ELSEVIER

Intelligent Data Analysis 3 (1999) 55–73

IDA INTELLIGENT DATA ANALYSIS

www.elsevier.com/locate/ida

DistAl: An inter-pattern distance-based constructive learning algorithm

Jihoon Yang [*], Rajesh Parekh [†], Vasant Honavar [1]

Department of Computer Science, Artificial Intelligence Research Group, Iowa State University, 226 Atanasoff Hall, Ames, IA 50011, USA

Received 3 May 1998; received in revised form 26 May 1998; accepted 2 November 1998

Abstract

Multi-layer networks of threshold logic units (TLU) offer an attractive framework for the design of pattern classification systems. A new constructive neural network learning algorithm (**DistAl**) based on inter-pattern distance is introduced. **DistAl** constructs a single hidden layer of hyperspherical threshold neurons. Each neuron is designed to determine 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 **DistAl** makes it a good candidate for datamining and knowledge acquisition from large datasets. The paper presents results of experiments using several artificial and real-world datasets. The results demonstrate that **DistAl** compares favorably with other learning algorithms for pattern classification. © 1999 Elsevier Science B.V. All rights reserved.

Keywords: Neural networks; Constructive learning algorithms; Pattern classification

1. Introduction

Trainable pattern classifiers find a broad range of applications in data mining and knowledge discovery [1,2], intelligent agents [3,4], diagnosis [5], computer vision [6], and automated knowledge acquisition [2,7–9] from data. Multi-layer networks of threshold logic units (TLU) [10–15] 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; representational and computational efficiency over disjunctive normal form (DNF) expressions and decision trees [11]; and simpler digital hardware implementations than their continuous counterparts such as sigmoid neurons used in networks trained with error backpropagation algorithm [16,17].

[*] Corresponding author. Current address: HRL Laboratories, 3011 Malibu Canyon Road, Malibu, CA 90265-4799, USA. E-mail: yang@wins.hrl.com

[†] Current address: All State Research and Planning Center, Menlo Park, CA 94040, USA. E-mail: rpare@allstate.com

[1] Partially supported by the National Science Foundation (through grant IRI-9409580) and the John Deere Foundation.

A TLU implements an $(N-1)$-dimensional hyperplane which partitions-dimensional Euclidean pattern space into two regions. A single TLU neural network 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 [11,18–24]. 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). Learning in such networks usually amounts to (typically error gradient guided) search for a suitable setting of numerical parameters, weights in a weight space defined by the choice of the network topology.
- *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 [25,26].

Constructive algorithms offer the following advantages over the conventional backpropagation style learning approaches [12,15,27,28]:

- They obviate the need for an *ad hoc*, *a priori* choice of the network topology. Instead, they determine the network topology dynamically to give high chance of producing *optimal* (or minimal size) network.
- They are guaranteed to converge to zero classification errors on all finite and non-contradictory datasets.
- They use elementary 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 (e.g., about the general topological constraints on the network) into the network construction algorithm [29].

Several constructive algorithms that incrementally construct networks of threshold neurons for 2-category pattern classification tasks have been proposed in the literature. These include the *tower* [30,31], *pyramid* [31], *tiling* [32], *upstart* [33], *perceptron cascade* [34], and *sequential* [35]. Recently, provably correct extensions of these algorithms to handle multiple output classes and real-valued pattern attributes were proposed (see [12–14]). 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 pocket algorithm [11]) 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 [36–38]) call for a fast neural network training algorithm.

Instance-based learning (IBL) [39–42] 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 (**X**,*c*) where **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 **X** belongs. Such a system, when used to classify a new pattern **Y**, uses some *distance function* (e.g., Euclidean distance in the case of real-valued patterns) that computes the distance of **Y** from each stored prototype and predicts the classification of **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 [43–45], case-based reasoning [46–48], artificial neural networks [49], cognitive psychology [50,51], and text classification [52]. Such distance-based techniques are also related to *radial basis function* networks [28,53–55].

Rule induction algorithms [56,57] learn sets of rules corresponding to given sets of training examples. They induce a rule to cover a subset of training examples. New rules are induced iteratively until all training examples are covered.

We present a new constructive neural network learning algorithm (**DistAl**), which can be viewed as a variant of the instance-based, nearest-neighbor, radial-basis function-based, and rule induction 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 2 describes **DistAl**. Section 3 presents the results of various experiments designed to evaluate the performance of neural networks trained using **DistAl** on some benchmark classification problems. It also presents the results of experiments using **DistAl** in conjunction with a genetic algorithm-based approach to feature subset selection on several benchmark problems as well as a document classification task. Section 4 concludes with a summary and discussion of some directions for future research.

2. DistAl: A new constructive 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 [8]. A spherical threshold neuron i has associated with it a weight vector $\mathbf{W}_i$, two thresholds – $\theta_{i,\text{low}}$ and $\theta_{i,\text{high}}$, 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 $\theta_{i,\text{low}} \leqslant d(\mathbf{W}_i, \mathbf{X}^p) \leqslant \theta_{i,\text{high}}$ 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 $\theta_{i,\text{low}}$ and $\theta_{i,\text{high}}$, 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 2.2 for details).

The use of one-time inter-pattern distance calculation instead of (usually) iterative, expensive and time-consuming perceptron training procedure makes the proposed algorithm significantly faster than most other constructive learning algorithms. In fact, the time and space complexities of **DistAl** can be shown to be polynomial in the size of the training set (see Section 2.6 for details).

2.1. 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 performance. Different distance metrics represent different notions of *distance* in the pattern space. They also impose different *inductive biases* [7,8] on the learning algorithm. Consequently, many researchers have investigated the use of alternative distance functions for instance-based learning [6,44,52,58,59]. 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 [59].

