Learning with preknowledge: clustering with point and graph matching distance measures

Steven Gold, Anand Rangarajan, and Eric Mjolsness

Research Report YALEU/DCS/RR-1037
May 1994

YALE UNIVERSITY
DEPARTMENT OF COMPUTER SCIENCE

Learning with preknowledge: clustering with point and graph matching distance measures

Steven Gold, Anand Rangarajan and Eric Mjolsness
Department of Computer Science
Yale University
New Haven, CT 06520-8285

Abstract

Prior knowledge constraints are imposed upon a learning problem in the form of distance measures. Prototypical 2-D point sets and graphs are learned by clustering with point matching and graph matching distance measures. The point matching distance measure is invariant under affine transformations - translation, rotation, scale and shear - and permutations. It operates between noisy images with missing and spurious points. The graph matching distance measure operates on weighted graphs and is invariant under permutations. Learning is formulated as an optimization problem. Large objectives so formulated ($\sim$ million variables) are efficiently minimized using a combination of optimization techniques - algebraic transformations, projection methods, clocked objectives, and deterministic annealing.

1 Introduction

While few biologists today would subscribe to Locke's description of the mind as a tabula rasa, the nature of the inherent constraints - Kant's preknowledge - that helps organize our perceptions remains much in doubt. Recently, the importance of such preknowledge for learning has been convincingly argued from a statistical framework [Geman et al, 1992]. Several researchers have proposed that our brains may incorporate preknowledge in the form of distance measures [Shepard, 1987;

von der Malsburg, 1988; Bienenstock and Doursat, 1991]. We have recently begun to explore this idea [Gold, Mjolsness, Rangarajan, 1994]. In this work, by enhancing both the scope and function of previously examined distance measures, we significantly expand the problem domains where learning may take place.

We learn objects consisting of noisy 2-D point-sets or noisy weighted graphs by clustering with point matching and graph matching distance measures. The point matching measure is invariant under permutations and affine transformations (separately decomposed into translation, rotation, scale and shear) and operates on point-sets with missing or spurious points. The graph matching measure is invariant under permutations. These distance measures and others like them may be constructed using Bayesian inference on a probabilistic model of the visual domain. Such models introduce a carefully designed bias into our learning, which reduces its generality outside the problem domain but increases its ability to generalize within the problem domain. From a statistical viewpoint, outside the problem domain it increases bias while within the problem domain it decreases variance. The resulting distance measures are similar to some of those hypothesized for cognition.

The distance measures and learning problem (clustering) are formulated as objective functions. Fast minimization of these objectives is achieved by a combination of optimization techniques - algebraic transformations, projection methods, clocked objectives, and deterministic annealing. Combining these techniques significantly increases the size of problems which may be solved with recurrent network architectures [Rangarajan, Gold, and Mjolsness, 1994]. Even on single-processor workstations, non-linear objectives with a million variables can routinely be minimized. With these methods we learn prototypical examples of 2-D points set and graphs from randomly generated experimental data.

2 Theory

2.1 An Affine Invariant Point Matching Distance Measure

The first distance measure quantifies the degree of dissimilarity between two unlabeled 2-D point images, irrespective of bounded affine transformations, i.e. differences in position, orientation, scale and shear. The two images may have a different number of points. The measure is calculated with an objective that can be used to find correspondence and pose for unlabeled feature matching in vision. Given two sets of points $\{X_j\}$ and $\{Y_k\}$, one can minimize the following objective to find the affine transformation and permutation which best maps Y onto X:

$$E_{pm}(m,t,A) = \sum_{j=1}^{J}\sum_{k=1}^{K} m_{jk}\|X_j - t - AY_k\|^2 + g(A) - \alpha\sum_{j=1}^{J}\sum_{k=1}^{K} m_{jk}$$

with constraints: $\forall j \ \sum_{k=1}^{K} m_{jk} \leq 1$, $\forall k \ \sum_{j=1}^{J} m_{jk} \leq 1$, $\forall jk \ m_{jk} \geq 0$ and

$$g(A) = \gamma a^2 + \kappa b^2 + \lambda c^2$$

A is decomposed into scale, rotation, vertical shear and oblique shear as follows:

$$A = s(a)R(\Theta)Sh_1(b)Sh_2(c) \ ,$$

where

$$s(a) = \begin{pmatrix} e^a & 0 \\ 0 & e^a \end{pmatrix}, \; Sh_1(b) = \begin{pmatrix} e^b & 0 \\ 0 & e^{-b} \end{pmatrix}, \; Sh_2(c) = \begin{pmatrix} \cosh(c) & \sinh(c) \\ \sinh(c) & \cosh(c) \end{pmatrix}$$

$R(\Theta)$ is the standard 2x2 rotation matrix. $g(A)$ serves to regularize the affine transformation by bounding the scale and shear components. m is a fuzzy correspondence matrix which matches points in one image with corresponding points in the other image. The constraints on m ensure that each point in each image corresponds to at most one point in the other image. However, partial matches are allowed, in which case the sum of these partial matches may add up to no more than one. The inequality constraint on m permits a null match or multiple partial matches.

