DistAl: An Inter-pattern Distance-based Constructive Learning Algorithm
Jihoon Yang, Rajesh Parekh and Vasant Honavar
Artificial Intelligence Research Group
Department of Computer Science
226 Atanasoff Hall
Iowa State University
Ames, IA 50011, U.S.A.
{yang|parekh|honavar}@cs.iastate.edu

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 (DistAl) based on inter-pattern distance is introduced. DistAl 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. Results of experiments on several artificial and real-world datasets show that DistAl compares favorably with other neural network learning algorithms for pattern classification.

I. INTRODUCTION

Multi-layer networks of threshold logic units (TLU) [1], [2], [3], [4] 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 [5].

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 [6], [7], [8], [9], [1], [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. The choice of the network topology is mostly ad hoc and several trials are often required before a network topology suitable for the particular classification task is identified.
- **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: [14], [15], [2],

- 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.

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, pyramid [16], tiling [17], upstart [18], perceptron cascade [19], and sequential [20]. Recently, provably correct extensions of these algorithms to handle multiple output classes and real-valued pattern attributes were proposed (see [2], [3], [4]).

Several practical pattern classification tasks (encountered in largescale datamining and knowledge acquisition) involve very large datasets. Further, neural networks are used as an inner loop of a more complex optimization process in several hybrid systems. Performance requirements of these applications demand that reasonably accurate neural network classifiers be rapidly. Most constructive neural network learning algorithms use an iterative perceptron-like learning rule to train individual neurons. The itera-
tive perceptron-like training rule used in most constructive neural network learning algorithms (though considerably faster than the corresponding error guided backpropagation training) tends to slow down the learning process for very large datasets. We present a new constructive learning algorithm (DistAl) which replaces the iterative weight training 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 experiments on networks trained using DistAl on several benchmark classification problems. Section IV concludes with a summary and outlines some directions for future research.

II. THE DISTAL LEARNING ALGORITHM

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

- It uses a variant of TLUs (spherical threshold units) in the hidden layer [21]. 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 \( d(W_i, x^p) \) between a given input pattern \( x^p \) and \( \mathbf{W}_i \). The corresponding output \( o_i = 1 \) if \( \theta_{i,\text{low}} \leq d(W_i, x^p) \leq \theta_{i,\text{high}} \) and 0 otherwise. The spherical neuron thus identifies a cluster of patterns that lie in the region between two concentric hyperspherical regions. \( W_i \) represents the common center and \( \theta_{i,\text{low}} \) and \( \theta_{i,\text{high}} \), respectively, represent the boundaries of the two regions.

- It sets the weights and thresholds of the individual hidden layer neurons directly using the inter-pattern distances instead of relying on the iterative perceptron-like weight update training. The inter-pattern distances are computed once for each pair of patterns in the training set which means that the algorithm converges considerably fast even on very large datasets. In fact, the time and space complexities of DistAl can be shown to be polynomial in the size of the training set (see [22] for details).

A. Distance Metrics

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. The number and distribution of the clusters that result 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 [23], [24], [25].

The distance between two patterns is often computed on normalized values of the individual attributes. The distance between two patterns is often skewed by attributes having high magnitudes. Normalization 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 [25].

The distance computation for attributes with nominal values (say with attribute values \( x \) and \( y \)) is handled as follows [25]:

- **Overlap**: \( d_\omega(x, y) = 0 \) if \( x = y \); 1 otherwise.
- **Value difference**: \( d_{vd}(x, y) = \sum_{c=1}^{C} \frac{|N_{a,x,c} - N_{a,y}|}{N_{a,y}} \)

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 \( x \) 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 for an attribute in either of the patterns, the distance for that component (of the entire pattern vector) is 1.