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

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

where $N_{a,x}(N_{a,y})$ is the number of patterns in the training set that have value $x(y)$ for attribute a, $N_{a,x,c}(N_{a,y,c})$ is the number of patterns in the training set that have value $x(y)$ for attribute a and output class c, C is the number of output classes, q is 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 $\mathbf{X}^p = [X_1^p, \ldots, X_n^p]$ and $\mathbf{X}^q = [X_1^q, \ldots, X_n^q]$ 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 (point-to-point)

$$\sqrt{\sum_{i=1}^{n} \left[\left(\frac{X_i^p - X_i^q}{\max\limits_i - \min\limits_i} \right)^2 \text{ or } d_{\mathrm{vd}}(X_i^p, X_i^q)^2 \right]}.$$

2. Range, value-difference based Manhattan (citi-block)

$$\sum_{i=1}^{n} \left[\frac{X_i^p - X_i^q}{\max\limits_i - \min\limits_i} \text{ or } d_{\mathrm{vd}}(X_i^p, X_i^q) \right].$$

3. Range, value-difference based Maximum Value

$$\max_i \left[\frac{|X_i^p - X_i^q|}{\max\limits_i - \min\limits_i} \text{or } d_{\mathrm{vd}}(X_i^p, X_i^q) \right].$$

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

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

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.

2.2. Network construction

DistAl determines a "region" (defined by a spherical hidden neuron) iteratively by a greedy strategy (in terms of the number of training patterns). In other words, it finds a maximal subset of training patterns that can be clustered in a region. The training patterns included in a region are eliminated from further consideration. This set of *ordered* regions are generated until all patterns are included in a region. The first match is chosen for the classification. If there is no match, the closest region (by a distance metric) is chosen for the classification. Fig. 1 shows how regions are generated for a dataset of 15 patterns with two classes, O and X. R1, R2, R3, R4 and R5 are determined sequentially to cover 5, 4, 3, 2 and 1 training patterns, respectively. (Another example will be given in Section 2.4 with a detailed explanation of 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 D. Each row of D is sorted in ascending order. Thus, row k of D corresponds to the training pattern $\mathbf{X}^k$ and the elements $D[k, i]$ correspond to the distance of $\mathbf{X}^k$ to the other training patterns. $D[k, 0]$ is the distance to the closest pattern and $D[k, N]$ is the distance to the farthest pattern from $\mathbf{X}^k$. Simultaneously, the attribute values of the training patterns are stored in D'. D' is essentially the entire training set with $D'[k, i]$ representing the ith attribute value of the kth training pattern. Each column (attribute) of 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 D (or D'). Then, they are fully connected to

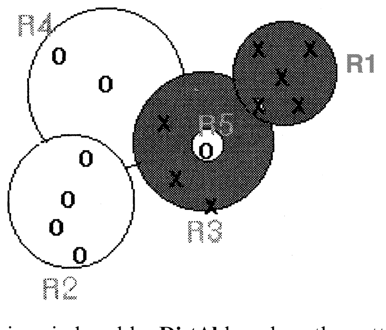

Fig. 1. Regions induced by **DistAl** based on the pattern space.

M output TLUs (1 for each output class) in an output layer. The representation of the patterns at the hidden layer is linearly separable [35]. 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. The following pseudo-code summarizes the process of network construction:

Initialize the number of hidden neurons: $h = 0$;

```
while $S$ is not empty
do
1. Double all existing weights (if any) between hidden and output neurons
       $\mathbf{W}_m^o = \mathbf{W}_m^o * 2, \forall m$
2. Increment the number of hidden neurons: $h = h + 1$
3. Inter-pattern distance based: Identify a row $k$ of $D$ that excludes the largest subset of patterns
   in $S$ that belong to the same class $m$ as follows:
   (a) For each row $r = 1, \ldots, N$ do
       (i) Let $i_r$ and $j_r$ be column indices (corresponding to row $r$) for the matrix $D$ such that the
           patterns corresponding to the elements $D[r, i_r], D[r, i_r + 1], \ldots, D[r, j_r]$ 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 $\mathbf{X}^k$, $d_{\text{low}}^k = D[k, i_k]$
       (distance to the closest pattern of the cluster) and $d_{\text{high}}^k = D[k, j_k]$ (distance to the farthest pat-
       tern of the cluster).
4. Attribute based: Analogously, using $D'$ 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 $a$, $d_{\text{low}}^a$ and $d_{\text{high}}^a$ 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 $\mathbf{W}^h = \mathbf{X}^k, \theta_{\text{low}} = d_{\text{low}}^k, \theta_{\text{high}} = d_{\text{high}}^k$.
   (b) $S = S - S_k$
   else
   (a) Define a neuron corresponding to attribute $a$ with $\theta_{\text{low}} = d_{\text{low}}^a, \theta_{\text{high}}^a$.
   (b) $S = S - S_a$.
6. Connect the new hidden neuron to output neurons: $W_{mh}^o = 1; W_{nh}^o = 0 \forall n \neq m$
end while
```

2.3. 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 Sections 2.2 and 100 training accuracy for any finite non-contradictory set of training patterns. (See Section 2.5 for detailed convergence proof).

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.

2.4. Example

Although **DistAl** 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:

$$D = \begin{matrix} 0 & 1 & 1 & 2 \\ 0 & 1 & 1 & 2 \\ 0 & 1 & 1 & 2 \\ 0 & 1 & 1 & 2 \end{matrix}$$

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 $\mathbf{X}^3, \mathbf{X}^1, \mathbf{X}^4$ and $\mathbf{X}^2$ from $\mathbf{X}^3$. The last row of the matrix is the distance of $\mathbf{X}^4, \mathbf{X}^2, \mathbf{X}^3$ and $\mathbf{X}^1$ 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 $\mathbf{W}_1^h = [00], \theta_{\text{low}} = \theta_{\text{high}} = 1$, $W_{\text{B1}}^o = 1, W_{\text{A1}}^o = 0$. $\mathbf{X}^2$ 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 $\mathbf{W}_2^h = [00], \theta_{\text{low}} = 0, \theta_{\text{high}} = 2, W_{\text{A2}}^o = 1, W_{\text{B2}}^o = 0$, $W_{\text{A1}}^o = W_{\text{A1}}^o * 2 = 0$, $W_{\text{B1}}^o = W_{\text{B1}}^o * 2 = 2$. Now the algorithm stops since the entire training set is correctly classified (i.e., $S = S - S_k = \phi$). Fig. 2 shows the network construction process.