The α term biases the objective towards matches. The decomposition of A in the above is not required, since A could by left as a 2x2 matrix and solved for directly in the algorithm that follows. The decomposition just provides for more precise regularization, i.e., specification of the likely kinds of transformations. Also $Sh_2(c)$ could be replaced by another rotation matrix.

Then given two sets of points $\{X_j\}$ and $\{Y_k\}$ the distance between them is defined as:

$$D(\{X_j\},\{Y_k\}) = \min_{m,t,A}(E_{pm}(m,t,A) \mid \text{ constraints on } m)$$

This measure is an example of a more general image distance measure derived in [Mjolsness, 1992]:

$$d(x,y) = \min_T d(x,T(y)) \in [0,\infty)$$

where T is a set of transformation parameters introduced by a visual grammar.

We transform our inequality constraints into equality constraints by introducing slack variables, a standard technique from linear programming:

$$\forall j \sum_{k=1}^{K} m_{jk} \leq 1 \;\; \rightarrow \;\; \forall j \sum_{k=1}^{K+1} m_{jk} = 1$$

and likewise for our column constraints. An extra row and column is added to the permutation matrix m to hold our slack variables. These constraints are enforced by applying the Potts glass mean field theory approximations [Peterson and Söderberg, 1989] and a Lagrange multiplier and then using an equivalent form of the resulting objective, which employs Lagrange multipliers and an $x \log x$ barrier function [Yuille and Kosowsky, 1994]:

$$\begin{aligned} E_{pm}(m,t,A) \;\; = \;\; & \sum_{j=1}^{J}\sum_{k=1}^{K} m_{jk}\|X_j - t - AY_k\|^2 + g(A) - \alpha\sum_{j=1}^{J}\sum_{k=1}^{K} m_{jk} \\ & + \frac{1}{\beta}\sum_{j=1}^{J+1}\sum_{k=1}^{K+1} m_{jk}(\log m_{jk} - 1) \\ & + \sum_{j=1}^{J}\mu_j(\sum_{k=1}^{K+1} m_{jk} - 1) + \sum_{k=1}^{K}\nu_k(\sum_{j=1}^{J+1} m_{jk} - 1) \end{aligned} \tag{1}$$

In this objective, we are looking for a saddle point. Equation (1) is minimized with respect to m, t, and A which are the correspondence matrix, translation, and affine transform, and is maximized with respect to μ and ν, the Lagrange multipliers that enforce the row and column constraints for m. m is fuzzy, with the degree of fuzziness dependent upon β.

The above defines a series of distance measures, since given the decomposition of A it is trivial to construct measures which are invariant only under some subset of the transformations (such as rotation and translation). The regularization and α terms may also be individually adjusted in an appropriate fashion for a specific problem domain.

2.2 Weighted Graph Matching Distance Measures

The following distance measure quantifies the degree of dissimilarity between two unlabeled weighted graphs. Given two graphs, represented by adjacency matrices G_{jl} and g_{km}, one can minimize the objective below to find the permutation which best maps G onto g [Rangarajan and Mjolsness, 1994]:

$$E_{gm}(m) = \sum_{j=1}^{J}\sum_{k=1}^{K}(\sum_{l=1}^{L} G_{jl}m_{lj} - \sum_{m=1}^{M} m_{jm}g_{mk})^2$$

with constraints: $\forall j \ \sum_{k=1}^{K} m_{jk} = 1$, $\forall k \ \sum_{j=1}^{J} m_{jk} = 1$, $\forall jk \ m_{jk} \geq 0$. These constraints are enforced in the same fashion as in (2) with an $x \log x$ barrier function and Lagrange multipliers. The objective is simplified with a fixed point preserving transformation of the form $X^2 \rightarrow 2\sigma X - \sigma^2$. The additional variable (σ) introduced in such a transformation, described as a reversed neuron in [Mjolsness and Garrett, 1990], is similar to a Lagrange parameter. A self-amplification term is also added to push the match variables towards 0 or 1. This term (with the γ parameter below) is similarly transformed with a reversed neuron. The resulting objective is:

