Neural Computation, in press, 8(4):???-???, 1996

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

Steven Gold
Department of Computer Science
Yale University
New Haven, CT 06520-8285
e-mail: gold-steven@cs.yale.edu

Anand Rangarajan
Dept. of Diagnostic Radiology
Yale University
New Haven, CT 06520-8042
e-mail: anand@noodle.med.yale.edu

Eric Mjolsness
Department of Computer Science and Engineering
University of California at San Diego (UCSD)
La Jolla, CA 92093-0114
e-mail: emj@cs.ucsd.edu

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 approx. 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—softassign, algebraic transformations, clocked objectives, and deterministic annealing.

1 Introduction

While few biologists today would subscribe to Locke's description of the nascent 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 minds may incorporate preknowledge in the form of distance measures (Shepard, 1989; Bienenstock and Doursat, 1991). The neural network community has begun to explore this idea via tangent distance (Simard et al., 1993) and model learning (Williams et al., 1993). However neither of these distance measures have been invariant under permutation of the labeling of the feature points or nodes. Permutation invariant distance measures must solve the correspondence problem, a computationally intractable problem fundamental to object recognition systems (Grimson, 1990). Such distance measures may be better suited for the learning of the higher level, more complex representations needed for cognition. In this work, we introduce the use of more powerful, permutation invariant distance measures in learning.

The unsupervised learning of object prototypes from collections of noisy 2-D point-sets or noisy weighted graphs is achieved by clustering with point matching and graph matching distance measures. The point matching measure is approx. 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—softassign, algebraic transformations, clocked objectives, and deterministic annealing. Combining these techniques significantly increases the size of problems which may be solved with recurrent network architectures (Rangarajan et al.,). Even on single-processor workstations, non-linear objectives with a million variables can be minimized relatively quickly (a few hours). With these methods we learn prototypical examples of 2-D points-set and graphs from randomly generated experimental data.

2 Relationship to Previous Clustering Methods

Clustering algorithms may be classified as central or pair-wise (Buhmann and Hofmann, 1994). Central clustering algorithms generally use a distance measure, like Euclidean or Mahalonobis, which operates on feature vectors within a pattern matrix (Jain and Dubes, 1988; Duda and Hart, 1973). These algorithms calculate cluster centers (pattern prototypes) and compute the distances between patterns within a cluster and the cluster center (i.e. pattern–cluster center distances). Pair-wise clustering algorithms, in contrast may use only the distances between patterns and may operate on a proximity matrix (a pre-computed matrix containing all the distances between every pair of patterns). Pair-wise clustering algorithms need not produce a cluster center and do not have to re-calculate distance measures during the algorithm.

We introduce central clustering algorithms that employ higher-level distance measures. In the few cases where higher-level distance measures have been used in clustering (Kurita et al., 1994) they have all to our knowledge been employed in pair-wise clustering algorithms, which used pre-computed proximity matrices and did not calculate prototypes. Consequently, while classification was learned, exemplars were not.

As is the case for central clustering algorithms, the algorithm employed here tries to minimize the cluster center–cluster member distances. However, because it uses complex distance measures it has an outer and inner loop. The outer loop uses the current values of the cluster center–cluster member distances to recompute assignments (reclassify). After reclassification, the inner loop recomputes the distance measures. The outer loop is similar to several other algorithms employing mean field approximations for clustering (Rose et al., 1990; Buhmann and Kuhnel, 1993). It is also similar to fuzzy ISODATA clustering (Duda and Hart, 1973), with annealing on the fuzziness parameter. The clustering algorithm used here is formulated as a combinatorial optimization problem, however it may also be related to parameter estimation of mixture models using the maximum likelihood method (Duda and Hart, 1973) and the expectation-maximization (EM) algorithm (Dempster et al., 1977; Hathaway, 1986). The inner loop uses the newly discovered distance measures for point (Gold et al., 1995) and graph matching. In the following we will first describe these new distance measures and then show how they are incorporated in the rest of the algorithm.

3 Formulation of the Objective Functions

3.1 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 some points of X onto some points of Y:

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

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

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

A is the affine transformation which is decomposed into scale, rotation, and two components of 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 possibly 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. (Note: simplex constraints on m, and its linear appearance in $E(m)$, imply that any local minimum of (m, A, t) occurs at a vertex in the m simplex. But m's trajectory can use the interior of the m simplex to avoid local minima in the optimization of A and t.)

