDT_1$ and $LDT_2$ are trained, the intermediate variables $z_1$ and $z_2$ for the training samples can be estimated by $LDT_1$ and $LDT_2$ respectively. The estimated values of $z_1$ and $z_2$, the values of $x_5$, and the values of the goal variable $y$ in the training samples are used to train the $LDT_3$, and finally the goal variable $y$ can be estimated by $LDT_3$. Here, we introduce a recursive postorder implementation of the LAH training algorithm (see Algorithm 3).

$\mathcal{H}$ is a global variable, indicating the LAH to be trained. Initially, $v$ is the root of the LAH to be trained, and all leaves in the LAH are tagged as visited, as they are the input attributes from sensor nodes. $v.ch$ represents the child set of node $v$, indicating
Algorithm 2 SOLAH\((d\text{Corr}\,XY, S, \theta)\)

1: \( [\mathcal{R}] = \text{avDistCorr}(d\text{Corr}\,XY, S); \)
2: \( Z = n; K = |S|; \)
3: \( [R, t] = \text{sort}(\mathcal{R}); \) %increasingly
4: \( \mathcal{I} = \phi, \mathcal{H} = \phi; \)
5: \( i = 0, a = 0; \)
6: \( H_v = \phi; \)
7: \( \text{while } (i < K) \) do
8: \( i = i+1; Z = Z+1, \)
9: \( \text{if } ((R_i - R_a < \theta) \) then
10: \( \mathcal{I} = (Z); \)
11: \( \mathcal{H} = (Z, S_t); \)
12: \( \text{else } \)
13: \( a = i; \) % starting a new level
14: \( \mathcal{H} = (Z, \mathcal{I}, S_t); \)
15: \( \mathcal{I} = Z; \)
16: \( \text{end if } \)
17: \( \text{end while } \)
18: \( \text{if } (|\mathcal{I}| > 1) \) then
19: \( H \leftarrow (Z + 1, \mathcal{I}); \)
20: \( Y = Z+1; \)
21: \( \text{else } \)
22: \( Y = Z; \)
23: \( \text{end if } \)

5.4. The LID3 algorithm for LDT training

LID3 was used to train an LDT with a given database \([38, 7]\). It is an update of the classic ID3 algorithm \([43]\), through combining label semantics. The training process is conducted through selecting the attribute that obtains the maximum information gain to extend a branch. The functions used in LID3 have been formulated in \([38, 7]\), such as information entropy of a branch \((E)\), expected entropy \((EE)\) when an attribute node \(x\) is added to a branch, and information gain \((IG)\). Here, a recursive algorithm implementation of LID3 is proposed (Algorithm 4).

A threshold \(\vartheta\) is setup to stop the branch extension when the conditional probability \(P(C|B)\) reaches the threshold \(\vartheta\). The maximum of branch length is the number of
attributes that feed the LDT. Initially, \( T = \phi \) (empty), hence, current node point \( B_v = \phi \) as well. For each branch, the conditional probabilities for all classes \( C \in \mathcal{C} \) are calculated. The most informative attribute is selected as the next node to be extended to the existing LDT, and all focal sets that are appropriated to describe the attribute in the domain of \( \Omega_x \) will be appended to the tree \( T \). The process is continued until the maximum conditional probability arrives the specified threshold.

**Algorithm 4** \( LID3(B_v, S) \)

\[
\text{for (all } C_i \in \mathcal{C}_y \text{) do}
\]
\[
P(C_i | B_v) = \text{conditionProb}(B_v);
\]
\[
\text{end for}
\]
\[
\text{if } (\max \{P\} \ge \emptyset) \text{ or } S = \phi \text{ then}
\]
\[
\text{return;}
\]
\[
\text{end if}
\]
\[
\text{for (all } x \in S \text{) do}
\]
\[
IG(B_v, x) = E(B_v) - EE(B_v, x);
\]
\[
\text{end for}
\]
\[
\hat{x} = \arg\max_{x \in S} \{IG\};
\]
\[
B_v = B_v + x
\]
\[
T \leftarrow F \in \mathcal{F}_x;
\]
\[
S = S \setminus \hat{x};
\]
\[
\text{for (all } F \in \mathcal{F}_x \text{) do}
\]
\[
LID3(B_v, S);
\]
\[
\text{end for}
\]

### 6. Experiments and Evaluation

#### 6.1. Experiment methodologies

All databases, used for the experiments, are from UCI machine learning repository [14]. The experiments are conducted in two stages:

1. A case study will be done on the benchmark database of Message Spams. This experiment is to demonstrate the use of developed approach for self-organisation of a linguistic attributes hierarchy, given the data.