$$\begin{aligned} E_{gm}(m) \quad = \quad & \sum_{j=1}^{J}\sum_{k=1}^{K}\mu_{jk}(\sum_{l=1}^{L} G_{jl}m_{lk} - \sum_{m=1}^{M} m_{jm}g_{mk}) - \frac{1}{2}\sum_{j=1}^{J}\sum_{k=1}^{K}\mu_{jk}^2 \\ & -\gamma\sum_{j=1}^{J}\sum_{k=1}^{K}\sigma_{jk}m_{jk} + \frac{\gamma}{2}\sum_{j=1}^{J}\sum_{k=1}^{K}\sigma_{jk}^2 \\ & +\frac{1}{\beta}\sum_{j=1}^{J}\sum_{k=1}^{K} m_{jk}(\log m_{jk} - 1) \\ & +\sum_{j=1}^{J}\kappa_j(\sum_{k=1}^{K} m_{jk} - 1) + \sum_{k=1}^{K}\lambda_k(\sum_{j=1}^{J} m_{jk} - 1) \end{aligned} \tag{2}$$

As in section 2.1, we look for a saddle point. Equation (2) is minimized with respect to m and σ which are the correspondence matrix and reversed neuron of the transform, and is maximized with respect to κ, λ, and μ, the Lagrange multipliers that enforce the row and column constraints for m and the reversed neuron parameter enforcing the first fixed point transformation. m may be fuzzy, so a given vertex in

one graph may partially match several vertices in the other graph, with the degree of fuzziness dependent upon β, however the self-amplification term dramatically reduces the fuzziness at high β.

A second, functionally equivalent, graph matching objective is also used in the clustering problem (as explained in section 3.3):

$$E_{gm'}(m) = \sum_{j=1}^{J}\sum_{l=1}^{L}\sum_{k=1}^{K}\sum_{m=1}^{M} m_{jk}m_{lm}(G_{jl} - g_{km})^2 \tag{3}$$

with constraints: $\forall j \ \sum_{k=1}^{K} m_{jk} = 1, \ \forall k \ \sum_{j=1}^{J} m_{jk} = 1, \ \forall jk \ m_{jk} \geq 0.$

2.3 The Clustering Objective

The learning problem is formulated as follows: Given a set of I objects, $\{X_i\}$ find a set of A cluster centers $\{Y_a\}$ and match variables $\{M_{ia}\}$ defined as

$$M_{ia} = \begin{cases} 1 & \text{if } X_i \text{ is in } Y_a\text{'s cluster} \\ 0 & \text{otherwise,} \end{cases}$$

such that each object is in only one cluster, and the total distance of all the objects from their respective cluster centers is minimized. To find $\{Y_a\}$ and $\{M_{ia}\}$ minimize the cost function,

$$E_{cluster}(Y, M) = \sum_{i=1}^{I}\sum_{a=1}^{A} M_{ia}D(X_i, Y_a)$$

with constraints: $\forall i \ \sum_a M_{ia} = 1, \ \forall ia \ M_{ia} \geq 0$. $D(X_i, Y_a)$, the distance function, is a measure of dissimilarity between two objects.

The constraints on M are enforced in a manner similar to that described for the distance measure, except that now only the rows of the matrix M need to add to one, instead of both the rows and the columns. The Potts glass mean field theory method is applied and an equivalent form of the resulting objective is used:

$$\begin{aligned} E_{cluster}(Y, M) \ = \ & \sum_{i=1}^{I}\sum_{a=1}^{A} M_{ia}D(X_i, Y_a) + \frac{1}{\beta}\sum_{i=1}^{I}\sum_{a=1}^{A} M_{ia}(\log M_{ia} - 1) \\ & + \sum_{i=1}^{I}\lambda_i(\sum_{a=1}^{A} M_{ia} - 1) \end{aligned} \tag{4}$$