The α term biases the objective towards matches. The decomposition of A in the above is not required, since A could be 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, using the singular value decomposition of A.

Then given two sets of points $\{X_j\}$ and $\{Y_k\}$ the distance between them may be 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 from Mean Field Theory assumptions in (Mjolsness, 1993):

$$D(x,y) = -\log \frac{\Pr(x|y)}{\max_x \Pr(x|y)} \approx \min_T d(x, T(y))$$

where

$$d(x,y) = -\log \frac{\hat{\Pr}(x|y)}{\max_x \hat{\Pr}(x,y)}$$

and T is a set of transformation parameters introduced by a visual grammar (Mjolsness, 1994) and $\hat{\Pr}$ is the probability that x arises from y without transformations T.

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

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

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 Soderberg, 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; Rangarajan et al., ; Mjolsness and Garrett, 1990):

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

In this objective, we are looking for a saddle point. Equation (2) 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 on β.

The above defines a series of distance measures, since given the decomposition of A it is trivial to construct measures which are approx. invariant only under some subset of the transformations (such as rotation and translation). The regularization, $g(A)$, and α terms may also be individually adjusted in an appropriate fashion for a specific problem domain. For example, replacing A with $R(\Theta)$ in (1) and removing $g(A)$ would define a new distance measure which is only invariant under rotation and translation.

3.2 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_{kp}, one can minimize the objective below to find the permutation which best maps G onto g (Rangarajan and Mjolsness, 1994; von der Malsburg, 1988; Hopfield and Tank, 1986):

$$E_{gm}(m) = \sum_{j=1}^{J}\sum_{k=1}^{K}(\sum_{l=1}^{L} G_{jl}m_{lk} - \sum_{p=1}^{P} m_{jp}g_{pk})^2$$

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 zero or one. This term (with the γ parameter below) is similarly transformed with a reversed neuron. The resulting objective is:

$$\begin{aligned} E_{gm}(m) \;=\; & \sum_{j=1}^{J}\sum_{k=1}^{K}\mu_{jk}(\sum_{l=1}^{L} G_{jl}m_{lk} - \sum_{p=1}^{P} m_{jp}g_{pk}) - \frac{1}{2}\sum_{j=1}^{J}\sum_{k=1}^{K}\mu_{jk}^2 \\ & -\gamma\sum_{j=1}^{J}\sum_{k=1}^{K}\sigma_{jk}m_{jk} + \frac{\gamma}{2}\sum_{j=1}^{J}\sum_{k=1}^{K}\sigma_{jk}^2 \\ & +\frac{1}{\beta}\sum_{j=1}^{J}\sum_{k=1}^{K} m_{jk}(\log m_{jk} - 1) \\ & +\sum_{j=1}^{J}\kappa_j(\sum_{k=1}^{K} m_{jk} - 1) + \sum_{k=1}^{K}\lambda_k(\sum_{j=1}^{J} m_{jk} - 1) \end{aligned} \tag{3}$$

As in section 2.1, we look for a saddle point. Equation (3) 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):

$$E_{gm'}(m) = \sum_{j=1}^{J}\sum_{l=1}^{L}\sum_{k=1}^{K}\sum_{p=1}^{P} m_{jk}m_{lp}(G_{jl} - g_{kp})^2 \tag{4}$$

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

3.3 The Clustering Objective

The object learning problem is formulated as follows: Given a set of I noisy observations $\{X_i\}$ of some unknown objects, find a set of B cluster centers $\{Y_b\}$ and match variables $\{M_{ib}\}$ defined as

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

such that each observation is in only one cluster, and the total distance of all the observations from their respective cluster centers is minimized. The cluster centers are the learned approximations of the original objects. To find $\{Y_b\}$ and $\{M_{ib}\}$ minimize the cost function,

$$E_{cluster}(Y, M) = \sum_{i=1}^{I} \sum_{b=1}^{B} M_{ib} D(X_i, Y_b)$$

with constraints: $\forall i \ \sum_b M_{ib} = 1, \ \ \forall ib \ M_{ib} \geq 0$. $D(X_i, Y_b)$, the distance function, is a measure of dissimilarity between two objects. This problem formulation may be derived from Bayesian inference of a set of object models $\{Y\}$ from the data $\{X\}$ they explain (Mjolsness, 1993). It is also a clustering objective with a domain-specific distance measure (Gold et al., 1994).