2. A set of databases will be tested with the SOLAHs. The performance are used to validate the developed approach to self-organising LAHs by comparing with the performance obtained with the single LDT.

**Experimental environment:** The experiments are carried out on a laptop with 64-bit Windows 10 and x64-based processor with Intel (R) Core (TM) i5-4210U CPU @1.7GHZ, 2.4GHZ, 8GB memory. The LAH training algorithm is implemented in C++.

**Discretisation:** For simplicity, all attributes are expressed with three labels, except binary variables, decision or goal variables, which are expressed with the labels as they have. All neighbouring fuzzy intervals are overlapping with 50%.

**Mass assignments of attributes:** In terms of Label Semantics, if \( n \) labels are used to describe a continuous variable \( x \), then there will be \( 2n - 1 \) focal sets that are appropriate to describe \( x \in \Omega_x \). For discrete variables, the mass assignment on a focal set that contains only one label is 1, but the mass assignment on a focal set that contains two successive labels is 0.
Ten-fold cross validation: 90% of data is used for training, 10% is used for test. Therefore, data is partitioned to ten parts equally.

Performance measure: The performance is measured with:

1. Accuracy: \( A = \frac{TP + TN}{N} \), where, \( TP \) is the number of true positive estimates and \( TN \) is the number of true negative estimates, and \( N \) is the number of samples in the database.

2. Area under ROC curve (AUR): A ROC curve is used to measure how well the classifier separates the two classes without reference to a goal threshold. The area under the ROC curve has been formalized in [9].

3. Rule number \( \beta \): As defined in [9], the rule number of an LAH is the sum of branch numbers, extracted from all LDTs in the LAH. Since a probability threshold was introduced during the process of LDT training, a branch training may stop earlier. Hence, the actual branch number in an LDT may be much less than a full LDT.

4. The running time \( t \): it is the time, spending on the process of ten-fold crossing validation, including training, testing, and overhead for data splitting and exchanging.

6.2. A case study on a benchmark database

To demonstrate the developed approach, the first experiment is carried out on the benchmark database, 'SMSSpamCollection'. The performance for the LAH that was constructed by the proposed approach, the LAH in [13], obtained manually, and the single LDT, is evaluated and compared.

6.2.1. The database

The SMSSpamCollection database [44], has 5574 raw messages, including 747 spams. The two sets of features, extracted for the research [45, 13], are used for the experiments. Table 1 shows the set of 20 features, and Table 2 presents the set of 14 features.

| \( x \) | key word | \( x \) | key-word |
|-------|---------|-------|---------|
| 0     | urgent  | 10    | stop    |
| 1     | congrat | 11    | click   |
| 2     | !       | 12    | Text,Txt|
| 3     | WIN/WON | 13    | sex     |
| 4     | Offer   | 14    | dir     |
| 5     | Award   | 15    | cash    |
| 6     | Prize   | 16    | free    |
| 7     | Call    | 17    | 0p, 1p, ..., 9p |
| 8     | Reply   | 18    | EURO, GBP, pound, $, € |
| 9     | Send    | 19    | price   |

6.2.2. Experimental results on SMS-20

The six clusters obtained by the proposed distance correlation-based clustering algorithm are: \( S_1: \{x_7, x_1, x_4, x_{18}\}, S_2: \{x_0, x_{19}\}, S_3: \{x_{15}, x_{16}\}, S_4: \{x_6, x_2, x_3, x_8\}, S_5: \{x_9, x_{11}\}, S_6: \{x_5, x_{10}, x_{12}, x_{13}, x_{14}, x_{17}\} \).
Table 2: The set of 14 features [13]

| x | key word       | x | key word      |
|---|---------------|---|---------------|
| 0 | urgent        | 7 | call          |
| 1 | congrat       | 8 | Reply/send    |
| 2 | !             | 9 | stop/click    |
| 3 | WIN/WON       | 10| text/txt      |
| 4 | Offer         | 11| cash/price    |
| 5 | Award/Prize   | 12| free/0p...9p  |
| 6 | sex/girl      | 13| EURO, GBP, pound, $, € |