Here, the objects are point-sets or weighted graphs. If point-sets are used, the distance measure $D(X_i, Y_a)$ is replaced by (1), if graphs it is replaced by (2) or (3). For example, after replacing the distance measure by (1), we obtain:

$$\begin{aligned} E_{cluster}(Y, M, t, A, m) = & \sum_{i=1}^{I}\sum_{a=1}^{A} M_{ia}[\sum_{j=1}^{J}\sum_{k=1}^{K} m_{iajk}(\|X_{ij} - t_{ia} - A_{ia}Y_{ak}\|^2 - \alpha) \\ & + g(A_{ia})] + \sum_{i=1}^{I}\sum_{a=1}^{A}[\frac{1}{\beta_m}\sum_{j=1}^{J+1}\sum_{k=1}^{K+1} m_{iajk}(\log m_{iajk} - 1) + \sum_{j=1}^{J}\mu_{iaj}(\sum_{k=1}^{K+1} m_{iajk} - 1) \end{aligned}$$

$$+\sum_{k=1}^{K}\nu_{iak}(\sum_{j=1}^{K+1}m_{iajk}-1)]+\frac{1}{\beta_M}\sum_{i=1}^{I}\sum_{a=1}^{A}M_{ia}(\log M_{ia}-1)$$
$$+\sum_{i=1}^{I}\lambda_i(\sum_{a=1}^{A}M_{ia}-1) \quad (5)$$

A saddle point is required. The objective is minimized with respect to Y, M, m, t, and A which are respectively the cluster centers, the cluster membership matrix, the correspondence matrices, the translations and affine transformations. It is maximized with respect to λ, which enforces the row constraint for M, and μ and ν which enforce the column and row constraints for m. M is a cluster membership matrix indicating for each object i, which cluster a it falls in, and m_{ia} is a permutation matrix which assigns to each point in cluster center Y_a a corresponding point in object X_i. (A_{ia}, t_{ia}) gives the affine transform between object i and cluster center a. Both M and m are fuzzy, so a given object may partially fall in several clusters, with the degree of fuzziness depending upon β_m and β_M.

Therefore, given a set of objects, X, we construct $E_{cluster}$ and upon finding the appropriate saddle point of that objective, we will have Y, their cluster centers, and M, their cluster memberships.

An objective similar to (5) may be constructed using the graph matching distance measure in (2) or (3) instead.

3 The Algorithm

3.1 Overview - Clocked Objective Functions

The algorithm to minimize the clustering objectives consists of two loops - an inner loop to minimize the distance measure objective (either (1) or (2)) and an outer loop to minimize the clustering objective (4). Using coordinate descent in the outer loop results in dynamics similar to the EM algorithm for clustering [Hathaway, 1986]. The EM algorithm has been similarly used in supervised learning [Jordan and Jacobs, 1993]. All variables occurring in the distance measure objective are held fixed during this phase. The inner loop uses coordinate ascent/descent which results in repeated row and column normalizations for m. The minimization of m, and the distance measure variables (either t, A of (1) or μ, σ of (2)), occurs in an incremental fashion - that is their values are saved after each inner loop call from within the outer loop and are then used as initial values for the next call to the inner loop. This tracking of the values of the distance measure variables in the inner loop is essential to the efficiency of the algorithm since it greatly speeds up each inner loop optimization. Most coordinate ascent/descent phases are computed analytically, further speeding up the algorithm. Poor local minima are avoided, by deterministic annealing in both the outer and inner loops.

The resulting dynamics can be concisely expressed by formulating the objective as a clocked objective function, which is optimized over distinct sets of variables in phases, (letting $\{D\}$ be the set of distance measure variables (e.g. $\{A,t\}$ for (1))

excluding the match matrix),

$$E_{clocked} = E_{cluster}\langle\langle\langle(\mu, m)^A, (\nu, m)^A\rangle_\oplus, \{D\}\rangle_\oplus, (\lambda, M)^A, Y^A\rangle_\oplus$$

with this special notation employed recursively:

$E\langle x, y\rangle_\oplus$: coordinate descent on x, then y, iterated (if necessary)
x^A : use analytic solution for x phase

The algorithm can be expressed less concisely in English, as follows:

Initialize $\{D\}$ to the equivalent of an identity transform, Y to random values
Begin Outer Loop
  Begin Inner Loop
    Initialize $\{D\}$ with previous values
    Find m, $\{D\}$ for each ia pair :
      Find m by softmax, normalizing across j, then k, iteratively
      Find $\{D\}$ by coordinate descent
  End Inner Loop
  If first time through outer loop $\uparrow \beta_m$ and repeat inner loop
  Find M,Y using fixed values of m, $\{D\}$, determined in inner loop:
    Find M by softmax, across i
    Find Y by coordinate descent
  $\uparrow \beta_M$, β_m
End Outer Loop

When the distances are calculated for all the X - Y pairs the first time through the outer loop, annealing is needed to minimize the objectives accurately. However on each succeeding iteration, since good initial estimates are available for $\{D\}$, (the values from the previous iteration of the outer loop) annealing is unnecessary and the minimization is much faster.