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:

$$\begin{aligned} E_{cluster}(Y, M) \ &= \ \sum_{i=1}^{I} \sum_{b=1}^{B} M_{ib} D(X_i, Y_b) + \frac{1}{\beta_M} \sum_{i=1}^{I} \sum_{b=1}^{B} M_{ib}(\log M_{ib} - 1) \\ &+ \sum_{i=1}^{I} \lambda_i (\sum_{b=1}^{B} M_{ib} - 1) \end{aligned} \tag{5}$$

Here, the objects are point-sets or weighted graphs. If point-sets are used, the distance measure $D(X_i, Y_b)$ is replaced by (1); if graphs it is replaced by (3), without the terms that enforce the constraints, or (4). For example, after replacing the distance measure by (1), we obtain:

$$\begin{aligned} E_{cluster}(Y, M, t, A, m) = \sum_{i=1}^{I} \sum_{b=1}^{B} M_{ib} \left[\sum_{j=1}^{J} \sum_{k=1}^{K} m_{ibjk} (\| X_{ij} - t_{ib} - A_{ib} Y_{bk} \|^2 - \alpha) + g(A_{ib}) \right] \\ + \sum_{i=1}^{I} \sum_{b=1}^{B} \Big[\frac{1}{\beta_m} \sum_{j=1}^{J+1} \sum_{k=1}^{K+1} m_{ibjk} (\log m_{ibjk} - 1) + \sum_{j=1}^{J} \mu_{ibj} (\sum_{k=1}^{K+1} m_{ibjk} - 1) \\ + \sum_{k=1}^{K} \nu_{ibk} (\sum_{j=1}^{J+1} m_{ibjk} - 1) \Big] + \frac{1}{\beta_M} \sum_{i=1}^{I} \sum_{b=1}^{B} M_{ib} (\log M_{ib} - 1) + \sum_{i=1}^{I} \lambda_i (\sum_{b=1}^{B} M_{ib} - 1) \end{aligned} \tag{6}$$

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 other affine transformations. It is maximized with respect to λ, which enforces the row constraints 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 b it falls in, and m_{ib} is a permutation matrix which assigns to each point in cluster center Y_b a corresponding point in observation X_i. (A_{ib}, t_{ib}) gives the affine transform between object i and cluster center b. Both M and m are fuzzy, so a given object may partially fall in several clusters, with the degree of fuzziness depending on β_m and β_M.

Therefore, given a set of observations, 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 (6) may be constructed using the graph matching distance measure in (3) or (4) instead.

4 The Algorithm

4.1 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 (2) or (3)) and an outer loop to minimize the clustering objective (5). 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, 1994). 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. This is described as a softassign (Gold et al., 1995; Gold and Rangarajan, 1995; Rangarajan et al.,) (see section 4.2). The minimization of m, and the distance measure variables (either t, A of (2) or μ, σ of (3)), 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. Some 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 (Mjolsness and Miranker, 1993), which is optimized over distinct sets of variables in phases, as (letting $\mathcal{D}$ be the set of distance measure variables (e.g. $\{A,t\}$ for (2)) excluding the match matrix),

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

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 $\mathcal{D}$ to the equivalent of an identity transform, Y to random values

Begin Outer Loop
  Begin Inner Loop
    Initialize $\mathcal{D}$ with previous values
    Find m, $\mathcal{D}$ for each ib pair :
      Find m by softassign
      Find $\mathcal{D}$ by coordinate descent
  End Inner Loop
  If first time through outer loop increase β_m and repeat inner loop
  Find M,Y using fixed values of m, $\mathcal{D}$, determined in inner loop:
      Find M by softmax, across i
      Find Y by coordinate descent
  increase β_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 $\mathcal{D}$ (namely 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 ib pair when its M_{ib} membership variable drops below a threshold.