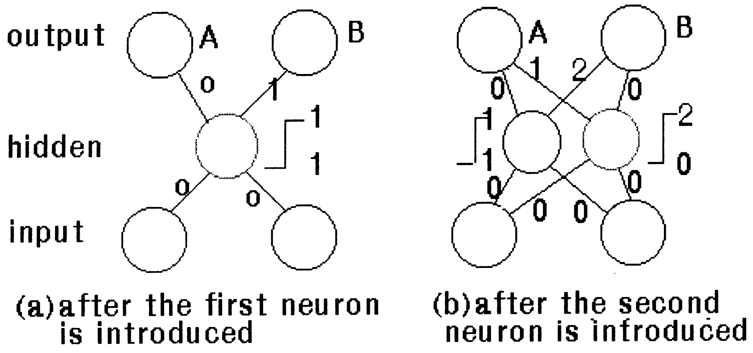
output
hidden
input
A
B
(a)after the first neuron is introduced
(b)after the second neuron is introduced

Fig. 2. Process of network construction for the example in **DistAl**.

2.5. Convergence proof

Theorem. *Given a finite non-contradictory set of training examples E, DistAl is guaranteed to converge to zero classification error after adding a finite number (h) of hidden neurons, where* $h \leqslant |E|$. *(In practice,* $h \ll |E|$.*)*

Proof. Let Z_i be the set of patterns that are excluded by *i*th hidden neuron. Each hidden neuron finds the largest subset of patterns to be excluded. **DistAl** keeps introducing a hidden neuron until S becomes an empty set (i.e., $S = S - Z_i$). Since $S = \{\mathbf{X}^1, \ldots, \mathbf{X}^N\}$ is the training set with the cardinality of N, $h = |Z_1, Z_2, \ldots, Z_h| \leqslant N$ where Z_h is the last subset of patterns to be eliminated. It is clear that at least one pattern ($\mathbf{X}^p$) can be excluded by a new hidden neuron i with $\mathbf{W}_i^h = \mathbf{X}^p$ and 0 thresholds. [2] Since there are a finite number of patterns in the training set, and since each added hidden neuron is guaranteed to correctly classify a non-empty subset of the training set which is then eliminated from further consideration, no more than $|E|$ hidden neurons are needed.

The internal representation of the hidden layer for a pattern $\mathbf{X}^p$ (which is a member of the *i*th cluster) has the form

$$\mathbf{H}^p = (0, 0, \ldots, 0, 1, *, \ldots, *) \tag{1}$$

(it has 0's in the first $i - 1$ hidden neurons, 1 in the *i*th hidden neuron and either 0 or 1 in the remaining hidden neurons) for a network with h hidden neurons. The weights from hidden to output neurons are set directly as explained in Section 2.2 and $\mathbf{W}_{ji}^o = 2^{h-i}$ if j is the right class of hidden neuron I, 0 otherwise. Consider a pattern $\mathbf{X}^p$ which belongs to the subset Z_i of patterns excluded by the *i*th hidden neuron that represents the pattern $\mathbf{X}^k$. Let c_j be the classification of $\mathbf{X}^k$. Then $W_{ji}^o > W_{li}^o \forall j \neq l$. Also, the internal representation (1) guarantees the net input of output neuron j to be larger than that of any other output neuron. Consequently, $\mathbf{X}^p$ is correctly classified

[2] Note that this is not always true for maximum value distance metric and attribute-based approach. That is because there can be many patterns of different classifications that have the same maximum values/attributes values. Therefore, the convergence proof given here and the complexity analysis in Section 2.6 apply to distance-based approaches (excluding Maximum value metric), but not attribute-based approach.

in the output layer by the WTA strategy. As an example, assume $\mathbf{H}^p = (1, 1, 1)$ for a pattern $\mathbf{X}^p$ belonging to class A, and the hidden neurons represent clusters for class A, B and B, respectively. Then, when $\mathbf{X}^p$ is fed into input neurons, the net input to the output neuron for class A will be $2^{3-1} = 4$ and that to the output neuron for class B will be $2^{3-2} + 2^{3-3} = 3$. Thus, $\mathbf{X}^p$ will be correctly classified as class A.

Therefore, **DistAl** is guaranteed to converge to zero classification error after adding a finite number of hidden neurons for a finite non-contradictory set of training examples. $\square$

2.6. Complexity analysis

This section presents the complexity analysis for **DistAl**. The complexity analysis assumes that network construction is based on a single distance metric.

Let N_{pat} be the number of training patterns and N_{att} be the number of attributes in a dataset, respectively. Let N_{out} be the number of output neurons. Assume $N_{\text{pat}} > N_{\text{att}}$ and $N_{\text{pat}} \gg \max[N_{\text{out}}, h]$.

2.6.1. Time complexity

Computing and sorting the distance matrix D takes $O(\max[N_{\text{pat}}^2 \cdot N_{\text{att}}, N_{\text{pat}}^2 \cdot \log N_{\text{pat}}])$. [3] Now, consider the pseudo-code given in Section 2.2. Step 1 takes $O(N_{\text{out}} \cdot h)$. Step 2 takes $O(1)$. Step 3 takes $O(N_{\text{pat}}^2)$ because we need to go through the entire matrix D to determine S_k. [4] Step 5 takes $O(N_{\text{pat}})$ to update S. Step 6 takes $O(N_{\text{out}})$. Thus, the **while** loop takes $O(N_{\text{pat}}^3)$ in the worst case. Therefore, the overall worst-case time complexity is $O(N_{\text{pat}}^3)$. In practice, **DistAl** runs significantly faster than the worst-case time complexity because it eliminates a subset of elements from the original training set instead of a single pattern. This makes **DistAl** particularly well-suited for largescale datamining tasks.