The speed of the above algorithm is increased by not recalculating the X - Y distance for a given ia pair when its M_{ia} membership variable drops below a threshold.

3.2 Inner Loop

The inner loop proceeds in two phases. In phase one, while $\{D\}$ are held fixed, m is initialized with the softmax function and then iteratively projected across its rows and columns until the procedure converges. In phase two $\{D\}$ are updated using coordinate descent. Then β_m is increased and the loop repeats. Let E_{dmwc} be the distance measure objective [(1) or (2)] without the terms that enforce the constraints (i.e. the $x \log x$ barrier function and the Lagrange parameters).

In phase one, m is updated with softmax:

$$m_{iajk} = \frac{\exp(-\beta_m \partial E_{dmwc}(X_i, Y_a)/\partial m_{iajk})}{\sum_{k'=1}^{K+1} \exp(-\beta_m \partial E_{dmwc}(X_i, Y_a)/\partial m_{iajk'})}$$

Then m is iteratively normalized across j and k until $\sum_{j=1}^{J}\sum_{k=1}^{K} \Delta m_{iajk} < \epsilon$:

$$m_{iajk} = \frac{m_{iajk}}{\sum_{j'=1}^{J+1} m_{iaj'k}} \;\; ; \;\; m_{iajk} = \frac{m_{iajk}}{\sum_{k'=1}^{K+1} m_{iajk'}}$$

Using coordinate descent, the $\{D\}$ are updated in phase two. If a member of $\{D\}$ cannot be computed analytically (such as the terms of A which are regularized), Newton's method is used to compute the root of the function. So if d_n is the nth member of $\{D\}$ then in phase two we update d_{ian} such that:

$$\frac{\partial E_{dmwc}(X_i, Y_a)}{\partial d_{ian}} = 0$$

Finally β_m is increased and the loop repeats.

By setting the partial derivatives of E_{dm} to zero and initializing the Lagrange parameters to zero, the algorithm for phase one may be derived.

Beginning with a small β_m allows minimization over a fuzzy correspondence matrix m, for which a global minimum is easier to find. Raising β_m drives the m's closer to 0 or 1, as the algorithm approaches a saddle point.

3.3 Outer Loop

The outer loop proceeds in three phases: (1) distances are calculated by calling the inner loop, (2) M is projected across a using the softmax function, (3) coordinate descent is used to update Y.

Therefore, using softmax, M is updated in phase two:

$$M_{ia} = \frac{\exp(-\beta_M D(X_i, Y_a))}{\sum_{a'=1}^{A} \exp(-\beta_M D(X_i, Y_{a'}))} \tag{6}$$

Y, in phase three is calculated using coordinate descent. Let y_n be the nth member of $\{Y\}$. y_n is updated such that:

$$\frac{\partial E_{cluster}}{\partial y_{an}} = 0 \tag{7}$$

Then β_M is increased and the loop repeats.

When learning prototypical point-sets, y_{an} in (6) will be either the x or y coordinate of a point in the prototype (cluster center). If weighted graphs are being learned then y_{an} will be a link in the cluster center graph. When clustering graphs, (2) is used for the distance in (6) while (3) is used to calculate y_{an} in (7). This results in a faster calculation of (6), but for (7) results in an easy analytic solution.

When analytic solutions are computed for (7) the outer loop takes a form similar to fuzzy ISODATA clustering [Duda and Hart, 1973], with annealing on the fuzziness parameter.

4 Methods and Experimental Results

Five series of experiments were run to evaluate the learning algorithms. Point sets were clustered in four experiments and weighted graphs were clustered in the fifth. In each experiment, a set of object models were used. In one experiment handwritten character data were used for the object models, in the other four experiments

the object models were randomly generated. From each object model, a set of object instances were created by transforming the object model according to the problem domain assumed for that experiment. For example, an object represented by points in two dimensional space was translated, rotated, scaled, sheared, and permuted to form a new point set. A object represented by a weighted graph was permuted. Noise was added to further distort the object. Parts of the object were deleted and spurious features (points) were added. In this manner, from a set of object models, a larger number of object instances were created. Then, with no knowledge of the original objects models or cluster memberships, we clustered the object instances using the algorithms described above.

The bulk of our experimental trials were on randomly generated patterns. However, in order to clearly demonstrate our methods and visually display our results, we will first report the results of the experiment where we used handwritten character models.

