DistAI: An Inter-pattern Distance-based Constructive Learning Algorithm

Jihoon Yang, Rajesh Parekh and Vasant Honavar
AI Research Group, Department of Computer Science
226 Atanasoff Hall, Iowa State University, Ames, IA 50011. U.S.A.

Abstract— **Multi-layer networks of threshold logic units offer an attractive framework for the design of pattern classification systems. A new constructive neural network learning algorithm (DistAI) based on inter-pattern distance is introduced. DistAI constructs a single hidden layer of spherical threshold neurons. Each neuron is designed to exclude a cluster of training patterns belonging to the same class. The weights and thresholds of the hidden neurons are determined directly by comparing the inter-pattern distances of the training patterns. This offers a significant advantage over other constructive learning algorithms that use an iterative (and often time consuming) weight modification strategy to train individual neurons. The individual clusters (represented by the hidden neurons) are combined by a single output layer of threshold neurons. The speed of DistAI makes it a good candidate for datamining and knowledge acquisition from very large datasets. Results of experiments on several artificial and real-world datasets show that DistAI compares favorably with other neural network learning algorithms for pattern classification.**

Keywords— **Constructive neural network learning algorithm, inter-pattern distance**

I. Introduction

Multi-layer networks of threshold logic units (TLU) [1], [2], [3] offer an attractive framework for the design of trainable pattern classification systems for a number of reasons including: potential for parallelism and fault and noise tolerance; significant representational and computational efficiency over disjunctive normal form (DNF) expressions and decision trees [1]; and simpler digital hardware implementations than their continuous counterparts such as sigmoid neurons used in networks trained with the error backpropagation algorithm [4].

A TLU implements an (N-1)-dimensional hyperplane which partitions N-dimensional Euclidean pattern space into two regions. A single TLU is sufficient to classify patterns in two classes if they are *linearly separable.* A number of learning algorithms that are guaranteed to find a TLU weight setting that correctly classifies a linearly separable pattern set have been proposed in the literature [1], [5], [6], [7], [8], [9], [10], [11]. However, when the given set of patterns is not linearly separable, a multi-layer network of TLUs is needed to learn a complex decision boundary that is necessary to correctly classify the training examples.

Broadly speaking, there are two approaches to the design of multi-layer neural networks for pattern classification:

- *A-priori fixed topology* networks: the number of layers, the number of hidden neurons in each hidden layer, and the connections between each neuron are defined a-priori for each classification task. This is done on the basis of problem-specific knowledge (if available), or in ad hoc fashion (requiring a process of trial and error).
- *Adaptive topology* networks: the topology of the target network is determined dynamically by introducing new neurons, layers, and connections in a controlled fashion using generative or constructive learning algorithms. In some cases, pruning mechanisms that discard redundant neurons and connections are used in conjunction with the network construction mechanisms [12], [13].

Constructive algorithms offer the following advantages over the conventional backpropagation style learning approaches: [3], [14], [15].

- They obviate the need for an *ad-hoc*, *a-priori* choice of the network topology.
- They are guaranteed to converge to zero classification errors on all finite and non-contradictory datasets.
- They use elementary threshold logic units (TLU) that are trained using the *perceptron* style weight update rules.
- They do not involve extensive parameter fine tuning.
- They provide a natural framework for exploiting problem-specific knowledge into the initial network configuration or heuristic knowledge into the network construction algorithm [16].

Several constructive algorithms that incrementally construct networks of threshold neurons for 2-category pattern classification tasks were proposed in the literature. These include the *tower* [17], [18], *pyramid* [18], *tiling* [19], *upstart* [20], *perceptron cascade* [21], and *sequential* [22]. Recently, provably correct extensions of these algorithms to handle multiple output classes and real-valued pattern attributes were proposed (see [3]). With the exception of the sequential learning algorithm, these constructive learning algorithms are based on the idea of transforming the hard task of determining the necessary network topology and weights to two subtasks:

- Incremental addition of one or more threshold neurons to the network when the existing network topology fails to achieve the desired classification accuracy on the training set.
- Training the added threshold neuron(s) using some variant of the perceptron training algorithm (e.g., the

This research was partially supported by grants from the National Science Foundation (IRI-9409580) and the John Deere Foundation to Vasant Honavar.

pocket algorithm [1]) to improve the classification accuracy of the network.