\[ S_6 \] has the lowest average distance correlation to the decision variable. Hence, the LDT fed by \[ S_6 \] should be at the bottom of the SOLAH, and produces an intermediate attribute \( z_{20} \). Here \( z \) is used to represent an intermediate attribute. \( z_{20} \) with set \( S_4 \) feeds an LDT in the second level according to the average distance correlation of \( S_4 \) to decision variable. Sets \( S_5 \) and \( S_4 \) have similar average distance correlations to decisional variable. Hence, they sit at the same level in the SOLAH, and their outputs \( z_{21} \) and \( z_{22} \) feed the third level of LDT constructed with set \( S_1 \). Sets \( S_2 \) and \( S_3 \) have similar average distance correlations to the decision variable. Hence they are at the same level. As they are at the top level, \( S_2 \), \( S_3 \) and the output \( z_{23} \) of the LDT at the third level will be an input of the top LDT. Figs. 2 and 3 illustrate the SOLAH and the best \( LAH_m \), manually produced in [13]. The two LAHs have the same levels and the same number of LDTs. But the compositions are different. Table 3 shows the

![Figure 2: The SOLAH on SMS-20](image)

![Figure 3: A LAH_m on SMS-20](image)

performance for the SOLAH, the \( LAH_m \) in [13] and the single LDT fed by the whole feature vector (denoted as \( LDT(\vec{x}) \)). It can be seen that the accuracy \( A \) of SOLAH is very close to the performance of \( LAH_m \) and \( LDT(\vec{x}) \), and the performance of the
AUC is slightly smaller than that of $LAH_m$, but better than that of $LDT(\vec{x})$. However, $SOLAH$ has the smallest branch number $\beta$ and running time $T$, which is the time of ten-fold crossing validation on a solution, measured in milliseconds (ms).

Table 3: The performance of solutions

| LAHs         | $A$    | $AUC$   | $\beta$ | $T$(ms) |
|--------------|--------|---------|---------|---------|
| $SOLAH$, Fig. 2 | 0.936491 | 0.946402 | 197     | 101766  |
| $L AH_m$ [13], Fig. 3 | 0.948511 | 0.949233 | 267     | 143718  |
| $L DT(\vec{x})$ | 0.956225 | 0.895815 | 775     | 1339234 |

6.2.3. Experimental results on SMS-14

For this experiment, the cluster number is preset to 4. But the clusters obtained by the DCC algorithm are: $S_1:\{x_6,x_1\}$, $S_2:\{x_9,x_4,x_8,x_{10}\}$, $S_3:\{x_5,x_0,x_1,x_2,x_3,x_{12},x_{13}\}$. Set $S_1$ has the lowest average distance correlation to the decision variable, thus the LDT fed by $S_1$ is placed at the bottom of the SOLAH. The LDT fed by set $S_1$ and the output of the first level at the second level of the SOLAH, and the LDT fed by set $S_3$ sits at the top level of the LAH.

Figs. 4 and 5 illustrate the SOLAH and the $L AH_m$ with the best $AUC$, which was obtained by the attribute composition 3 in [13]. The two LAHs have the same levels, and the attributes are decomposed to 3 clusters. But the compositions of LAHs are different. In the $L AH_m$, all LDTs fed by the clusters of input attributes are placed at the bottom of the LAH. Hence a new LDT, fed by all intermediate attributes, which were produced by the bottom level of LDTs, is added on the top of the LAH. $SOLAH$ has higher level than $L AH_m$, and all LDTs fed by the attribute clusters are cascaded.

Figure 4: The SOLAH on SMS-14

Figure 5: A LAH on SMS-14, obtained manually in [13]

Table 4 shows the performance for the SOLAH, the $L AH_m$, constructed with composition 3 manually in [13] and the single LDT $(L DT(\vec{x}))$. All solutions in Table 4
obtained similar performance in accuracy and the area under ROC. All performance values of SOLAH are in between LAH_m and LDT(\bar{x}).

| LAHs                  | A          | AUC       | \(\beta\) | \(T\) (ms) |
|-----------------------|------------|-----------|------------|------------|
| SOLAH, Fig. 2         | 0.955328   | 0.943391  | 339        | 101766     |
| LAH_\(m\)[13], Fig. 3 | 0.951202   | 0.957466  | 130        | 79719      |
| LDT(\bar{x})          | 0.961607   | 0.924536  | 583        | 613704     |