4.1 Handwritten Character Models

An X-windows tool was used to draw handwritten characters with a mouse on a writing pad. The contours of the images were discretized and expressed as a set of points in the plane. Twenty-five points for each character were used. The four characters used as models are displayed in row 1 of Figure 1. Each character model was transformed in the manner described above to create 32 character instances (128 characters for all four). Specifically (in units normalized approximately to the height of 'b' in Figure 1): $\mathcal{N}(0, .02)$ of Gaussian noise was added to each point. Each point had a 10% probability of being deleted and a 5% probability of generating a spurious point. The components of the affine transformation were selected from a uniform distribution within the following bounds; translation: $\pm.5$, rotation: $\pm 27°$, log(*scale*): $\pm \log(.7)$. log(*vertical shear*): $\pm \log(.7)$, and log(*oblique shear*): $\pm \log(.7)$. In rows 2-5 of Figure 1, 16 out of the 128 characters generated are displayed. The clustering algorithm using the affine distance measure of section 2.1 was run with the 128 characters as input and no knowledge of the cluster memberships. Figure 2 shows the results after 0, 4, 16, 64, 128 and 257 iterations of the algorithm. Note that the initial cluster center configurations (row 1 of Figure 2) were selected at random from a uniform distribution over a unit square.

4.2 Randomly Generated Models

In the next four experiments, the object models (corresponding to the models in Row 1 of Figure 1) were generated at random. The results were evaluated by comparing the object prototypes (cluster centers) formed by each experimental run to the object models used to generate the object instances for that experiment. The distance measures used in the clustering were used for this comparison, i.e. to calculate the distance between the learned prototype and the original object. This distance measure also incorporates the transformations used to create the object instances. The mean and standard deviations of these distances were plotted (Figure 3) over hundreds of trials, varying the object instance generation noise. The straight line appearing on each graph displays the effect of the Gaussian noise only. It is the expected object model-object prototype distance if no transformations were

applied, no features were deleted or added, and the cluster memberships of the object instances were known. It serves as an absolute lower bound on our learning algorithm. The noise was increased in each series of trials until the curve flattened - that is the object instances became so distorted by noise that no information about the orginal objects could be recovered by the algorithm.

In the first experiment (Figure 3a), point set objects were translated, rotated, scaled, and permuted. Initial object models were created by selecting points with a uniform distribution within a unit square. The transformations to create the object instance were selected with a uniform distribution within the following bounds; translation: $\pm.5$, rotation: $\pm 27°$, log(*scale*): $\pm \log(.5)$. For example, within these bounds the largest object instances that are generated may be four times the size of the smallest. 100 object instances were generated from 10 object models. All objects contained 20 points.The standard deviation of the Gaussian noise was varied from .02 to .16 in steps of .02. At each noise level, there were 15 trials. The data point at each error bar represents 150 distances (15 trials times 10 model-prototype distances for each trial).

In the second and third experiments (Figures 3b and 3c), point set objects were translated, rotated, scaled, sheared (obliquely and vertically), and permuted. Each object point had a 10% probability of being deleted and a 5% probability of generating a spurious point. Object points and transformations were randomly generated as in the first experiment, except for these bounds; log(*scale*): $\pm \log(.7)$, log(*vertical shear*): $\pm \log(.7)$, and log(*oblique shear*): $\pm \log(.7)$. In experiment 2, 64 object instances and 4 object models of 15 points each were used. In experiment 3, 256 object instances and 8 object models of 20 points each were used. Noise levels as in experiment 1 were used. 20 trials were run at each noise level in experiment 2 and 10 trials run at each noise level in experiment 3.

In the fourth experiment (Figure 3d), object models were represented by fully connected weighted graphs. The link weights in the initial object models were selected with a uniform distribution between 0 and 1. The objects were then randomly permuted to form the object instance and uniform noise was added to the link weights. 64 object instances were generated from 4 object models consisting of 10 node graphs with 100 links. The standard deviation of the noise was varied from .01 to .12 in steps of .01. There were 30 trials at each noise level.

In most experiments, at low noise levels ($\leq .06$ for point sets, $\leq .03$ for graphs), the object prototypes learned were very similar to the object models. As an example of what the plotted distances mean in terms of visual similarity - the average model-prototype distance in the handwritten character example (Row 1 of Figure 1 and Row 6 of Figure 2) was .5. Even at higher noise levels, object prototypes similar to the object models are formed, though less consistently. Results from about 700 experiments are plotted, which took several thousand hours of SGI R4400 workstation processor time. The objective for experiment 3 contained close to one million variables and converged in about 4 hours. The convergence times of the objectives of experiments 1,2, and 4 were 120, 40 and 10 minutes respectively. Each point set trial was a best of four run, in which the object models and object instances were the same for each run, but the initial randomly selected starting cluster centers (Row 1 of Figure 2) were varied and the run with the lowest ending energy was choosen.