2.6.2. Space complexity

The space requirement for the input patterns and their targets is $O(N_{\text{pat}} \cdot [N_{\text{att}} + N_{\text{out}}])$. The weights require $O(N_{\text{out}} \cdot h + h \cdot N_{\text{in}})$. The distance matrix requires $O(N_{\text{pat}}^2)$. Thus, the total space complexity is $O(N_{\text{pat}}^2)$.

*2.7. Improving the performance of **DistAl** using feature-subset selection*

DistAl performs comparably to other learning algorithms on various real-world as well as artificial datasets. (See Section 3 for detailed comparisons). This section describes a practical way to improve the performance.

In pattern classification tasks, the choice of features (or attributes) used to represent patterns affect:

- *Learning time*: The attributes used to describe the patterns implicitly determine the search space that needs to be explored by the learning algorithm. The larger the search space, the more time the learning algorithm needs for learning a sufficiently accurate classification function [7,60].

[3] Computation of D' in attribute-based approach takes only $O(N_{\text{att}} \cdot N_{\text{pat}} \log N_{\text{pat}})$because distance computation is not necessary.

[4] Step 4 is not considered here because it is used only with the attribute-based metric. The time required for Step 4 is comparable to the time required for Step 3.

- *Number of examples needed*: All other things being equal, the larger the number of attributes used to describe the patterns, the larger is the number of examples need to learn a classification function to a desired accuracy [7,60].
- *Cost of classification*: In many real-world pattern classification tasks (e.g., medical diagnosis), some of the attributes may be observable symptoms and others might require diagnostic tests. Different diagnostic tests might have different costs as well as risks associated with them.

This presents us with a *feature subset selection problem* in automated design of pattern classifiers. The feature subset selection problem refers the task of identifying and selecting a useful subset of attributes to be used to represent patterns from a larger set of attributes. Satisfactory solution of this problem is particularly critical if instance-based, nearest-neighbor, or similarity-based learning algorithms like **DistAl** are used to build the classifier. This is due to the fact that such classifiers rely on the use of inter-pattern distances which are intricately linked to the choice of features used to represent the patterns. Presence of irrelevant or misleading features (e.g., social security numbers in a medical diagnosis task) can skew the distance calculation and hence adversely affect the generalization performance of the resulting classifier.

A detailed discussion of feature subset selection is beyond the scope of this paper. The interested reader is referred to [37,38] for discussion of a variety of alternative approaches to feature subset selection. Since exhaustive search over all possible subsets of features is computationally infeasible, most approaches make restrictive assumptions (e.g., monotonicity – which states that the addition of features does not worsen classification accuracy) or use a variety of heuristics. Genetic algorithms [61–63] offer a particularly promising approach to feature subset selection for a number of reasons [36–38]:

- They do not have to rely on the often unrealistic monotonicity assumption.
- They are particularly effective tools for exploring large search spaces for near-optimal solutions [61–63].

The use of a genetic algorithm in any search or optimization problem requires:

- choice of a representation for encoding candidate solutions to be manipulated by the genetic algorithm;
- definition of a fitness function that is used to evaluate the candidate solutions;
- definition of a selection-scheme (e.g., fitness-proportionate selection);
- definition of suitable genetic operators that are used to transform candidate solutions (and thereby explore the search space);
- setting of user-controlled parameters (e.g., probability of applying a particular genetic operator, size of the population, etc.).

In our use of genetic algorithm for feature subset selection for **DistAl**, each candidate solution represented a subset of features used to encode patterns as input to **DistAl**. The fitness of the candidate solution was computed as the generalization accuracy (computed using a 10-fold cross-validation) of a classifier constructed using **DistAl**. Standard mutation and crossover operators were used on a fixed length binary vector representation of candidate solutions (with a 1 indicating a selected feature). Experiments were run using the rank-based selection strategy with the following parameter settings:

Population size is 50; Number of generation is 300; The probability of crossover is 0.5; The probability of mutation is 0.01; The probability of selection of the highest ranked individual is 0.6. (See [37,38] for detailed explanations on the experiments).

3. Experimental evaluation of DistAl

This section presents results of experiments using **DistAl** on several benchmark problems both with and without feature subset selection and compares them with the results of other learning algorithms. It also presents the performance of **DistAl** on a real-world document classification task.

3.1. Datasets

Two artificial datasets (parity and two spirals) and a wide range of real-world datasets from the machine learning data repository at the University of California at Irvine [64] were chosen to test the performance of **DistAl**. **DistAl** is also used for classifying paper abstracts and news articles. The paper abstracts were chosen from three different sources: IEEE Expert magazine, Journal of Artificial Intelligence Research and Neural Computation. The news articles were obtained from Reuters dataset. Each document is represented in the form of a vector of numeric weights for each of the words (terms) in the vocabulary. The weights correspond to the term frequency and inverse document frequency (TFIDF) [65,66] values for the corresponding words. The training sets for paper abstracts were generated based on the classification of the corresponding documents into two classes (interesting and not interesting) by two different individuals, resulting in two different data sets (**Abstract1** and **Abstract2**). The classifications for news articles were given based on their topics (6, 4 and 8 classes) following [67], resulting in three different datasets (**Reuters1**, **Reuters2** and **Reuters3**), respectively. Table 1 summarizes the characteristics of the datasets selected for our experiments.

3.2. Experimental results

DistAl is deterministic in the sense that its behavior is always identical for a given training set. Most other constructive learning algorithms are non-deterministic because their behavior is not always identical in different runs with the same training set and even with the same learning parameters due to the randomness in selecting initial weights, pattern presentations, and so on. Therefore, just one run of **DistAl** per dataset is sufficient to study the performance.