6.2.4. Impact of the preset cluster number

Similar to classic \(k\)-means clustering algorithm, the preset cluster number \(k\) is important for the construction of LAHs, and thus has important impact on the performance of decision making or classification. Here, the relationships between \(k\) and structures of LAHs and between \(k\) and performance are observed. It should be noticed that the real cluster number may be different to the preset cluster number, as the preset cluster number decides the range (\(\alpha\)) of distance correlation of a cluster \(C\), which is formed with the attributes that have the distance correlations in the range \(\alpha\) to an attribute \(x\), which has the largest distance correlation in the distance correlation matrix. The rest attributes could have larger distance correlations than the members except attribute \(x\) in cluster \(C\). Table 5 shows the solutions for \(k=2\) to 10. It can be seen that the real cluster numbers are different to the preset cluster number \(k\) in some cases. The cluster numbers varies from 3 to 6. All SOLAHs have similar performance. The SOLAH constructed with 3 clusters obtained the highest accuracy, but it has more branches. The SOLAHs constructed with 4 clusters have the highest AUC values. The clusters obtained when \(k=7\) is the same as that when \(k=8\), which is happened when \(k=9\), 10 as well. In Table 5 \(\beta\) denotes the branch number, \(\iota\) denotes the level of the hierarchy.

| \(k\) | \(k\) | A          | AUC       | \(\beta\) | \(T\) (ms) |
|-------|-------|------------|-----------|------------|------------|
| 2     | 3     | 0.956      | 0.946     | 339        | 4          | 221547     |
| 3     | 4     | 0.937      | 0.953     | 184        | 4          | 101875     |
| 4     | 4     | 0.936      | 0.950     | 154        | 4          | 84093      |
| 5     | 6     | 0.932      | 0.949     | 159        | 5          | 91375      |
| 6     | 6     | 0.932      | 0.948     | 154        | 6          | 89828      |
| 7.8   | 6     | 0.932      | 0.945     | 108        | 6          | 63313      |
| 9, 10 | 6     | 0.932      | 0.941     | 130        | 5          | 76859      |

Similarly, Table 6 lists the performance of SOLAHs constructed with various \(k\) values from 2 to 10. The AUC values of all SOLAHs are very close. When the preset \(k=2\), the SOLAH was constructed with two clusters. It obtained the highest accuracy, but lowest AUC value and the largest branch number. When the preset \(k=9, 10\), the same cluster sets were produced, with which, the SOLAH was produced. It has the lowest accuracy, but the smallest branch number. For all preset \(k=3...6\), different sets of 5 clusters were produced, with which, four different SOLAHs were produced. They have the same level, and their performance values (\(A\) and AUC) are very close. When
the preset $k=5,6,7$, the produced SOLAHs have the same performance values ($A$ and $AUC$). But they have different branch numbers.

### Table 6: The properties of LAHs on SMS20

| $k$ | $K$ | $A$  | $AUC$ | $\beta$ | $\iota$ | $T$(ms) |
|-----|-----|------|-------|---------|--------|---------|
| 2   | 2   | 0.957| 0.933 | 540     | 3      | 467328  |
| 3   | 5   | 0.934| 0.950 | 189     | 5      | 111016  |
| 4   | 5   | 0.949| 0.941 | 203     | 5      | 114265  |
| 5   | 5   | 0.936| 0.946 | 187     | 5      | 101594  |
| 6   | 5   | 0.936| 0.946 | 197     | 5      | 107766  |
| 7   | 6   | 0.936| 0.946 | 204     | 5      | 115750  |
| 8   | 7   | 0.929| 0.940 | 163     | 6      | 93891   |
| 9,10| 7   | 0.891| 0.944 | 157     | 7      | 104938  |

From the data on both SMS20 and SMS14, it can be seen that a suitable cluster size $K$ will have a good trade-off on accuracy performance and time complexity (branch number).

### 6.3. Validation on some benchmark databases

#### 6.3.1. The 12 benchmark databases

The experiments on 12 benchmark databases from UCI machine learning repository are conducted for validating the proposed approach of constructing LAHs. Table 7 provides the basic properties of the 12 data sets, including, database name, attribute number ($n$), goal state number ($N_g$), total sample number ($N$), class distribution ($N_c$).

### Table 7: Database properties

| Datasets                  | $n$ | $N_g$ | $N$  | $N_c$ |
|---------------------------|-----|-------|------|-------|
| Beast Cancer (BC)         | 9   | 2     | 286  | 201,85|
| Wisconsin BC              | 9   | 2     | 569  | 357,212|
| Ecoli                     | 7   | 8     | 336  | 143,77,52,35,20,5,2,2 |
| Glass                     | 9   | 6     | 214  | 70,17,76,13,9,29 |
| Heart-C                   | 13  | 2     | 303  | 165,138|
| Heart-statlog             | 13  | 2     | 270  | 150,120|
| Hepatitis                 | 19  | 2     | 155  | 70,85 |
| Wine                      | 13  | 3     | 178  | 59,71,48|
| Liver Disorders           | 6   | 2     | 346  | 146,200|
| Diabetes                  | 8   | 2     | 768  | 500,268|
| Ionosphere                | 33  | 2     | 351  | 231,120|
| Sonar                     | 60  | 2     | 208  | 97,111|