4.2 Inner Loop

The inner loop proceeds in two phases. In phase one, while $\mathcal{D}$ are held fixed, m is initialized with a coordinate descent step, described below, and then iteratively normalized across its rows and columns until the procedure converges (Kosowsky and Yuille, 1994). This phase is analogous to a softmax update, except that instead of enforcing a one-way, winner-take-all (maximum) constraint, a two-way, assignment constraint is being enforced. Therefore we describe this phase as a softassign (Gold et al., 1995; Gold and Rangarajan, 1995; Rangarajan et al.,). In phase two m is held fixed and $\mathcal{D}$ are updated using coordinate descent. Then β_m is increased and the loop repeats. Let E_{dmwc} be the distance measure objective [(2) or (3)] 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 a softassign, which consists of a coordinate descent update:

$$m_{ibjk} = \exp(-\beta_m \partial E_{dmwc}(X_i, Y_b)/\partial m_{ibjk})$$

And then (also as part of the softassign) m is iteratively normalized across j and k until $\sum_{j=1}^{J}\sum_{k=1}^{K} \Delta m_{ibjk} < \epsilon$:

$$m_{ibjk} = \frac{m_{ibjk}}{\sum_{j'=1}^{J+1} m_{ibj'k}} \quad ; \quad m_{ibjk} = \frac{m_{ibjk}}{\sum_{k'=1}^{K+1} m_{ibjk'}}$$

Using coordinate descent, the $\mathcal{D}$ are updated in phase two. If a member of $\mathcal{D}$ 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 $\mathcal{D}$ then in phase two we update d_{ibn} such that:

$$\frac{\partial E_{dmwc}(X_i, Y_b)}{\partial d_{ibn}} \approx 0$$

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 (Rangarajan et al.,).

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.

4.3 Outer Loop

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

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

$$M_{ib} = \frac{\exp(-\beta_M D(X_i, Y_b))}{\sum_{b'=1}^{B} \exp(-\beta_M D(X_i, Y_{b'}))} \tag{7}$$

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

$$\frac{\partial E_{cluster}}{\partial y_{bn}} = 0 \tag{8}$$

Then β_M is increased and the loop repeats.

When learning prototypical point-sets, y_{bn} in (7) will be either the x or y coordinate of a point in the prototype (cluster center). If weighted graphs are being learned then y_{bn} will be a link in the cluster center graph. When clustering graphs, (3) is used for the distance in (7) while (4) is used to calculate y_{bn} in (8). This results in a faster calculation of (7), but for (8) results in an easy analytic solution.

5 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 and in the other experiments the object models were randomly generated. From each object model, a set of object observations 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. An object represented by a weighted graph was permuted. Independent noise of known variance was added to all real-valued parameters 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 object 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 in which we used handwritten character models.

5.1 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 $\log(oblique\ shear)$: $\pm\log(.7)$. Note in equation (1) of section 3.1, $a = \log(scale)$, $b = \log(vertical\ shear)$ and $c = \log(oblique shear)$. 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 256 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. The original models were reconstructed to high accuracy from the data, up to affine transformations within the allowed ranges.

5.2 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 the accuracy of our learning algorithm. The variance of the real-valued parameter 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 original 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 (both components), 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. In these experiments the temperature parameter of the inner loop equaled the temperature parameter of the outer loop ($\beta_m = \beta_M$) and both were increased by a factor of 1.03 on each iteration of the outer loop. In the point set experiments, each trial was a best of four series. The object models and object instances were the same for each of the four executions within the trial, but the initial randomly selected starting cluster centers (Row 1 of Figure 2) were varied for each execution and only the result from the execution with the lowest ending energy was reported.

The time for recognition, which simply involved running the distance measures alone, was at most a few seconds for the largest point sets which contained twenty-five points.

6 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: parameter 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, i.e. we may include some prior knowledge regarding the classification or labeling of our objects. While the experiments in this paper incorporated only a few missing points within the object sets, the point matching distance measures are capable of matching objects arising from real image data with large amounts of occlusion and with feature points that do not necessarily lie in one-to-one correspondence with each other as did the artificially generated point sets of this paper (Gold et al., 1995). Supervised learning algorithms may be better able to exploit the power of these distance measures. Finally, more powerful, recently developed graph-matching distance measures (Gold and Rangarajan, 1995) may be used which are able to operate on graphs with attributed nodes, multiple link types and deleted or missing nodes and links.