3.2.1. Parity datasets

The seven, eight and nine-bit parity datasets (**P7**, **P8**, **P9**) were used to evaluate the performance of **DistAl** in terms of the network size. The Manhattan distance metric was used to train the entire set of patterns. Table 2 presents the size of the network generated by several algorithms. It shows that **DistAl** is capable of generating compact networks comparable to other algorithms for non-trivial tasks like the parity problem. Note that **DistAl** is also very fast. Since **DistAl** does not require iterative perceptron training procedure and keeps eliminating a subset of patterns that are not considered further in the learning process, it converges significantly fast. [5]

[5] It is not feasible to make a fair, thorough comparison of speeds of different algorithms. **DistAl** converged fairly quickly for almost all datasets. (See Section 2.6 for detailed analysis of time complexity). **GA-MLP** [68] is based on a genetic algorithm and thus it usually takes significant amount of time to get a quality solution. **Cascade correlation** [69] uses *Quickprop* [71]. *Quickprop* uses an iterative gradient descent method based on a second order heuristic.

Table 1
Datasets used in the experiments (*Size* is the number of patterns in the dataset, *Dimension* is the number of input attributes, *Missing?* is whether there are any missing values, and *Class* is the number of output classes)

Dataset	Size	Dimension	Attribute type	Missing?	Class
7-bit parity (**P7**)	128	7	Numeric	No	2
8-bit parity (**P8**)	256	8	Numeric	No	2
9-bit parity (**P9**)	512	9	Numeric	No	2
Two spirals (**2SP**)	192	2	Numeric	No	2
Annealing database (**Annealing**)	798	38	Numeric, nominal	Yes	5
Audiology database (**Audiology**)	200	69	Nominal	Yes	24
Pittsburgh bridges (**Bridges**)	105	11	Numeric, nominal	Yes	6
Breast cancer (**Cancer**)	699	9	Numeric	Yes	2
Credit screening (**CRX**)	690	15	Numeric, nominal	Yes	2
Flag database (**Flag**)	194	28	Numeric, nominal	No	8
Glass identification (**Glass**)	214	9	Numeric	No	6
Heart disease (**Heart**)	270	13	Numeric, nominal	No	2
Heart disease [Cleveland] (**HeartCle**)	303	13	Numeric, nominal	Yes	2
Heart disease [Hungarian] (**HeartHun**)	294	13	Numeric, nominal	Yes	2
Heart disease [Long Beach] (**HeartLB**)	200	13	Numeric, nominal	Yes	2
Heart disease [Swiss] (**HeartSwi**)	123	13	Numeric, nominal	Yes	2
Hepatitis domain (**Hepatitis**)	155	19	Numeric, nominal	Yes	2
Horse colic (**Horse**)	300	22	Numeric, nominal	Yes	2
Ionosphere structure (**Ionosphere**)	351	34	Numeric	No	2
Iris plants (**Iris**)	150	4	Numeric	No	3
Liver disorders (**Liver**)	345	6	Numeric	No	2
Monks problems (**Monks-1,2,3**)	432	6	Nominal	No	2
Pima indians diabetes (**Pima**)	768	8	Numeric	No	2
DNA sequences (**Promoters**)	106	57	Nominal	No	2
Sonar classification (**Sonar**)	208	60	Numeric	No	2
Large soybean (**Soylarge**)	307	35	Nominal	Yes	19
Small soybean (**Soysmall**)	47	35	Nominal	No	4
Vehicle silhouettes (**Vehicle**)	846	18	Numeric	No	4
House votes (**Votes**)	435	16	Nominal	Yes	2
Vowel recognition (**Vowel**)	528	10	Numeric	No	11
Wine recognition (**Wine**)	178	13	Numeric	No	3
Zoo database (**Zoo**)	101	16	Numeric, nominal	No	7
Paper abstracts 1 (**Abstract1**)	100	790	Numeric	No	2
Paper abstracts 2 (**Abstract2**)	100	790	Numeric	No	2
News articles 1 (**Reuters1**)	939	1568	Numeric	No	6
News articles 2 (**Reuters2**)	139	435	Numeric	No	4
News articles 3 (**Reuters3**)	834	1440	Numeric	No	8

3.2.2. Various datasets from UCI repository

DistAl was run once for each distance metric to compare the performance in terms of the generalization accuracy and the network size. A simple pruning technique was implemented to produce compact networks: When a new hidden neuron is introduced, the generalization accuracy of the network is computed. The current best generalization accuracy is stored in a pocketalong with the network size. After the training is completed (i.e., 100% training accuracy is obtained) or no further training is possible (i.e., the limit of allowable hidden neurons (currently set to 100) is reached or no more patterns can be eliminated in Maximum value metric or attribute-based

Table 2
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**
DistAl	5	5	6
GA-MLP [68]	9	15	–
Perceptron cascade [34]	3	4	4
Cascade correlation [69]	4–5	5–6	–
Upstart [33]	6	7	8
Growth algorithm [70]	7	8	9
Sequential [35]	7	8	9
Tiling [32]	7	8	9
Tower [30,31]	3.5	4	4.5

approach), the network with the best generalization accuracy in the pocket is restored by pruning the unnecessary hidden neurons.

A 10-fold cross-validation was performed for each dataset with all the distance metrics introduced in Section 2.1. No single distance metric outperformed other metrics on all datasets. That is because the performance depends on the distribution of the data. A particular distance metric might be appropriate for certain kinds of datasets while it might not for others. The Euclidean and Manhattan distance metrics outperformed other metrics in many datasets, and gave comparable results to the best ones in other datasets considered.

It is impossible to do a thorough and fair comparison between various learning algorithms since each algorithm has its own *optimal* parameter settings which is usually unknown and not feasible to obtain within a reasonable amount of time. Also, the training and test sets that had been generated and used are not identical in general under the assumption that the experiments have been done a finite number of times. (An infinite number of experiments with random partitions of training and test sets from the same distributions of data can increase the confidence level). Following comparisons should be interpreted in light of those considerations. The best results of **DistAl** are compared with the best results produced by various learning algorithms in the literature. In particular, the results in [59] are compared separately since they are recent and also obtained by a nearest-neighbor algorithm with a 10-fold cross-validation. Table 3 summarizes the comparison.

As we can see from Table 3, **DistAl** gave comparable results on most datasets (except **Audiology**, **Soylarge** and **Vehicle**).