#### 6.3.2. Comparisons with LDT

Table 8 shows the performance of SOLAHs and corresponding single LDTs on the 12 benchmark data sets. It can be seen that the SOLAH achieved dominated better performance in accuracy $A$ and $AUC$ than the single LDT for most databases. For
Table 8: Performance comparison between constructed LAHs and the single LDTs for different databases, the unit of time (T) is second

| Databases   | SOLAH  | LDT    |
|-------------|--------|--------|
|             | AUC    | β      | T   | AUC    | β      | T   |
| BreastC.    | 0.78   | 0.84   | 985 | 9     | 0.90   | 0.64 | 1201 |
| WBC         | 0.97   | 0.98   | 1100| 36    | 0.95   | 0.96 | 1029 |
| Ecoli       | 0.85   | 0.93   | 1103| 15    | 3      | 0.85 | 0.89 | 3337 |
| Glass       | 0.81   | 0.91   | 5124| 48    | 2      | 0.68 | 0.83 | 9433 |
| Heart-c     | 0.81   | 0.81   | 1452| 24    | 5      | 0.76 | 0.78 | 1345 |
| Heart-s.    | 0.79   | 0.83   | 972 | 11    | 5      | 0.77 | 0.80 | 1053 |
| Hepatitis   | 0.92   | 0.94   | 4278| 47    | 6      | 0.79 | 0.82 | 1677 |
| Wine        | 1.00   | 0.99   | 929 | 6     | 5      | 0.94 | 0.98 | 1309 |
| Liver       | 0.56   | 0.60   | 226 | 3     | 3      | 2    | 0.57 | 0.56 | 1585 |
| Diabetes    | 0.76   | 0.82   | 1018| 32    | 3      | 2    | 0.74 | 0.79 | 1486 |
| Ionosphere  | 0.87   | 0.71   | 4393| 14    | 5      | 0    | 0.87 | 0.87 | 1214 |
| Sonar       | 0.73   | 0.80   | 26328| 52    | 6      | 0.67 | 0.73 | 9253 |

In order to observe the performance improvement of the LAH that is trained by 90% of data when ten-fold crossing validation is applied, Table 9 particularly lists the performance of the SOLAHs with ten-fold crossing validation on the two high
dimensional databases. The performance in both accuracy and the area under ROC is improved, especially for the database, Ionosphere, the performance in accuracy and the area under ROC are improved very much, compared to the SOLAH with two-fold crossing validation. Moreover, the running times of the SOLAHs for the two databases are 143s and 507s, respectively, which are much shorter than that of the single LDT with two-fold crossing validation in Table 8.

| Databases | A     | AUC | \(T_{(ms)}\) | \(\epsilon\) | \(K\) |
|-----------|-------|-----|-------------|-------------|------|
| Ionosphere| 0.95  | 0.98| 7717        | 5           | 9    |
| Sonar     | 0.78  | 0.84| 33820       | 6           | 22   |

### 6.3.3. Comparisons with other machine learning algorithms

The average accuracy and standard deviations of SOLAH for ten runs on the 12 data sets from UCI machine learning repository are further compared with the optimal LAH, obtained by a Genetic Algorithm Wrapper in [9], as well as three well-known machine learning algorithms, such as C4.5, Naive Bayes (NB) and Neural Networks (NN). The performance of these algorithms WAS evaluated with WEKA [49] by Qin and Lawry in [38], where, WEKA [49] was used to generate the results of J48 (C4.5 in WEKA) unpruned tree, Naive Bayes and Neural Networks with default parameter settings for ten runs of 50%-50% splitting training and test data.

Table 10 shows the performance of SOLAHs and the existing results in literature for the 12 data sets. SOLAH wins 4 data sets, which is next to OptLAH, which wins 5 data sets. SOLAH obtains comparable performance in accuracy, compared to other algorithms, and even achieves better performance than the LAHs, optimised by GAW, for some tested data sets.