In the case of the sequential learning algorithm, hidden neurons are added and trained by an appropriate weight training rule to exclude patterns belonging to the same class from the rest of the pattern set. The time-consuming, iterative nature of the perceptron training algorithm (though considerably faster than the corresponding error guided backpropagation training) often makes the use of such algorithms impractical for very large datasets (e.g., in largescale datamining and knowledge acquisition tasks), especially in applications where reasonably accurate classifiers have to be learned in almost real time. Similarly, hybrid learning systems that use neural network learning as the inner loop of a more complex optimization process (e.g., feature subset selection using a genetic algorithm where evaluation of fitness of a solution requires training a neural network based on a subset of input features represented by the solution and evaluating its classification accuracy [23]) call for a fast neural network training algorithm.

Instance-based learning (IBL) [24], [25], [26] is an approach to learning in which the learning algorithm typically stores some or all of the training examples as prototypes. Each prototype is stored as an ordered pair $(\mathbf{X}, c)$ where $\mathbf{X}$ is a *pattern* represented in some chosen instance language (typically, in the form of a vector of attribute values), and c is the *class* to which $\mathbf{X}$ belongs. Such a system, when used to classify a new pattern $\mathbf{Y}$, uses some *distance function* (e.g., Euclidean distance in the case of real-valued patterns) that computes the distance of $\mathbf{Y}$ from each stored prototype and predicts the classification of $\mathbf{Y}$ using the known classification of the nearest prototype (or prototypes). Such algorithms, also referred to as *nearest neighbor* techniques have been investigated by researchers in pattern recognition [27], [28], [29], case-based reasoning [30], [31], artificial neural networks [32], cognitive psychology [33], and text classification [34]. Such distance-based techniques are also related to *radial basis function* networks [15], [35], [36].

We present a new constructive neural network learning algorithm (DistAl), which can be viewed as a variant of the instance-based, nearest-neighbor, and radial-basis function-based approaches to pattern classification. DistAl replaces the iterative weight update of neurons that is typically used in constructive learning algorithms by a comparison of pair-wise distances among the training patterns. Since the inter-pattern distances are computed only once during the execution of the algorithm our approach achieves a significant speed advantage over other constructive learning algorithms.

The rest of the paper is organized as follows: Section II describes DistAl. Section III presents the results of various experiments designed to evaluate the performance of neural networks trained using DistAl on some benchmark classification problems. Section IV concludes with a summary and discussion of some directions for future research.

II. The DistAl Learning Algorithm

DistAl differs from other constructive learning algorithms mentioned above in two respects:

- It uses *spherical* threshold units – a variant of the TLU – as hidden neurons. The regions that are defined (or separated) by TLUs are unbounded. This motivates us to use spherical threshold units that cover locally bounded regions [37]. A spherical threshold neuron i has associated with it a weight vector $\mathbf{W}_i$, two thresholds – $\theta_{i,low}$ and $\theta_{i,high}$, and a suitably defined distance metric d. It computes the distance $d(\mathbf{W}_i, \mathbf{X}^p)$ between a given input pattern $\mathbf{X}^p$ and $\mathbf{W}_i$. The corresponding output $o_i^p = 1$ if $\theta_{i,low} \leq d(\mathbf{W}_i, \mathbf{X}^p) \leq \theta_{i,high}$ and 0 otherwise. The spherical neuron thus identifies a cluster of patterns that lie in the region between two concentric hyperspherical regions. $\mathbf{W}_i$ represents the common center and $\theta_{i,low}$ and $\theta_{i,high}$ respectively represent the boundaries of the two regions.
- DistAl does not use an iterative algorithm for finding the weights and the thresholds. Instead, it computes the inter-pattern distance once between each pair of patterns in the training set and determines the weight values for hidden neurons by a greedy strategy (that attempts to correctly classify as many patterns as possible with the introduction of each new hidden neuron). The weights and thresholds are then set without the computationally expensive iterative process (see Section II-B for details).

A. Distance Metrics

Each hidden neuron introduced by DistAl essentially represents clusters of patterns that fall in the region bounded by two concentric hyperspherical regions in the pattern space. The weight vector of the neuron defines the center of the hyperspherical regions and the thresholds determine the boundaries of the regions (relative to the choice of the distance metric used). The choice of an appropriate distance metric for the hidden layer neurons is critical to achieving a good clustering performance. Different distance metrics represent different notions of *distance* in the pattern space. They also impose different *inductive biases* [37], [38] on the learning algorithm. Consequently, many researchers have investigated the use of alternative distance functions for instance-based learning [34], [39], [40]. The number and distribution of the clusters that result from specific choices of distance functions is a function of the distribution of the patterns as well as the clustering strategy used. Since it is difficult to identify the best distance metric in the absence of knowledge about the distribution of patterns in the pattern space, we chose to explore a number of different distance metrics proposed in the literature.