Let \( x^p = [x_1^p, \ldots, x_n^p] \) and \( x^q = [x_1^q, \ldots, x_n^q] \) be two pattern vectors. Let \( \max_i, \min_i \) and \( \sigma_i \) be the maximum, minimum, and the standard deviation of values of the \( i \)-th attribute of patterns in a dataset, respectively. Then the distance between \( x^p \) and \( x^q \), for different choices of the distance metric \( d \) is defined as follows:

1. **Range, value-difference based Euclidean**:

\[
\sqrt{\frac{1}{n} \sum_{i=1}^{n} \frac{|x_i^p - x_i^q|}{\max_i - \min_i}^2} \lor \frac{d_{vd}(x_i^p, x_i^q)^2}
\]

2. **Range, value-difference based Manhattan**:

\[
\frac{1}{n} \sum_{i=1}^{n} \frac{|x_i^p - x_i^q|}{\max_i - \min_i} \lor \frac{d_{vd}(x_i^p, x_i^q)}{}
\]

3. **Range, value-difference based Maximum Value**:

\[
\arg\max_i \frac{|x_i^p - x_i^q|}{\max_i - \min_i} \lor \frac{d_{vd}(x_i^p, x_i^q)}{}
\]

Similarly, \( 4 \times \sigma_i \) can be used instead of \( \max_i - \min_i \) for standard deviation based metrics, and \( d_\omega(x_i^p, x_i^q) \) 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. Canberra:

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

Attribute based clustering:

Occasionally, the values of a single attribute between two bounds (say \(a_l\) and \(a_h\)) might exclusively identify a pattern belonging to a particular output class. Thus, a hidden neuron that remembers the name of the attribute \(a\) and the two thresholds (\(a_l\) and \(a_h\)) 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. Constructing the Network

Let \(S = \{X^1, X^2, \ldots, X^N\}\) 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 \(X^k\) and the elements \(D[k, i]\) correspond to the distances of \(X^k\) to the other training patterns with \(i = 0\) corresponding to the closest pattern and \(i = N\) corresponding to the farthest pattern from \(X^k\). Simultaneously, the attribute values of the training patterns are stored in \(D'\) with the values for each attribute sorted in ascending order.

The hidden units are added and trained as follows:

\[\text{while } S \neq \phi \text{ do}\]

- Identify a row \(k\) of \(D\) that excludes the largest subset of patterns in \(S\) that belong to the same class as follows:
  - Let \(i_k\) and \(j_k\) be column indices (corresponding to row \(k\)) for the matrix \(D\) such that the patterns corresponding to the elements \(D[k, i_k], D[k, i_k + 1], \ldots, D[k, j_k]\) all belong to the same class and also belong to \(S\).
  - Let \(c_k = j_k - i_k + 1\) (the number of patterns excluded).
  - Select \(k\) to be the one for which the corresponding \(c_k\) is the largest.
  - Let \(S_k\) be the corresponding set of patterns that are excluded by pattern \(X^k\), \(d_{\text{low}}^k = D[k, i_k]\) (distance to the closest pattern of the cluster) and \(d_{\text{high}}^k = D[k, j_k]\) (distance to the farthest pattern of the cluster).
- Analogously, using \(D'\) identify an attribute \(a\) that excludes the largest number of patterns in \(S\) that belong to the same output class (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 \(S_a\).

- if \(c_k > c_a\) then
  - Define a spherical threshold neuron with \(W = X^k, \theta_{\text{low}} = d_{\text{low}}^k, \theta_{\text{high}} = d_{\text{high}}^k\).
  - \(S = S - S_k\)
- else
  - Define a neuron corresponding to attribute \(a\) with \(\theta_{\text{low}} = d_{\text{low}}^a, \theta_{\text{high}} = d_{\text{high}}^a\).
  - \(S = S - S_a\)

An output layer of \(M\) TLU (1 for each output class) is added to the network after the single hidden layer has sequentially excluded all the patterns of the training set. The representation of the patterns at the hidden layer is linearly separable [20]. An iterative perceptron like learning rule can be used to train the output weights. The output weights can also be directly set as follows: The weights between output and hidden neurons are chosen such that each hidden neuron overweights the effect of the hidden neurons generated later. If there are a total of \(s\) hidden neurons (numbered \(1, 2, \ldots, s\) from left to right) then the weight between the output neuron \(j\) and the hidden neuron \(i\) is set to \(2^{i-s}\) if the hidden neuron \(i\) excludes patterns belonging to class \(j\) and zero otherwise.

C. Convergence Proof

Theorem:

DistAl 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 [22] for the detailed proof.