| Databases | C4.5 | N.B. | N.N. | OptLAH | SOLAH |
|-----------|------|------|------|--------|-------|
| BreastC   | 69.16± 4.14 | 71.26± 2.96 | 66.50± 3.48 | 71.77± 2.06 | 75.17± 4.30 |
| WBC       | 94.38± 1.42 | 96.28± 0.73 | 94.96± 0.80 | 96.67± 0.20 | 94.85± 0.25 |
| Ecoli     | 78.99± 2.23 | 85.36± 2.42 | 82.62± 3.18 | 84.02± 0.92 | 86.11± 0.91 |
| Glass     | 64.77± 5.10 | 45.99± 7.00 | 64.30± 3.38 | 71.31± 1.75 | 71.28± 8.08 |
| Heart-c   | 75.50± 3.79 | 84.24± 2.09 | 79.93± 3.99 | 82.81± 4.25 | 78.38± 3.50 |
| Heart-s   | 75.78± 3.16 | 84.00± 1.68 | 78.89± 3.05 | 84.85± 2.31 | 78.15± 1.05 |
| Hepatitis | 76.75± 4.68 | 83.25± 3.99 | 81.69± 2.48 | 94.84± 1.01 | 85.15± 9.12 |
| Ionosphere| 89.60± 2.13 | 82.97± 2.51 | 87.77± 2.88 | 89.80± 1.63 | 92.12± 4.03 |
| Liver     | 65.23± 3.86 | 55.41± 5.39 | 66.74± 4.89 | 58.46± 0.76 | 56.16± 0.90 |
| Diabetics | 72.16± 2.80 | 75.05± 2.37 | 74.64± 1.41 | 76.07± 1.33 | 75.09± 1.39 |
| Sonar     | 70.48± 0.00 | 70.19± 0.00 | 81.05± 0.00 | 74.81± 4.81 | 75.72± 3.74 |
| Wine      | 88.09± 4.14 | 96.29± 2.12 | 96.85± 1.57 | 97.58± 0.27 | 98.17± 2.95 |

### 7. Conclusions

In this paper, the contribution of the innovative research are summarised as below:
(1) We proposed heuristic algorithm to construct a linguistic attribute hierarchy for semantic attribute deep learning in decision making or classification. The self-constructed linguistic attribute hierarchy provides a new form of deep learning, in contrast to conventional deep learning.

(2) The proposed algorithm for the self-organisation of an LAH is much more efficient than meta-heuristic algorithm, and the self-organised linguistic attribute hierarchy can obtain the fusion performance better than or comparable to the single linguistic decision tree, fed with the full set of attributes.

(3) The most important is that the heuristical self-organisation of such linguistic attribute hierarchy can effectively solve the ‘curse of dimensionality’ in machine learning, which is critical challenge in the implementation of IoT intelligence. Hence, the research results will promote a wider of applications of the linguistic attribute hierarchy in big data analysis and IoT intelligence.

(4) A linguistic attribute hierarchy, embedded with linguistic decision trees, will provide a transparent hierarchical decision making or classification. Hence, it could help us to look insight of the decision making process for different purposes (e.g. the effect of adversary samples in decision making).

(5) Comparing with other machine learning in literature, the self-organised LAH obtains comparable performance on the tested data sets, and even achieved better performance than the optimal LAHs, obtained by GAW for some tested data sets.

We will implement the LAH on an embedded system (e.g. a Raspberry Pi system) for a specific task, and will further improve the algorithms and develop new algorithms to construct high efficient and effective linguistic attribute hierarchy, embedded with other machine learning models for decision making or classification in future.

References

[1] Microchip datasheet: dsPIC33EPXXXGM3XX/6XX/7XX. Accessed 24 May 2020. https://ww1.microchip.com/downloads/en/DeviceDoc/70000689d.pdf.

[2] A. L. Blum and R. L. Rivest, Training a 3-Node Neural Network is NP-Complete. Neural Networks, 5, 1992, pp. 117-127.

[3] G.U. Raju, J. Zhou, and R. A. Kiner. Hierarchical fuzzy control. Int. J. Control, Vol. 54:55, pp. 1201-1216, 1991.

[4] Campello RJGB and Amaral WC (2006) Hierarchical Fuzzy Relational Models: Linguistic Interpretation and Universal Approximation, IEEE Transactions on Fuzzy Systems, 14(3):446-453.

[5] Pedrycs W (1993) Fuzzy Control and Fuzzy Systems, 2nd Ed, Research Studies Press, Ltd, Somerset, UK.

[6] Lawry J and He H (2008) Multi-Attribute Decision Making Based on Label Semantics, the International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems, 16(2) supp, pp. 69 - 86.
[7] He H and Lawry J (2009) Optimal Cascade Hierarchies of Linguistic Decision Trees for Decision Making, in Proc. of IAENG International Conference on Artificial Intelligence and Applications (ICAIA'09), Hong Kong.