The distance between two patterns is often skewed by attributes that have high values. *Normalization* of individual attributes overcomes this problem in the distance computation. Normalization can be achieved by dividing each pattern attribute by the *range* of possible values for that attribute, or by 4 times the standard deviation for that attribute [40]. Normalization also allows attributes with nominal and/or missing values to be considered in distance computation. The distance for attributes with nominal values (say with attribute values x and y) is computed as follows [40]:

- *Overlap*: $d_{ol}(x, y) = 0$ if $x = y$; 1 otherwise.
- *Value difference*:

$$d_{vd}(x, y) = \sum_{c=1}^{C} \left| \frac{N_{a,x,c}}{N_{a,x}} - \frac{N_{a,y,c}}{N_{a,y}} \right|^q$$

where

- $N_{a,x}(N_{a,y})$: number of patterns in the training set that have value $x(y)$ for attribute a
- $N_{a,x,c}(N_{a,y,c})$: number of patterns in the training set that have value $x(y)$ for attribute a and output class c
- C: number of output classes
- q: a constant (Euclidean: 2, Manhattan: 1)

If there is a missing value in either of the patterns, the distance for that component (of the entire pattern vector) is taken to be 1.

Let $\mathbf{X}^p = [X_1^p, \cdots, X_n^p]$ and $\mathbf{X}^q = [X_1^p, \cdots, X_n^q]$ be two pattern vectors. Let max_i, min_i and σ_i be the maximum, minimum, and the standard deviation of values of the ith attribute of patterns in a dataset, respectively. Then the distance between $\mathbf{X}^p$ and $\mathbf{X}^q$, for different choices of the distance metric d is defined as follows:

1. Range, value-difference based Euclidean:

$$\sqrt{\frac{1}{n}\sum_{i=1}^{n}[(\frac{X_i^p - X_i^q}{max_i - min_i})^2 \;\; or \;\; d_{vd}(X_i^p, X_i^q)^2]}$$

2. Range, value-difference based Manhattan:

$$\frac{1}{n}\sum_{i=1}^{n}[\frac{X_i^p - X_i^q}{max_i - min_i} \;\; or \;\; d_{vd}(X_i^p, X_i^q)]$$

3. Range, value-difference based Maximum Value:

$$max_i[\frac{|X_i^p - X_i^q|}{max_i - min_i} \;\; or \;\; d_{vd}(X_i^p, X_i^q)]$$

Similarly, $4 * \sigma_i$ can be used instead of $max_i - min_i$ for standard deviation based metrics, and $d_{ol}(X_i^p, X_i^q)$ can be used instead of $d_{vd}(X_i^p, X_i^q)$ for overlap based metrics in above formulas.

4. Dice coefficient:

$$1 - \frac{2\sum_{i=1}^{n} X_i^p X_i^q}{\sum_{i=1}^{n}(X_i^p)^2 + \sum_{i=1}^{n}(X_i^q)^2}$$

5. Cosine coefficient:

$$1 - \frac{\sum_{i=1}^{n} X_i^p X_i^q}{\sqrt{\sum_{i=1}^{n}(X_i^p)^2 \cdot \sum_{i=1}^{n}(X_i^q)^2}}$$

6. Jaccard coefficient:

$$1 - \frac{\sum_{i=1}^{n} X_i^p X_i^q}{\sum_{i=1}^{n}(X_i^p)^2 + \sum_{i=1}^{n}(X_i^q)^2 - \sum_{i=1}^{n} X_i^p X_i^q}$$

7. Camberra:

$$\sum_{i=1}^{n} \frac{|X_i^p - X_i^q|}{|X_i^p + X_i^q|}$$

Attribute based clustering:
Occasionally, the values of a single attribute between two bounds (say a_{lo} and a_{hi}) might exclusively identify patterns belonging to a particular output class. Thus, a hidden neuron that remembers the name of the attribute a and the two thresholds (a_{lo} and a_{hi}) can be used to form a cluster of patterns belonging to the same class. We use the attribute based comparison to obtain homogeneous clusters in conjunction with the inter-pattern distance based clustering.