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

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

Buhmann, J. and Hofmann, T. (1994). Central and pairwise data clustering by competitive neural networks. In Cowan, J., Tesauro, G., and Alspector, J., editors, *Advances in Neural Information Processing Systems 6*, pages 104–111. Morgan Kaufmann, San Francisco, CA.

Buhmann, J. and Kuhnel, H. (1993). Complexity optimized data clustering by competitive neural networks. *Neural Computation*, 5(1):75–88.

Dempster, A. P., Laird, N. M., and Rubin, D. B. (1977). Maximum likelihood from incomplete data via the EM algorithm. *J. R. Statist. Soc. Ser. B*, 39:1–38.

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

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

Gold, S., Lu, C. P., Rangarajan, A., Pappu, S., and Mjolsness, E. (1995). New algorithms for 2-D and 3-D point matching: pose estimation and correspondence. In Tesauro, G., Touretzky, D. S., and Leen, T. K., editors, *Advances in Neural Information Processing Systems 7*, pages 957–964. MIT Press, Cambridge, MA.

Gold, S., Mjolsness, E., and Rangarajan, A. (1994). Clustering with a domain specific distance measure. In Cowan, J., Tesauro, G., and Alspector, J., editors, *Advances in Neural Information Processing Systems 6*, pages 96–103. Morgan Kaufmann, San Francisco, CA.

Gold, S. and Rangarajan, A. (1995). A graduated assignment algorithm for graph matching. Technical Report YALEU/DCS/RR-1062, Department of Computer Science, Yale University.

Grimson, E. (1990). *Object Recognition by Computer: The Role of Geometric Constraints.* MIT Press, Cambridge, MA.

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

Hopfield, J. J. and Tank, D. W. (1986). Collective computation with continuous variables. In *Disordered Systems and Biological Organization*, pages 155–170. Springer–Verlag.

Jain, A. K. and Dubes, R. C. (1988). *Algorithms for Clustering Data.* Prentice Hall, Englewood Cliffs, NJ.

Jordan, M. I. and Jacobs, R. A. (1994). Hierarchical mixtures of experts and the EM algorithm. *Neural Computation*, 6(2):181–214.

Kosowsky, J. J. and Yuille, A. L. (1994). The invisible hand algorithm: Solving the assignment problem with statistical physics. *Neural Networks*, 7(3):477–490.

Kurita, T., Sekita, I., and Otsu, N. (1994). Invariant distance measures for planar shapes based on complex autoregressive model. *Pattern Recognition*, 27(7):903–911.

Mjolsness, E. (1993). Bayesian inference on visual grammars by relaxation nets. Unpublished manuscript.

Mjolsness, E. (1994). Connectionist grammars for high-level vision. In Honavar, V. and Uhr, L., editors, *Artificial Intelligence and Neural Networks: Steps Toward Principled Integration*, pages 423–451. Academic Press, Inc., San Diego, CA.

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

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

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

Rangarajan, A., Gold, S., and Mjolsness, E. A novel optimizing network architecture with applications. *Neural Computation*, (in press), 1996.

Rangarajan, A. and Mjolsness, E. (1994). A Lagrangian relaxation network for graph matching. In *IEEE Intl. Conf. on Neural Networks (ICNN)*, volume 7, pages 4629–4634. IEEE Press.

Rose, K., Gurewitz, E., and Fox, G. (1990). Statistical mechanics and phase transitions in clustering. *Physical Review Letters*, 65(8):945–948.

Shepard, R. (1989). Internal representation of universal algorithms: A challenge for connectionism. In Nadel, L., Cooper, L., Culicover, P., and Harnish, R., editors, *Neural Connections, Mental Computation*, pages 104–134. Bradford/MIT Press, Cambridge, MA.

Simard, P., le Cun, Y., and Denker, J. (1993). Efficient pattern recognition using a new transformation distance. In Hanson, S., Cowan, J., and Giles, C., editors, *Advances in Neural Information Processing Systems 5*, pages 50–58. Morgan Kaufmann, San Mateo, CA.

von der Malsburg, C. (1988). Pattern recognition by labeled graph matching. *Neural Networks*, 1:141–148.