[8] He H and Lawry J (2009) Optimal Cascade Linguistic Attribute Hierarchies for Information Propagation, IAENG International Journal of Computer Science, 36(2):129-136.

[9] [8] H. He and J. Lawry, Linguistic Attribute Hierarchy and Its Optimisation for Classification Problems, Soft Computing, 18(10), pp 1967-1984, Oct. 2014. DOI: 10.1007/s00500-013-1179-3.

[10] H. He, Z. Zhu and E. Makinen, Task-oriented Distributed Data Fusion in Autonomous Wireless Sensor Networks, Soft Computing, 19(8), 2015, pp. 2305–2319, DOI 10.1007/s00500-014-1421-7.

[11] S. Mitra, K.M. Konwar and S.K. Pal, Fuzzy decision tree, linguistic rules and fuzzy knowledge-based network: Generation and evaluation, IEEE Transactions on Systems, Man, and Cybernetics, Part C: Applications and Reviews, 32(4), 2002, pp. 328-339.

[12] H. He, Z. Zhu, A. Tiwari and Andrew Mills, A Cascade of Linguistic CMAC Neural Networks for Decision Making, in Proceedings of the International Joint Conference on Neural Networks (IJCNN2015), Killarney, Ireland, 12-17 July, 2015.

[13] H. He, T. Watson, C. Maple, J. Mehnens, A. Tiwari, Semantic Attribute Deep Learning with A Hierarchy of Linguistic Decision Trees for Spam Detection, IJCNN2017, Anchorage, Alaska, USA, 14-19 May 2017.

[14] Lichman, M. (2013). UCI Machine Learning Repository [http://archive.ics.uci.edu/ml]. Irvine, CA: University of California, School of Information and Computer Science.

[15] Y. Miao, M. Gowayyed and F. Metze, EESEN: End-to-End Speech Recognition using Deep RNN Models and WFST-based Decoding, CoRR, abs/1507.08240, 2015, http://arxiv.org/abs/1507.08240, arXiv:1507.08240.

[16] Z. Zhang, J. Geiger, J. Pohjalainen, A. E.-D. Mousa, W. Jin, and B. Schuller, Deep Learning for Environmentally Robust Speech Recognition: An Overview of Recent Developments, arXiv:1701.02720v1 [cs.CL], Jan. 2017.

[17] K. He, X. Zhang, S. Ren and J. Sun, Deep Residual Learning for Image Recognition, The IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2016, pp. 770-778.

[18] R. Girshick, J. Donahue, T. Darrell and J. Malik, Rich feature hierarchies for accurate object detection and semantic segmentation, The IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2014, pp. 580-587.
[19] S. Ren, K. He, R. Girshick and J. Sun, Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks, Journal IEEE Transactions on Pattern Analysis and Machine Intelligence, 39(6), June 2017, pp. 1137-1149.

[20] E. Gawehn, J. A. Hiss, G. Schneider, Deep Learning in Drug Discovery, Molecular Informatics, 35(1), 2016, pp. 3-14. WILEY-VCH Verlag, doi:10.1002/minf.201501008.

[21] Y. Park and M. Kellis, Deep learning for regulatory genomics, Nature Biotechnology 33, Aug. 2015, 825–826. doi:10.1038/nbt.3313.

[22] L. Chen, G. Papandreou, I. Kokkinos, K. Murphy and A. L. Yuille, DeepLab: Semantic Image Segmentation with Deep Convolutional Nets, Atrous Convolution, and Fully Connected CRFs, CoRR, abs/1606.00915, 2016, http://arxiv.org/abs/1606.00915, arXiv: 1606.00915.

[23] A. Karpathy, G. Toderici, S. Shetty, T. Leung, R. Sukthankar, F. F. Li, Large-scale Video Classification with Convolutional Neural Networks, IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Columbus, OH, USA, 23-28 Jun. 2014. DOI: 10.1109/CVPR.2014.223

[24] W. Zhang, M. Zhai, Z. Huang, C. Liu, W. Li, and Y. Towards End-to-End Speech Recognition with Deep Multipath Convolutional Neural Networks, In: Yu H., Liu J., Liu L., Ju Z., Liu Y., Zhou D. (eds) Intelligent Robotics and Applications. ICIRA 2019. Lecture Notes in Computer Science, vol 11745. Springer, Cham. DoI:10.1007/978-3-030-27529-7_29.