5 Conclusions

It has long been argued by many, that learning in complex domains typically associated with human intelligence requires some type of prior structure or knowledge. We have begun to develop a set of tools that will allow the incorporation of prior structure within learning. Our models incorporate many features needed in complex domains like vision - noise, missing and spurious features, non-rigid transformations. They can learn objects with inherent structure, like graphs. Many experiments have been run on experimentally generated data sets. Several directions for future research hold promise. One might be the learning of OCR data. Secondly, a supervised learning stage could be added to our algorithms. Finally the power of the distance measures can be enhanced to operate on graphs with deleted or missing nodes, attributed nodes and multi-level (part-whole) structures.

Acknowledgements

This work has been supported by AFOSR grant F49620-92-J-0465 and ONR/DARPA grant N00014-92-J-4048 and the Yale Center for Theoretical and Applied Neuroscience.

References

E. Bienenstock, and R. Doursat. (1991). Issues of representation in neural networks. In A. Gorea, (ed.), *Representations of Vision: Trends and Tacit Assumptions in Vision Research.* Cambridge: Cambridge University Press.

R. Duda and P. Hart. (1973). *Pattern Classification and Scene Analysis.* New York, NY: Wiley.

S. Geman, E. Bienenstock, and R. Doursat. (1992). Neural networks and the bias/variance dilemma. *Neural Computation*, 4:1-58.

S. Gold, E. Mjolsness and A. Rangarajan. (1994). Clustering with a domain-specific distance measure. In Cowan, J.D., Tesauro, G. and Alspector, J. (eds.), *Advances in Neural Information Processing Systems 6.* San Francisco, CA: Morgan Kaufmann Publishers.

S. Gold, C. P. Lu, A. Rangarajan, S. Pappu and E. Mjolsness. (1994). New algorithms for 2D and 3D point matching: pose estimation and correspondence. YALEU/DCS/TR-1035, Yale University, Department of Computer Science.

R. Hathaway. (1986). Another interpretation of the EM algorithm for mixture distributions. *Statistics and Probability Letters*, 4:53-56.

M. Jordan and R. Jacobs. (1994). Hierarchical mixtures of experts and the EM algorithm. *Neural Computation*, 6:181-214.

C. P. Lu and E. Mjolsness. (1994). Two-dimensional object localization by coarse-to-fine correlation matching. In Cowan, J.D., Tesauro, G. and Alspector, J. (eds.), *Advances in Neural Information Processing Systems 6.* San Francisco, CA: Morgan Kaufmann Publishers.

C. von der Malsburg. (1988). Pattern recognition by labeled graph matching.

Neural Networks, **1**:141-148.

J. Martin. (1991). Coding and processing of sensory information. In Kandel, E., Schwartz, J. and Jessell, T. (eds.), *Principles of Neural Science*, 3rd edition. New York, NY: Elsevier.

E. Mjolsness and C. Garrett. (1990). Algebraic transformations of objective functions. *Neural Networks*, **3**:651-669.

E. Mjolsness and W. Miranker. (1993). Greedy Lagrangians for neural networks: three levels of optimization in relaxation dynamics. Technical Report 945, Yale University, Department of Computer Science.

E. Mjolsness. (1992). Visual grammars and their neural networks. *SPIE Conference on the Science of Artificial Neural Networks*, **1710**:63-85.

C. Peterson and B. Söderberg. (1989). A new method for mapping optimization problems onto neural networks. *International Journal of Neural Systems*, **1**(1):3-22.

A. Rangarajan, S. Gold and E. Mjolsness. (1994). A novel optimizing network architecture with applications. YALEU/DCS/TR-1036, Yale University, Department of Computer Science.

A. Rangarajan and E. Mjolsness. (1994). A Lagrangian relaxation network for graph matching. In *Proc. ICNN '94*. IEEE Press.

R. Shepard. (1989). Internal representation of universal regularities: A challenge for connectionism. In L. Nadel, L. Cooper, P. Culicover and R. Harnish, (eds.), *Neural Connections, Mental Computation*. Cambridge, MA, London, England: Bradford/MIT Press.

E. Saund. (1994). Unsupervised learning of mixtures of multiple causes in binary data. In Cowan, J.D., Tesauro, G. and Alspector, J. (eds.), *Advances in Neural Information Processing Systems 6*. San Francisco, CA: Morgan Kaufmann Publishers.