The network size of three algorithms (**perceptron cascade** [34], **cascade correlation** [69], **upstart** [33]) for the two spirals problem is shown in [34]: 17.8 (**perceptron cascade**), 15.2 (**cascade correlation**), 91.4 (**upstart**). **DistAl** generated more compact networks with 7.7 hidden neurons.

It is shown that the combination of **DistAl** and feature subset selection yield fairly good results. The results indicate that the networks constructed using GA-selected subset of features compare quite favorably with networks that use all of the features. In particular, feature subset selection resulted in significant improvement in generalization. Also, the number of selected features is smaller than the entire set of features. For detailed explanation of implementation, results, related work and comparisons with other approaches see [37,38].

Table 3
Comparison of generalization accuracy between various algorithms. NN is the best results obtained by nearest neighbor algorithms in [59] and *Reported* is the available best results reported in the literature [72–79]

Dataset	**DistAl**	NN	Reported
2SP	83.7	–	–
Annealing	96.6	96.1	95.6
Audiology	66.0	77.5	77.7
Bridge	63.0	60.6	56.0
Cancer	97.8	95.6	95.9
CRX	87.7	81.5	85.0
Flag	65.8	58.8	–
Glass	70.5	72.4	66.3
Heart	86.7	83.0	74.8
HeartCle	85.3	80.2	77.0
HeartHun	85.9	81.3	77.0
HeartLB	80.0	71.5	79.0
HeartSwi	94.2	93.5	81.0
Hepatitis	84.7	82.6	83.0
Horse	86.0	76.8	80.9
Ionosphere	94.3	92.6	96.7
Iris	97.3	96.0	98.0
Liver	72.9	63.5	69.8
Monks-1	90.9	77.1	100
Monks-2	100	97.5	100
Monks-3	99.1	100	100
Pima	76.3	71.9	76.0
Promoters	88.0	92.4	96.2
Sonar	83.0	87.0	84.7
Soylarge	81.0	92.2	97.1
Soysmall	97.5	100	100
Vehicle	65.4	70.9	79.1
Votes	96.1	95.2	95.2
Vowel	69.8	99.2	61.0
Wine	97.1	97.8	100
Zoo	96.0	98.9	–

3.2.3. Document datasets

The same experimental setup was used as in Section 3.2.2. It is also verified that **DistAl** gives fairly good results for document classification as well. It gave reasonably high (over 80%) generalization accuracy for all datasets. Also, the GA-selected subset of features produced improved generalization accuracy, a much smaller subset of features selected with slightly larger network size. For detailed explanation of implementation, related work and comparisons with other approaches see [38,66].

4. Summary and discussion

A fast inter-pattern distance-based constructive learning algorithm, **DistAl**, is introduced and its performance on a number of datasets is demonstrated. **DistAl** is different from other constructive learning algorithms in two aspects. First, it does not require an iterative perceptron style weight update rules for determining the connections between neurons. Instead, it computes the distance (using one of the pre-defined distance metrics) between each pattern pair and uses it to set the weights (and the thresholds) between hidden neurons and inputs. The weights between the hidden and output neurons are set using a one-shot (as opposed to iterative) learning algorithm. Thus, **DistAl** is relatively fast compared in comparison with most neural network training algorithms that rely on an iterative update of weights and consequently require multiple passes through the training set. Furthermore, **DistAl** is guaranteed to converge to 100% classification accuracy on any non-contradictory training set for most of the distance metrics used in this paper. Second, it generates a single hidden layer composed of *hyperspherical* threshold neurons instead of TLU. Thus, the induced network can potentially discover natural clusters that exist in the data.

Despite its simplicity, experiments reported in this paper show that **DistAl** yields good performance on almost all real-world datasets that were considered. It also produced good performance on difficult artificial tasks such as parity and the two spirals data which have been used by numerous researchers for evaluation of supervised learning algorithms. In particular, **DistAl** is suitable to problems that have well-formed clusters and/or certain regularity (e.g., parity) in the pattern space.

DistAl, because of its reliance on inter-pattern distances, is sensitive to the presence of irrelevant or misleading attributes in the pattern representation. Consequently, its classification accuracy can be further improved by incorporating a suitable feature subset selection algorithm. This is borne out by the experiments using **DistAl** in conjunction with a genetic algorithm for feature subset selection [37,38].

A potential disadvantage of **DistAl** 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.

Because of its speed, **DistAl** is particularly well-suited to many real-world applications involving large amount of data and/or requesting real-time response such as largescale datamining and knowledge acquisition tasks and hybrid learning systems that use neural network learning as the inner loop of a more complex knowledge discovery process. An interesting direction for future research is the design of versions of **DistAl** that can be used to incremental learning and assimilation of classification knowledge from multiple, distributed, dynamic data sources. Some preliminary results based on experiments using **DistAl** to design mobile agents for text classification and retrieval from distributed document collections are reported in [66].

Constructive algorithms in general provide a natural framework for exploration of cumulative (life long) learning [80] and for knowledge-based theory refinement [29,81]. 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 [29].

References

[1] U. Fayyad, G. Piatetsky-Shapiro, P. Smyth, R. Uthurusamy, Advances in Knowledge Discovery and Data Mining, MIT Press, Cambridge, MA, 1996.