[25] H. Lee, Y. Largman, P. Pham and A. Y. Ng, Unsupervised feature learning for audio classification using convolutional deep belief networks, in Proc. of the 22nd International Conference on Neural Information Processing Systems (NIPS’09), Vancouver, British Columbia, Canada, 07 - 10 Dec. 2009, pp. 1096-1104.

[26] Y. LeCun, Y. Bengio and G. Hinton, Deep learning, Nature 521, 2015, pp. 436 - 444.

[27] Y. Taigman, M. Yang, M. Ranzato and L. Wolf, Deepface: closing the gap to human-level performance in face verification. In Proc. Conference on Computer Vision and Pattern Recognition, 2014, pp. 1701–1708.

[28] P. N. Druzhkov and V. D. Kustikova, A survey of deep learning methods and software tools for image classification and object detection, Pattern Recognition and Image Analysis, 26(1), pp. Jan. 2016, pp. 9–15

[29] C. Szegedy, A. Toshev, and D. Erhan, Deep neural networks for object detection, in Proc. of NIPS (Lake Tahoe, 2013), pp. 2553–2561.

[30] K. Simonyan, A. Vedaldi, and A. Zisserman, “Deep fisher networks for large-scale image classification,” in Proc. of NIPS (Lake Tahoe, 2013), pp. 163–171.
[31] A. Krizhevsky, I. Sutskever, and G. E. Hinton, “ImageNet classification with deep convolutional neural networks,” in Proc. of NIPS (Lake Tahoe, 2012), pp. 1097–1105.

[32] Y. He, K. Kavukcuoglu, Y. Wang, A. Szlam, and Y. Qi, “Unsupervised feature learning by deep sparse coding,” in Proc. of SIAM International Conference on Data Mining (Philadelphia, 2014), pp. 902–910.

[33] G. E. Dahl, Deep learning approaches to problems in speech recognition, computational chemistry, and natural language text processing, PhD thesis, University of Toronto, 2015.

[34] G. E. Dahl, T. N. Sainath, and Geoffrey E. Hinton. Improving deep neural networks for LVCSR using rectified linear units and dropout. In ICASSP, IEEE, 2013, pp. 8609–8613.

[35] [34] D. Yu and L. Deng, Automatic Speech Recognition – A Deep Learning Approach, 2015, Springer London Heidelberg New York Dordrecht. DOI 10.1007/978-1-4471-5779-3.

[36] R. Socher, A. Perelygin, J. Y. Wu, J. Chuang, C. D. Manning, A. Y. Ng and C. Potts, Recursive Deep Models for Semantic Compositionality Over a Sentiment Treebank, in Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing, Seattle, Washington, USA, October, 2013, pp. 1631-1642.

[37] Z. Qin, and J. Lawry, “Decision tree learning with fuzzy labels”, Information Sciences, Vol. 172, pp. 91-129, 2005.

[38] J. Lawry, “Appropriateness Measures: An Uncertainty Measure for Vague Concepts”, Synthese, Vol. 161, No. 2, pp. 255-269, March, 2008.

[39] Lawry J (2006) Modeling and Reasoning with Vague Concepts, Kacprzyk, J. (Ed.), Springer, New York.

[40] R. C. Jeffrey, The Logic of Decision, The University of Chicago Press, New York. 1990.

[41] G. J. Székely and M. L. Rizzo, Brownian Distance Covariance, The Annals of Applied Statistics, 3(4), 2009, pp. 1236–1265. DOI:10.1214/09-AOAS312.

[42] Quinlan JR (1986) Induction of Decision Trees, Machine Learning, 1:81-106. Kluwer Academic Publishers, Boston.
7.1. 14 attribute-based LAHs for $k = 2$ to 10 (Figs. 6-12)

Figure 6: LAH based on SMS14 with $k = 2$

Figure 7: LAH based on SMS14 with $k = 3$

7.2. LAHs based on 20 attributes for $k = 1$ to 10 (Figs. 13-20)
Figure 8: LAH based on SMS14 with $k = 4$

Figure 9: LAH based on SMS14 with $k = 5$

Figure 10: LAH based on SMS14 with $k = 6$

Figure 11: LAH based on SMS14 with $k = 7$ and $k = 8$. 

27
Figure 12: LAH based on SMS14 with $k = 9$ and $k=10$.

Figure 13: LAH for $k=2$.

Figure 14: LAH for $k=3$.

Figure 15: LAH for $k=4$.

Figure 16: LAH for $k=5$. 
Figure 17: LAH for $k=6$.

Figure 18: LAH for $k=7$.

Figure 19: LAH for $k=8$.

Figure 20: LAH for $k=9$ and $k=10$.