B. Network Construction

Let $S = \{\mathbf{X}^1, \mathbf{X}^2, \ldots, \mathbf{X}^N\}$ represents the N training patterns. DistAl calculates the pair-wise inter-pattern distances for the training set (using the chosen distance metric d) and stores them in the distance matrix $\mathcal{D}$. Each row of $\mathcal{D}$ is sorted in ascending order. Thus, row k of $\mathcal{D}$ corresponds to the training pattern $\mathbf{X}^k$ and the elements $\mathcal{D}[k, i]$ correspond to the distance of $\mathbf{X}^k$ to the other training patterns. $\mathcal{D}[k, 0]$ is the distance to the closest pattern and $\mathcal{D}[k, N]$ is the distance to the farthest pattern from $\mathbf{X}^k$. Simultaneously, the attribute values of the training patterns are stored in $\mathcal{D}'$. $\mathcal{D}'$ is essentially the entire training set with $\mathcal{D}'[k, i]$ representing the ith attribute value of the kth training pattern. Each column (attribute) of $\mathcal{D}'$ is sorted in ascending order.

The key idea behind DistAl is to generate a single layer of hidden neurons each of which separates a subset of patterns in a training set using $\mathcal{D}$ (or $\mathcal{D}'$). Then, they are fully connected to M output TLUs (1 for each output class) in an output layer. The representation of the patterns at the hidden layer is linearly separable [22]. Thus, an iterative perceptron learning rule can be used to train the output weights. However, the output weights can be directly set as follows: The weights between output and hidden neurons are chosen such that each hidden neuron overwhelms the effect of the hidden neurons generated later. If there are a total of h hidden neurons (numbered $1, 2, \ldots, h$ from left to right) then the weight between the output neuron j and the hidden neuron i is set to 2^{h-i} if the hidden neuron i excludes patterns belonging to class j and zero otherwise.

Let $\mathbf{W}_l^h$ be the weights between the lth hidden neuron and inputs. Let $\mathbf{W}_m^o$ be the weights between the output neuron for class m and hidden neurons, and W_{ml}^o be the weight between the output neuron for class m and the lth hidden neuron, respectively. Figure 1 summarizes the process of network construction.

C. Use of Network in Classification

The outputs in the output layer are computed by the *winner-take-all (WTA)* strategy. The output neuron m that has the highest net input produces 1 and all the other neurons produce 0's. The WTA strategy and the weight setting explained in Section II-B guarantee 100% training

Initialize the number of hidden neurons: $h = 0$;
while $S \neq \phi$ **do**

1. Double all existing weights (if any) between hidden and output neurons: $\mathbf{W}^o_m = \mathbf{W}^o_m * 2 \;\; \forall m$
2. Increment the number of hidden neurons: $h = h + 1$
3. Inter-pattern distance based:
   Identify a row k of $\mathcal{D}$ that excludes the largest subset of patterns in S that belong to the same class m as follows:
   (a) For each row $r = 1, \cdots, N$ do
      i. Let i_r and j_r be column indices (corresponding to row r) for the matrix $\mathcal{D}$ such that the patterns corresponding to the elements $\mathcal{D}[r, i_r]$, $\mathcal{D}[r, i_r + 1]$, ..., $\mathcal{D}[r, j_r]$ all belong to the same class and also belong to S.
      ii. Let $c_r = j_r - i_r + 1$ (the number of patterns excluded).
   (b) Select k to be the one for which the corresponding c_k is the largest: $k = \arg\max_r c_r$
   (c) Let S_k be the corresponding set of patterns that are excluded by pattern $\mathbf{X}^k$, $d^k_{low} = \mathcal{D}[k, i_k]$ (distance to the closest pattern of the cluster) and $d^k_{high} = \mathcal{D}[k, j_k]$ (distance to the farthest pattern of the cluster).
4. Attribute based:
   Analogously, using $\mathcal{D}'$ identify an attribute a that excludes the largest number of patterns in S that belong to the same output class m (i.e., identify a for which c_a is the largest among all attributes.); Let S_a be the corresponding set of patterns from S that are excluded by attribute a, d^a_{low} and d^a_{low} be the minimum and maximum values respectively for attribute a among the patterns in set S_a.
5. **if** [Inter-pattern distance based] **then**
   (a) Define a spherical threshold neuron with $\mathbf{W}^h = \mathbf{X}^k, \theta_{low} = d^k_{low}, \theta_{high} = d^k_{high}$.
   (b) $S = S - S_k$
   else
   (a) Define a neuron corresponding to attribute a with $\theta_{low} = d^a_{low}, \theta_{high} = d^a_{high}$.
   (b) $S = S - S_a$
6. Connect the new hidden neuron to output neurons: $W^o_{mh} = 1; W^o_{nh} = 0 \;\; \forall n \neq m$

end while

Fig. 1. **Pseudo-code for DistAI.**

accuracy for any finite non-contradictory set of training patterns.