D. Example

Consider the simple XOR problem assuming Manhattan distance metric for simplicity:

<table>
<thead>
<tr>
<th>input</th>
<th>class</th>
</tr>
</thead>
<tbody>
<tr>
<td>pattern 1: 0 0</td>
<td>A</td>
</tr>
<tr>
<td>pattern 2: 0 1</td>
<td>B</td>
</tr>
<tr>
<td>pattern 3: 1 0</td>
<td>B</td>
</tr>
<tr>
<td>pattern 4: 1 1</td>
<td>A</td>
</tr>
</tbody>
</table>

Then, \(D\) is:

\[
\begin{pmatrix}
 0 & 1 & 1 & 2 \\
 0 & 1 & 1 & 2 \\
 0 & 1 & 1 & 2 \\
 0 & 1 & 1 & 2
\end{pmatrix}
\rightarrow
\begin{pmatrix}
 1 & 2 & 3 & 4 \\
 2 & 1 & 4 & 3 \\
 3 & 1 & 4 & 2 \\
 4 & 2 & 3 & 1
\end{pmatrix}
\]

\[
\begin{pmatrix}
 0 & 1 & 1 & 2 \\
 0 & 1 & 1 & 2 \\
 0 & 1 & 1 & 2 \\
 0 & 1 & 1 & 2
\end{pmatrix}
\rightarrow
\begin{pmatrix}
 1 & 2 & 3 & 4 \\
 2 & 1 & 4 & 3 \\
 3 & 1 & 4 & 2 \\
 4 & 2 & 3 & 1
\end{pmatrix}
\]

Let \(W^h_i\) be the weights between the \(i\)th hidden neuron and inputs, and let \(W^h_{A(B)}\) be the weight between output neuron for class \(A(B)\) and \(i\)th hidden neuron.

Pattern 1 excludes the maximum number of patterns from a single class (i.e., patterns 2 and 3). A hidden neuron is introduced for this cluster with \(W^h_1 = [0 \ 0], W^h_2 = 1, W^h_3 = 0\) and \(\theta_{\text{low}} = \theta_{\text{high}} = 1\). Then, patterns 2 and 3 are eliminated from further consideration, which leaves pattern 1 and 4 and they can be excluded from any pattern (say, pattern 1 again) with another hidden neuron with
\( W_1 = [0 \ 0], W_{A2} = 1, W_{B2} = 0, \theta_{low} = 0, \theta_{high} = 2, \text{ and } W_{A1} = 2 \times W_{A1}, W_{B1} = 2 \times W_{B1}. \) Now, all the patterns are correctly classified and the algorithm stops. Figure 1 shows this process of network construction.

![Diagram of network construction](image)

Fig. 1. Execution of the DistAl algorithm for the XOR problem

### III. Experiments

Experiments were performed on two artificial datasets (viz. parity and two-spirals) and several real-world datasets available from the machine learning data repository at the University of California at Irvine.\(^1\) The experiment was run once for each distance metric. The output of the DistAl network was computed using the winner-take-all strategy. In the case of the parity datasets the entire pattern set was used for training. DistAl was able to correctly classify the 7,8, and 9 bit parity problems using 5,6, and 6 hidden neurons respectively. Experiments with the other datasets were conducted using ten fold cross validation to measure the generalization performance of DistAl. 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 [22].

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 DistAl with the results for the \(k\) nearest-neighbor algorithm reported in [25] and the results for the best generalization performance for some of the datasets from the results reported in the UCI repository.

As we can see from Table 1, DistAl gave comparable results to other algorithms in most datasets (except Soybean (large)) despite its fast execution. In case of Vowel dataset, Wilson’s paper reports a surprisingly higher accuracy than DistAl and other algorithms in the literature.

### IV. 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. Despite its simplicity, DistAl’s performance was good on almost all the artificial and real-world datasets considered.

The simplicity of DistAl and its rapid convergence even for large datasets makes it suitable for practical pattern classification tasks (encountered in large scale datamining and knowledge acquisition) that involve very large datasets. We have applied DistAl to the task of feature subset selection (i.e., obtaining the optimal feature subset from the entire set of input attributes) using genetic algorithms [26]. DistAl worked fairly well (both in terms of speed and generalization) on the feature subset selection task.

Detailed comparison of the performance DistAl with that of the different constructive learning algorithms discussed in [2] on a large number of datasets is currently in progress.