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

Yuille, A. L. and Kosowsky, J. J. (1994). Statistical physics algorithms that converge. *Neural Computation*, 6(3):341–356.

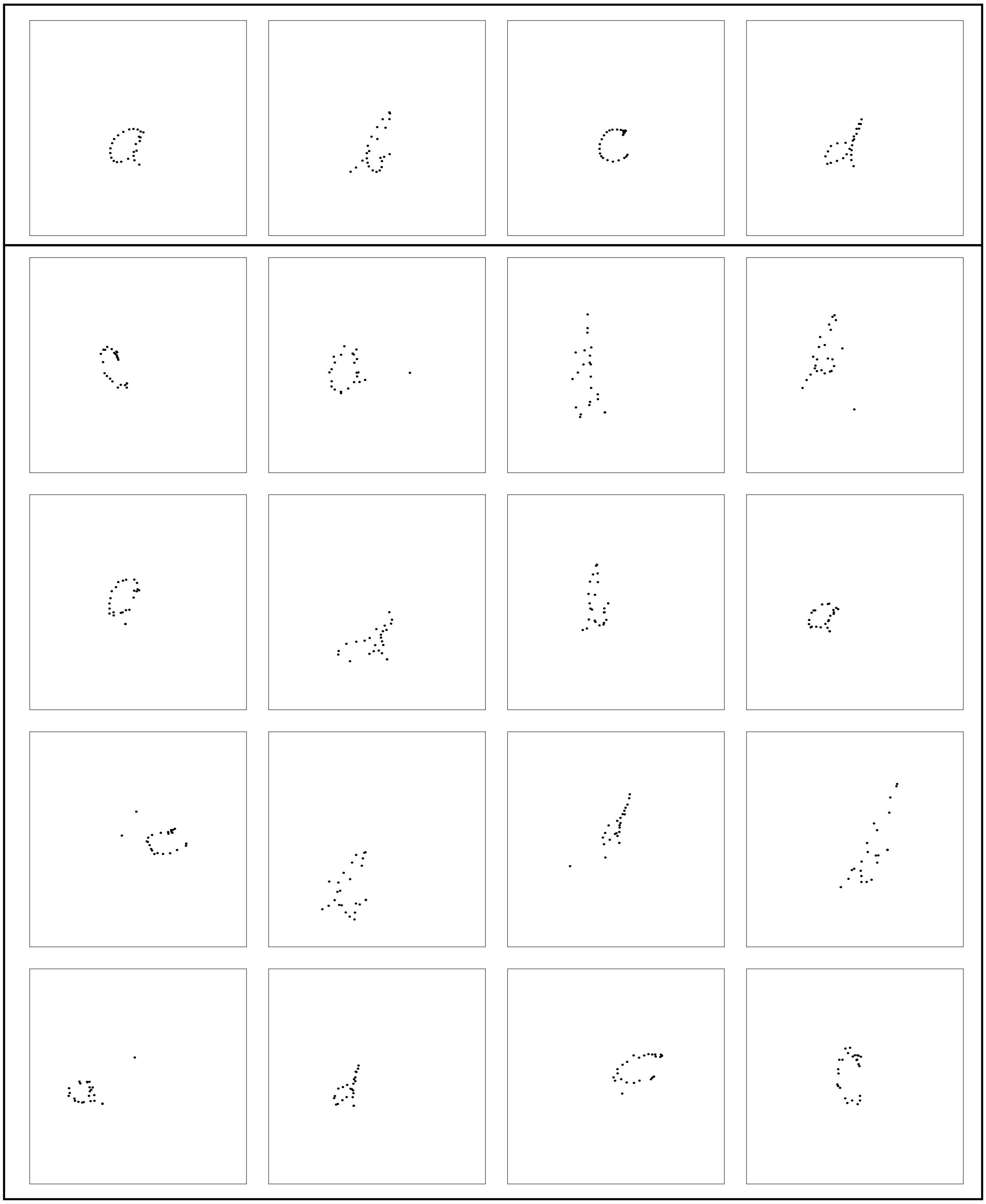

Figure 1: Row (1): Handwritten character models used to generate character instances. These models were not part of the input to the clustering algorithm. Rows (2-5): 16 character instances which (with 112 other characters) were clustered.