The generalization accuracy of a test set is computed by the same way. Each test pattern is fed into the network and the outputs are computed by the WTA strategy. If there is one or more hidden neurons that produce 1 (i.e., there exist one or more hidden neurons that include the test pattern within their thresholds), the outputs are computed by the WTA strategy in the output layer. Otherwise (i.e., all hidden neurons produce 0's and all output neurons produce 0's as well), the distance between the test pattern and the thresholds of each hidden neuron is computed. The hidden neuron that has the minimum distance is chosen to produce 1. Then the outputs are computed again in the output layer to compare with the desired classification.

D. Example

Although DistAI works on tasks with multi-category real-valued patterns, we will illustrate its operation using the simple XOR problem. We will assume the use of Manhattan distance metric. There are four training patterns ($S = \{\mathbf{X}^1, \mathbf{X}^2, \mathbf{X}^3, \mathbf{X}^4\}$):

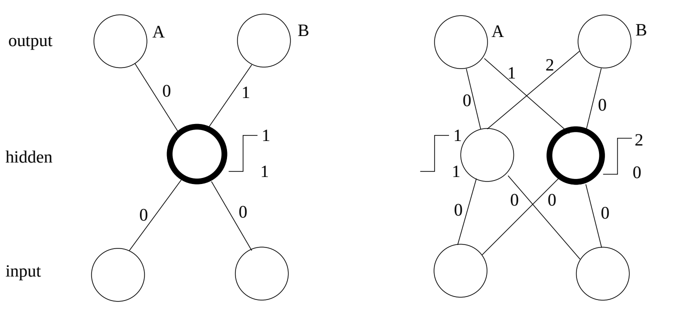
	input		class
$\mathbf{X}^1$:	0	0	A
$\mathbf{X}^2$:	0	1	B
$\mathbf{X}^3$:	1	0	B
$\mathbf{X}^4$:	1	1	A

This yields the following distance matrix after sorted:

$$\mathcal{D} = \begin{pmatrix} 0\;1\;1\;2 \\ 0\;1\;1\;2 \\ 0\;1\;1\;2 \\ 0\;1\;1\;2 \end{pmatrix}$$

The first row of the matrix is the distance of $\mathbf{X}^1, \mathbf{X}^2, \mathbf{X}^3$ and $\mathbf{X}^4$ from pattern $\mathbf{X}^1$. The second row of the matrix is the distance of $\mathbf{X}^2, \mathbf{X}^1, \mathbf{X}^4$ and $\mathbf{X}^3$ from $\mathbf{X}^2$. The third row of the matrix is the distance of $\mathbf{X}^3, \mathbf{X}^1, \mathbf{X}^4$ a nd $\mathbf{X}^2$ from $\mathbf{X}^3$. The last row of the matrix is the distance of $\mathbf{X}^4, \mathbf{X}^2, \mathbf{X}^3$ an d $\mathbf{X}^1$ from $\mathbf{X}^4$.

$\mathbf{X}^1$ excludes the maximum number of patterns from a single class (i.e., $S_k = \{\mathbf{X}^2, \mathbf{X}^3\}$, class = B). A hidden neuron is introduced for this cluster with $\mathbf{W}^h_1 = [0 \quad 0], \theta_{low} = \theta_{high} = 1, W^o_{B1} = 1, W^o_{A1} = 0$. $\mathbf{X}^2$ and $\mathbf{X}^3$ are now eliminated from further consideration (i.e., $S = S - S_k = \{\mathbf{X}^1, \mathbf{X}^4\}$) The remaining patterns ($S_k = \{\mathbf{X}^1, \mathbf{X}^4\}$, class = A) can be excluded by any pattern (say, $\mathbf{X}^1$ again) with another hidden neuron with $\mathbf{W}^h_2 = [0 \quad 0], \theta_{low} = 0, \theta_{high} = 2, W^o_{A2} = 1, W^o_{B2} = 0, W^o_{A1} = W^o_{A1} * 2 = 0, W^o_{B1} = W^o_{B1} * 2 = 2$. Now the algorithm stops since the entire training set is correctly classified (i.e., $S = S - S_k = \phi$). Figure 2 shows the network construction process.