### References


\(^1\) [http://www.ics.uci.edu/Al/ML/MLDBRepository.htm](http://www.ics.uci.edu/Al/ML/MLDBRepository.htm)
TABLE I
Comparison of generalization of DistAl with other approaches. Wilson is the best results in [25] and Reported is the available best results documented in the data repository at the University of California at Irvine.

<table>
<thead>
<tr>
<th>Dataset</th>
<th>DistAl</th>
<th>Wilson</th>
<th>Reported</th>
</tr>
</thead>
<tbody>
<tr>
<td>Annealing</td>
<td>96.6</td>
<td>96.1</td>
<td>-</td>
</tr>
<tr>
<td>Audiology</td>
<td>66.0</td>
<td>77.5</td>
<td>-</td>
</tr>
<tr>
<td>Bridge</td>
<td>63.0</td>
<td>60.6</td>
<td>-</td>
</tr>
<tr>
<td>Cancer</td>
<td>97.8</td>
<td>95.6</td>
<td>94</td>
</tr>
<tr>
<td>Credit</td>
<td>87.7</td>
<td>81.5</td>
<td>-</td>
</tr>
<tr>
<td>Flag</td>
<td>65.8</td>
<td>58.8</td>
<td>-</td>
</tr>
<tr>
<td>Glass</td>
<td>70.5</td>
<td>72.4</td>
<td>-</td>
</tr>
<tr>
<td>Heart</td>
<td>86.7</td>
<td>83.0</td>
<td>77</td>
</tr>
<tr>
<td>Heart (Cleveland)</td>
<td>85.3</td>
<td>80.2</td>
<td>-</td>
</tr>
<tr>
<td>Heart (Hungary)</td>
<td>85.9</td>
<td>81.3</td>
<td>-</td>
</tr>
<tr>
<td>Heart (LongBeach)</td>
<td>80.0</td>
<td>71.5</td>
<td>-</td>
</tr>
<tr>
<td>Heart (Swiss)</td>
<td>94.2</td>
<td>93.5</td>
<td>-</td>
</tr>
<tr>
<td>Hepatitis</td>
<td>84.7</td>
<td>82.6</td>
<td>83</td>
</tr>
<tr>
<td>Horse</td>
<td>86.0</td>
<td>76.8</td>
<td>-</td>
</tr>
<tr>
<td>Ionosphere</td>
<td>94.3</td>
<td>92.6</td>
<td>97</td>
</tr>
<tr>
<td>Iris</td>
<td>97.3</td>
<td>96.0</td>
<td>-</td>
</tr>
<tr>
<td>Liver</td>
<td>72.9</td>
<td>63.5</td>
<td>-</td>
</tr>
<tr>
<td>Monks-1</td>
<td>90.9</td>
<td>77.1</td>
<td>100</td>
</tr>
<tr>
<td>Monks-2</td>
<td>100</td>
<td>97.5</td>
<td>100</td>
</tr>
<tr>
<td>Monks-3</td>
<td>99.1</td>
<td>100</td>
<td>100</td>
</tr>
<tr>
<td>Pima</td>
<td>76.3</td>
<td>71.9</td>
<td>76</td>
</tr>
<tr>
<td>Promoters</td>
<td>88.0</td>
<td>92.4</td>
<td>-</td>
</tr>
<tr>
<td>Sonar</td>
<td>83.0</td>
<td>87.0</td>
<td>83</td>
</tr>
<tr>
<td>Soybean (large)</td>
<td>81.0</td>
<td>92.2</td>
<td>97</td>
</tr>
<tr>
<td>Soybean (small)</td>
<td>97.5</td>
<td>100</td>
<td>-</td>
</tr>
<tr>
<td>Vehicle</td>
<td>65.4</td>
<td>70.9</td>
<td>-</td>
</tr>
<tr>
<td>Votes</td>
<td>96.1</td>
<td>95.2</td>
<td>95</td>
</tr>
<tr>
<td>Vowel</td>
<td>69.8</td>
<td>99.2</td>
<td>56</td>
</tr>
<tr>
<td>Wine</td>
<td>97.1</td>
<td>97.8</td>
<td>100</td>
</tr>
<tr>
<td>Zoo</td>
<td>96.0</td>
<td>98.9</td>
<td>-</td>
</tr>
</tbody>
</table>