[2] V. Honavar, Machine learning: Principles and applications, in: J. Webster (Ed.), Encyclopedia of Electrical and Electronics Engineering, Wiley, New York, to appear.
[3] J. Bradshaw, Software Agents, MIT Press, Cambridge, MA, 1997.
[4] V. Honavar, Intelligent agents, in: J. Williams, K. Sochats (Eds.), Encyclopedia of Information Technology, Marcel Dekker, New York, to appear.
[5] K. Balakrishnan, V. Honavar, Intelligent diagnosis systems, International Journal of Intelligent Systems (1998).
[6] R. Duda, P. Hart, Pattern Classification and Scene Analysis, Wiley, New York, 1973.
[7] T. Mitchell, Machine Learning, McGraw-Hill, New York, 1997.
[8] P. Langley, Elements of Machine Learning, Morgan Kaufmann, Palo Alto, CA, 1995.
[9] V. Honavar, Toward learning systems that integrate multiple strategies and representations, in: V. Honavar, L. Uhr (Eds.), Artificial Intelligence and Neural Networks: Steps Toward Principled Integration, Academic Press: New York, 1994, pp. 615–644.
[10] C-H. Chen, R. Parekh, J. Yang, K. Balakrishnan, V. Honavar, Analysis of decision boundaries generated by constructive neural network learning algorithms, in: Proceedings of WCNN'95, July 17–21, Washington DC, vol. 1, 1995, pp. 628–635.
[11] S. Gallant, Neural Network Learning and Expert Systems, MIT Press, Cambridge, MA, 1993.
[12] R. Parekh, J. Yang, V. Honavar, Constructive neural network learning algorithms for multi-category real-valued pattern classification, Technical Report ISU-CS-TR97-06, Department of Computer Science, Iowa State University, 1997.
[13] R. Parekh, J. Yang, V. Honavar, Mupstart-a constructive neural network learning algorithm for multi-category pattern classification, in: Proceedings of the IEEE/INNS International Conference on Neural Networks, ICNN'97, 1997, pp. 1924–1929.
[14] J. Yang, R. Parekh, V. Honavar, MTiling- a constructive neural network learning algorithm for multi-category pattern classification, in: Proceedings of the World Congress on Neural Networks'96, San Diego, 1996, pp. 182–187.
[15] V. Honavar, Structural learning, in: J. Webster (Ed.), Encyclopediaof Electrical and Electronics Engineering, Wiley, New York, to appear.
[16] D. Rumelhart, G. Hinton, R. Williams, Learning internal representations by error propagation, in: Parallel Distributed Processing: Explorations into the Microstructure of Cognition, vol. 1 (Foundations). MIT Press, Cambridge, MA, 1986.
[17] P. Werbos, Beyond regression: New tools for prediction and analysis in behavioral sciences, Ph.D. Thesis, Harvard University, 1974.
[18] F. Rosenblatt, The perceptron: A probabilistic model for information storage and organization in the brain, Psychological Rev. 65 (1958) 386–408.
[19] N. Nilsson, The Mathematical Foundations of Learning Machines, McGraw-Hill, New York, 1965.
[20] W. Krauth, M. Mezard, Learning algorithms with optimal stability in neural networks, J. Phys. A: Math. Gen. 20 (1987) L745–L752.
[21] J. Anlauf, M. Biehl, Properties of an adaptive perceptron algorithm, in: Parallel Processing in Neural Systems and Computers. 1990, pp. 153–156.
[22] M. Frean, Small nets and short paths: Optimizing neural computation, Ph.D. Thesis, Center for Cognitive Science, Edinburgh University, UK, 1990.
[23] H. Poulard, Barycentric correction procedure: A fast method of learning threshold units, in: Proceedings of WCNN'95, July 17–21, Washington DC 1 (1995) 710–713.
[24] B. Raffin, M. Gordon, Learning and generalization with minimerror, a temperature-dependen learning algorithm, Neural Comput. 7 (1995) 1206–1224.
[25] R. Reed, Pruning algorithms – a survey, IEEE Trans. Neural Networks 4 (5) (1993) 740–747.
[26] R. Parekh, J. Yang, V. Honavar, Pruning strategies for constructive neural network learning algorithms. in: Proceedings of the IEEE/INNS InternationalConference on Neural Networks, ICNN'97, 1997. pp. 1960–1965.
[27] V. Honavar, Generative Learning structures and processes for generalized connectionist networks, Ph.D. Thesis, University of Wisconsin, Madison, 1990.
[28] V. Honavar, L. Uhr, Generative learning structures for generalized connectionist networks, Inform. Sci. 70 (1–2) (1993) 75–108.
[29] R. Parekh, V. Honavar, Constructive theory refinement in knowledge based neural networks, in: Proceedings of the International Joint Conference on Neural Networks, Anchorage, Alaska, 1998, pp. 2318–2323.
[30] J. Nadal, Study of a growth algorithm for a feedforward network, Internat. J. Neural Systems 1 (1) (1989) 55–59.
[31] S. Gallant, Perceptron based learning algorithms, IEEE Trans. Neural Networks 1 (2) (1990) 179–191.
[32] M. Mezard, J. Nadal, Learning feed-forward networks: The tiling algorithm, J. Phys. A: Math. Gen. 22 (1989) 2191–2203.
[33] M. Frean, The upstart algorithm: A method for constructing and training feedforward neural networks, Neural Comput. 2 (1990) 198–209.
[34] N. Burgess, A constructive algorithm that converges for real-valued input patterns, Internat. J. Neural Systems 5 (1) (1994) 59–66.
[35] M. Marchand, M. Golea, P. Rujan, A convergence theorem for sequential learning in two-layer perceptrons, Europhys. Lett. 11 (6) (1990) 487–492.
[36] J. Yang, V. Honavar, Feature subset selection using a genetic algorithm. in: Proceedings of the Genetic Programming Conference, GP'97, Stanford University, CA, 1997, pp. 380–385.