(a) after the first neuron is introduced

(b) after the second neuron is introduced (final network)

Fig. 2. **Execution of the DistAI algorithm for the XOR problem.**

TABLE I

Comparison of the network size generated by different algorithms for the parity datasets. A '-' indicates that the result is not reported in the corresponding reference.

Algorithm	P7	P8	P9
DistAI	5	5	6
GA-MLP [41]	9	15	-
Perceptron cascade [21]	3	4	4
Cascade correlation [42]	4-5	5-6	-
Upstart [20]	6	7	8
Growth algorithm [43]	7	8	9
Sequential [22]	7	8	9
Tiling [19]	7	8	9
Tower [17]	3.5	4	4.5

E. Convergence Proof

Theorem: DistAI guarantees to converge to 100% training accuracy with a finite number of hidden neurons for a dataset with a finite number of non-contradictory patterns.
Proof: See [44] for the detailed proof.

III. Experimental Evaluation of DistAI

Experiments were performed on artificial datasets (viz. parity) and several real-world datasets available from the machine learning data repository at the University of California at Irvine [45]. The experiment was run once for each distance metric. In the case of the parity datasets the entire pattern set was used for training.

Table I presents the size of the network generated by several algorithms for parity problems. It shows that DistAI is capable of generating compact networks comparable to other algorithms for non-trivial tasks like the parity problem.

Experiments with the other datasets were conducted using ten fold cross validation to measure the generalization performance of DistAI. For each fold, the network's generalization after adding each new hidden neuron was measured. The network was allowed to train until it achieved 100% classification accuracy on the training set. The best generalization (during the process of training) was then reported. For further details on the experimental set up and the datasets see [44].

A thorough comparison of our algorithm with all the existing pattern classification algorithms is beyond the scope of our project. In what follows we compare the performance of DistAI with the results for the *k nearest-neighbor* algorithm reported in [40].

As we can see from Table II, DistAI gave comparable results to other algorithms in most datasets (except *Vowel*) despite its fast execution. In case of *Vowel* dataset, the nearest neighbor algorithm [40] reports a even higher accuracy than DistAI. [1]

[1] The best results reported in the literature [45] is 56% for *Vowel* dataset.

TABLE II

Comparison of generalization accuracy between various algorithms. DistAI is the results of our approach and NN is the best results in [40].

Dataset	DistAI	NN
Annealing	96.6	96.1
Audiology	66.0	77.5
Bridge	63.0	60.6
Cancer	97.8	95.6
Credit	87.7	81.5
Flag	65.8	58.8
Glass	70.5	72.4
Heart	86.7	83.1
Heart (Cleveland)	85.3	80.2
Heart (Hungary)	85.9	81.3
Heart (LongBeach)	80.0	71.5
Heart (Switzerland)	94.2	93.5
Hepatitis	84.7	82.6
Horse	86.0	76.8
Ionosphere	94.3	92.6
Iris	97.3	96.0
Liver	72.9	63.5
Monks-1	90.9	77.1
Monks-2	100	97.5
Monks-3	99.1	100
Pima	76.3	71.9
Promoters	88.0	92.4
Sonar	83.0	87.0
Soybean (large)	81.0	92.2
Soybean (small)	97.5	100
Vehicle	65.4	70.9
Votes	96.1	95.2
Vowel	69.8	99.2
Wine	97.1	97.8
Zoo	96.0	98.9

IV. Summary and Discussion

A fast, inter-pattern distance-based constructive learning algorithm, DistAI, is introduced and its performance on a number of datasets is demonstrated. Despite its simplicity, DistAI's performance was good on almost all the artificial and real-world datasets considered.

The simplicity of DistAI and its rapid convergence even for large datasets makes it suitable for practical pattern classification tasks (encountered in largescale datamining and knowledge acquisition) that involve very large datasets. We have applied DistAI to the task of *feature subset selection* (i.e., obtaining the optimal feature subset from the entire set of input attributes) using genetic algorithms [23]. DistAI worked fairly well (both in terms of speed and generalization) on the feature subset selection task.