J. Utans. (1994). Learning in compositional hierarchies: inducing the structure of objects from data. In Cowan, J.D., Tesauro, G. and Alspector, J. (eds.), *Advances in Neural Information Processing Systems 6*. San Francisco, CA: Morgan Kaufmann Publishers.

C. Williams, R. Zemel, and M. Mozer. (1993). Unsupervised learning of object models. AAAI Technical Report FSS-93-04, University of Toronto, Department of Computer Science.

A. Yuille and J. Kosowsky. (1994). Statistical physics algorithms that converge. *Neural Computation*, **6**:341-356.

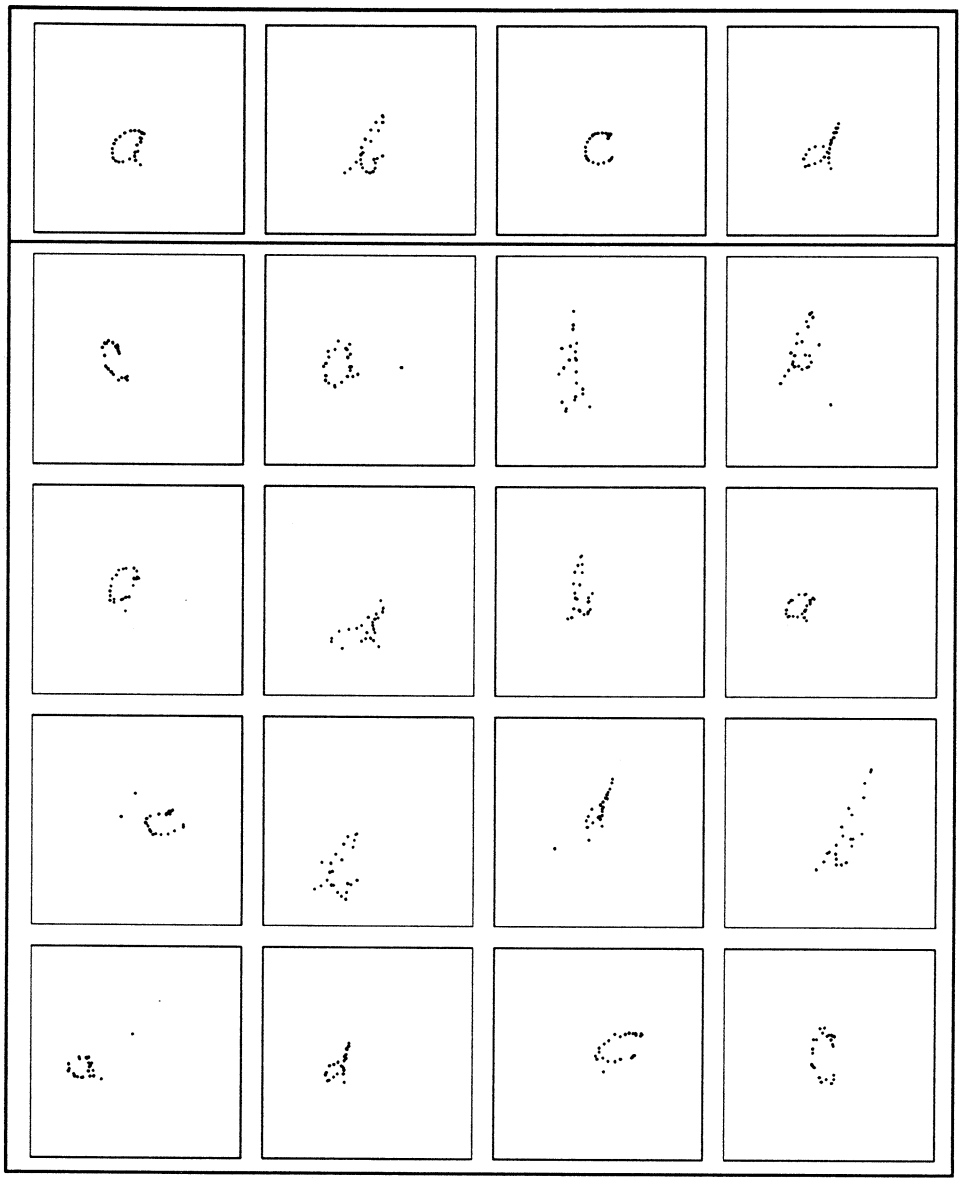

Figure 1: Row (1): Handwritten character models used to generate character instances. These models were not part of the input to the clustering algorithm. Rows (2-5): 16 character instances which (with 112 other characters) were clustered.