[37] J. Yang, V. Honavar, Feature subset selection using a genetic algorithm, IEEE Intell. Systems 13 (2) (1998) 44–49.
[38] J. Yang, V. Honavar, Feature subset selection using a genetic algorithm, in: Feature Extraction, Construction and Selection- A Data Mining Perspective. Kluwer Academic, New York, 1998.
[39] D. Aha, Incremental constructive induction: An instance-based approach, in: Proceedings of the Eighth International Workshop on Machine Learning, Evanston, IL, Morgan Kaufmann, 1991, pp. 117–121.
[40] D. Aha, D. Kibler, M. Albert, Instance-based learning algorithms, Mach. Learning 6 (1991) 37–66.
[41] P. Turney, Theoretical analyses of cross-validation error and voting in instance-based learning. J. Experimental and Theoretical Artificial Intell. 1994, pp. 331–360.
[42] P. Domingos, Rule induction and instance-based learning: A unified approach, in: Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI-95), 1995.
[43] T. Cover, P. Hart, Nearest neighbor pattern classification, IEEE Trans. Information Theory 13 (1967) 21–27.
[44] 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.
[45] B. Dasarathy, Nearest Neighbor (NN) Norms: NN Pattern Classification Techiniques, IEEE Computer Society Press, Los Alamitos, CA, 1991.
[46] C. Stanfill, D. Waltz, Toward memory-based reasoning, Commun. ACM 29 (12) (1986) 1213–1228.
[47] S. Cost, S. Salzberg, A weighted nearest neighbor algorithm for learning with symbolic features, Mach. Learning 10 (1) (1993) 57–78.
[48] J. Kolodner, Case-Based Reasoning, Morgan Kaufmann, San Francisco, 1993.
[49] G. Carpenter, S. Grossberg, Pattern Recognition by Self-Organizing Neural Networks, MIT Press, Cambridge, MA, 1991.
[50] A. Tversky, Features of similiarity, Psychological Rev. 84 (1977) 327–352.
[51] R. Nosofsky, Attention, similarity, and the identification-categorization relationship, J. Experimental Psychology: General 115 (1986) 39–57.
[52] G. Salton, M. McGill, Introduction to modern information retrieval, McGraw-Hill, New York, 1983.
[53] D. Broomhead, D. Lowe, Multivariable functional interpolation and adaptive networks, Complex Systems 2 (1988) 321–355.
[54] M. Powell, Radial basis functions for multivariable interpolation: A review, in: J. Mason, M. Cox (Eds.), Algorithms for Approximation, Clarendon Press, Oxford, 1987, pp. 143–167.
[55] F. Girosi, M. Jones, T. Poggio, Regularization theory and neural networks architectures, Neural Computation 7 (1995) 219–269.
[56] R. Michalski, I. Mozetic, J. Hong, H. Lavrac, The multi-purpose incremental learning system aq15 and its testing application to three medical domains, in: Proceedings of the Fifth National Conference on AI. Morgan Kaufmann, 1986, pp. 1041–1045.
[57] P. Clark, R. Niblett, The cn2 induction algorithm, Mach. Learning 3 (1989) 261–284.
[58] B. Batchelor, Pattern Recognition: Ideas in Practice, Plenum Press, New York, 1978.
[59] D. Wilson, T. Martinez, Improved heterogeneous distance functions, J. Artificial Intell. Res. 6 (1997) 1–34.
[60] B. Natarajan, Machine Learning: A Theoretical Approach, Morgan Kauffman, San Mateo, CA, 1991.
[61] D. Goldberg, Genetic Algorithms in Search, Optimization, and Machine Learning, Addison-Wesley, New York, 1989.
[62] M. Mitchell, An Introduction to Genetic algorithms, MIT Press, Cambridge, MA, 1996.
[63] Z. Michalewicz, Genetic Algorithms + Data Structures = Evolution Programs, 3rd ed., Springer, New York, 1996.
[64] P. Murphy, D. Aha, Repository of machine learning databases, Department of Information and Computer Science, University of California, Irvine, CA, 1994.
[65] G. Salton, Developments in automatic text retrieval, Science 53 (1991) 974–979.
[66] J. Yang, P. Pai, V. Honavar, L. Miller, Mobile intelligent agents for document classification and retrieval: A machine learning approach, in: 14th European Meeting on Cybernetics and Systems Research, Symposium on Agent Theory to Agent Implementation, Vienna, Austria, 1998.
[67] D. Koller, M. Sahami, Hierarchically classifying documents using very few words, in: International Conference on Machine Learning, 1997, pp. 170–178.
[68] H. Andersen, A. Tsoi, A constructive algorithm for the training of a multilayer perceptron based on the genetic algorithm, Complex Systems 7 (1993) 249–268.
[69] S. Fahlman, C. Lebiere, The cascade correlation learning algorithm, in: D. Touretzky (Ed.), Neural Information Systems 2, Morgan-Kauffman, 1990, pp. 524–532.
[70] M. Golea, M. Marchand, A growth algorithm for neural network decision trees, Europhysics Letters 12 (3) (1990) 205–210.
[71] S. Fahlman, Faster-learning variations on backpropagation: an empirical study, in: D. Touretzky, G. Hinton, T. Sejnowsky (Eds.), Proceedings of the 1988 Connectionist Models Summer School, Morgan-Kauffman, 1988, pp. 38–51.
[72] R. Kohavi, Feature subset selection as search with probabilistic estimates, in: AAAI Fall Symposium on Relevance, 1994.
[73] M. Richeldi, P. Lanzi. Performing effective feature selection by investigating the deep structure of the data, in: Proceedings of the Second International Conference on Knowledge Discovery and Data Mining. AAAI Press, 1996, pp. 379–383.
[74] J. Yang, V. Honavar, Experiments with the cascade-correlation algorithm, in: Proceedings of the Fourth UNB AI Symposium, Frederiction, Canada, 1991, pp. 369–380.

[75] R. Parekh, Constructive learning: Inducing grammars and neural networks, Ph.D. Thesis, Department of Computer Science, Iowa State University, Ames, IA, 1998.
[76] T. Andersen, T. Martinez The effect of decision surface fitness on dynamic multi-layer perceptron networks (dmp1), in: Proceedings of the World Congress on Neural Networks'96, pp. 177–181, San Diego, 1996, pp. 369–380.
[77] D. Lowe, Similarity metric learning for a variable-kernel classifier, Neural Computation 7 (1995) 72–85.
[78] S. Weiss, I. Kapouleas, An empirical comparison of pattern recognition, neural nets, and machine learning classification methods, in: Proceedings of the 11th International Joint Conference on Artificial Intelligence, 1989.
[79] C. Merz, P. Murphy, UCI Repository of Machine Learning Databases, 1998.
[80] S. Thrun, Lifelong learning: A case study, Technical Report CMU-CS-95-208, Carnegie Mellon University, 1995.
[81] 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.