A potential disadvantage of DistAI is its need for maintaining the inter-pattern distance matrix during learning. The memory needed to store this matrix grows quadratically with the size of the training set. This problem can be mitigated by freeing the memory for those patterns that are excluded by a new hidden neuron as learning progresses. It would be interesting to explore variants of DistAl that can avoid the need for maintaining the entire inter-pattern distance matrix during learning.

Constructive algorithms in general provide an natural framework for exploration of cumulative (life long) learning [46] and for knowledge-based theory refinement [16], [47]. An interesting direction for future research would be to explore the use of DistAl for this task using real-world datasets e.g., the genome data used in [16].

References

[1] S. Gallant, *Neural Network Learning and Expert Systems*, MIT Press, Cambridge, MA, 1993.

[2] C-H. Chen, R. Parekh, J. Yang, K. Balakrishnan, and V. Honavar, "Analysis of decision boundaries generated by constructive neural network learning algorithms," in *Proceedings of WCNN'95, July 17-21, Washington D.C.*, 1995, vol. 1, pp. 628–635.

[3] R. Parekh, J. Yang, and V. Honavar, "Constructive neural network learning algorithms for multi-category real-valued pattern classification," Tech. Rep. ISU-CS-TR97-06, Department of Computer Science, Iowa State University, 1997.

[4] D. Rumelhart, G. Hinton, and R. Williams, "Learning internal representations by error propagation," in *Parallel Distributed Processing: Explorations into the Microstructure of Cognition*, vol. 1 (Foundations). MIT Press, Cambridge, Massachusetts, 1986.

[5] F. Rosenblatt, "The perceptron: A probabilistic model for information storage and organization in the brain," *Psychological Review*, vol. 65, pp. 386–408, 1958.

[6] N. Nilsson, *The Mathematical Foundations of Learning Machines*, McGraw-Hill, New York, 1965.

[7] W. Krauth and M. Mézard, "Learning algorithms with optimal stability in neural networks," *J. Phys. A: Math. Gen.*, vol. 20, pp. L745–L752, 1987.

[8] J. Anlauf and M. Biehl, "Properties of an adaptive perceptron algorithm," in *Parallel Processing in Neural Systems and Computers*, pp. 153–156. 1990.

[9] M. Frean, "A thermal perceptron learning rule," *Neural Computation*, vol. 4, pp. 946–957, 1992.

[10] H. Poulard, "Barycentric correction procedure: A fast method of learning threshold units," in *Proceedings of WCNN'95, July 17-21, Washington D.C.*, 1995, vol. 1, pp. 710–713.

[11] B. Raffin and M. Gordon, "Learning and generalization with minimerror, a temperature-dependent learning algorithm," *Neural Computation*, vol. 7, pp. 1206–1224, 1995.

[12] R. Reed, "Pruning algorithms — a survey," *IEEE Transactions on Neural Networks*, vol. 4, no. 5, pp. 740–747, 1993.

[13] R. Parekh, J. Yang, and V. Honavar, "Pruning strategies for constructive neural network learning algorithms," in *Proceedings of the IEEE/INNS International Conference on Neural Networks, ICNN'97*, 1997, pp. 1960–1965.

[14] V. Honavar, *Generative Learning Structures and Processes for Generalized Connectionist Networks*, Ph.D. thesis, University of Wisconsin, Madison, 1990.

[15] V. Honavar and L Uhr, "Generative learning structures for generalized connectionist networks," *Information Sciences*, vol. 70, no. 1-2, pp. 75–108, 1993.

[16] R. Parekh and V. Honavar, "Constructive theory refinement in knowledge based neural networks," in *Proceedings of the International Joint Conference on Neural Networks*, Anchorage, Alaska, 1998, To appear.

[17] J. Nadal, "Study of a growth algorithm for a feedforward network," *International Journal of Neural Systems*, vol. 1, no. 1, pp. 55–59, 1989.

[18] S. Gallant, "Perceptron based learning algorithms," *IEEE Transactions on Neural Networks*, vol. 1, no. 2, pp. 179–191, June 1990.

[19] M. Mézard and J. Nadal, "Learning feed-forward networks: The tiling algorithm," *J. Phys. A: Math. Gen.*, vol. 22, pp. 2191–2203, 1989.

[20] M. Frean, "The upstart algorithm: A method for constructing and training feedforward neural networks," *Neural Computation*, vol. 2, pp. 198–209, 1990.

[21] N. Burgess, "A constructive algorithm that converges for real-valued input patterns," *International Journal of Neural Systems*, vol. 5, no. 1, pp. 59–66, 1994.

[22] M. Marchand, M. Golea, and P. Rujan, "A convergence theorem for sequential learning in two-layer perceptrons," *Europhysics Letters*, vol. 11, no. 6, pp. 487–492, 1990.

[23] J. Yang and V. Honavar, "Feature subset selection using a genetic algorithm," in *Feature Extraction, Construction and Selection - A Data Mining Perspective*. Kluwer Academic, NY, 1998, To appear.

[24] D. Aha, D. Kibler, and M. Albert, "Instance-based learning algorithms," *Machine Learning*, vol. 6, pp. 37–66, 1991.

[25] P. Turney, "Theoretical analyses of cross-validation error and voting in instance-based learning," *Journal of Experimental and Theoretical Artificial Intelligence*, pp. 331–360, 1994.

[26] P. Domingos, "Rule induction and instance-based learning: A unified approach," in *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI-95)*, 1995.

[27] T. Cover and P. Hart, "Nearest neighbor pattern classification," *IEEE Transactions on Information Theory*, vol. 13, pp. 21–27, 1967.

[28] E. Diday, "Recent progress in distance and similarity measures in pattern recognition," in *Proceedings of the Second International Joint Conference on Pattern Recognition*, 1974, pp. 534–539.

[29] B. Dasarathy, *Nearest Neighbor (NN) Norms: NN Pattern Classification Techiniques*, IEEE Computer Society Press, Los Alamitos, CA, 1991.

[30] C. Stanfill and D. Waltz, "Toward memory-based reasoning," *Communications of the ACM*, vol. 29, no. 12, pp. 1213–1228, 1986.

[31] J. Kolodner, *Case-Based Reasoning*, Morgan Kaufmann, San Francisco, 1993.

[32] G. Carpenter and S. Grossberg, *Pattern Recognition by Self-Organizing Neural Networks*, MIT Press, Cambridge, MA, 1991.

[33] A. Tversky, "Features of similiarity," *Psychological Review*, vol. 84, pp. 327–352, 1977.

[34] G. Salton and M. McGill, *Introduction to Modern Information Retrieval*, McGraw Hill, New York, 1983.

[35] D. Broomhead and D. Lowe, "Multivariable functional interpolation and adaptive networks," *Complex Systems*, vol. 2, pp. 321–355, 1988.

[36] F. Girosi, M. Jones, and T. Poggio, "Regularization theory and neural networks architectures," *Neural Computation*, vol. 7, pp. 219–269, 1995.

[37] P. Langley, *Elements of Machine Learning*, Morgan Kaufmann, Palo Alto, CA, 1995.

[38] T. Mitchell, *Machine Learning*, McGraw Hill, New York, 1997.

[39] R. Duda and P. Hart, *Pattern Classification and Scene Analysis*, Wiley, New York, 1973.

[40] D. Wilson and T. Martinez, "Improved heterogeneous distance functions," *Journal of Artificial Intelligence Research*, vol. 6, pp. 1–34, 1997.

[41] H. Andersen and A. Tsoi, "A constructive algorithm for the training of a multilayer perceptron based on the genetic algorithm," *Complex Systems*, vol. 7, pp. 249–268, 1993.

[42] S. Fahlman and C. Lebiere, "The cascade correlation learning algorithm," in *Neural Information Systems 2*, D. Touretzky, Ed., pp. 524–532. Morgan-Kauffman, 1990.

[43] M. Golea and M. Marchand, "A growth algorithm for neural network decision trees," *Europhysics Letters*, vol. 12, no. 3, pp. 205–210, 1990.

[44] J. Yang, R. Parekh, and V. Honavar, "DistAl: An inter-pattern distance-based constructive learning algorithm," Tech. Rep. ISU-CS-TR 97-05, Iowa State University, 1997.

[45] P. Murphy and D. Aha, "Repository of machine learning databases," Department of Information and Computer Science, University of California, Irvine, CA, 1994.

[46] S. Thrun, "Lifelong learning: A case study," Tech. Rep. CMU-CS-95-208, Carnegie Mellon University, 1995.

[47] J. W. Shavlik, "A framework for combining symbolic and neural learning," in *Artificial Intelligence and Neural Networks: Steps Toward Principled Integration*. Academic Press, Boston, 1994.