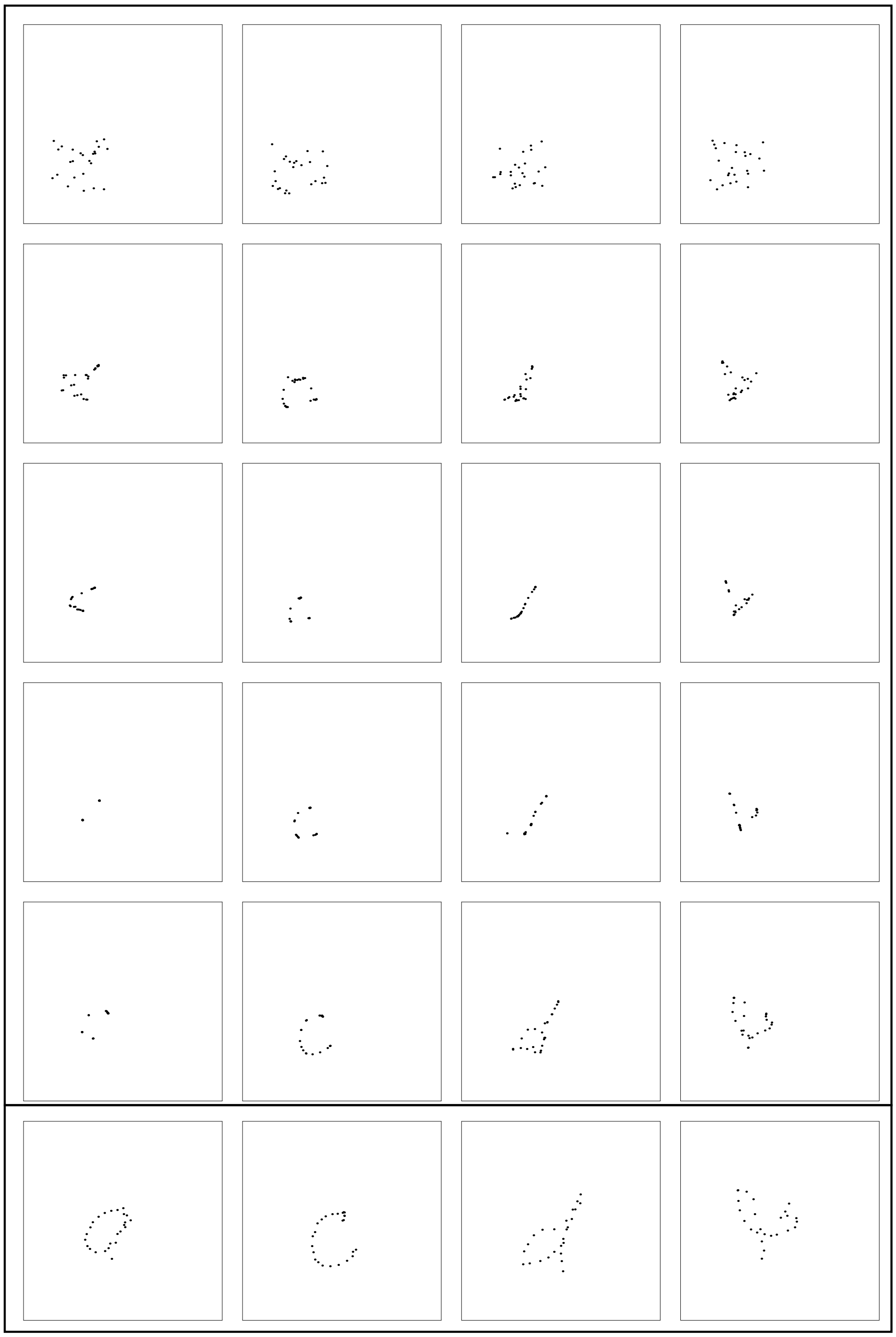

Figure 2: Row (1): initial cluster centers (randomly generated). Rows (2-6): character prototypes (cluster centers) after 4, 16, 64, 128 and 256 iterations.

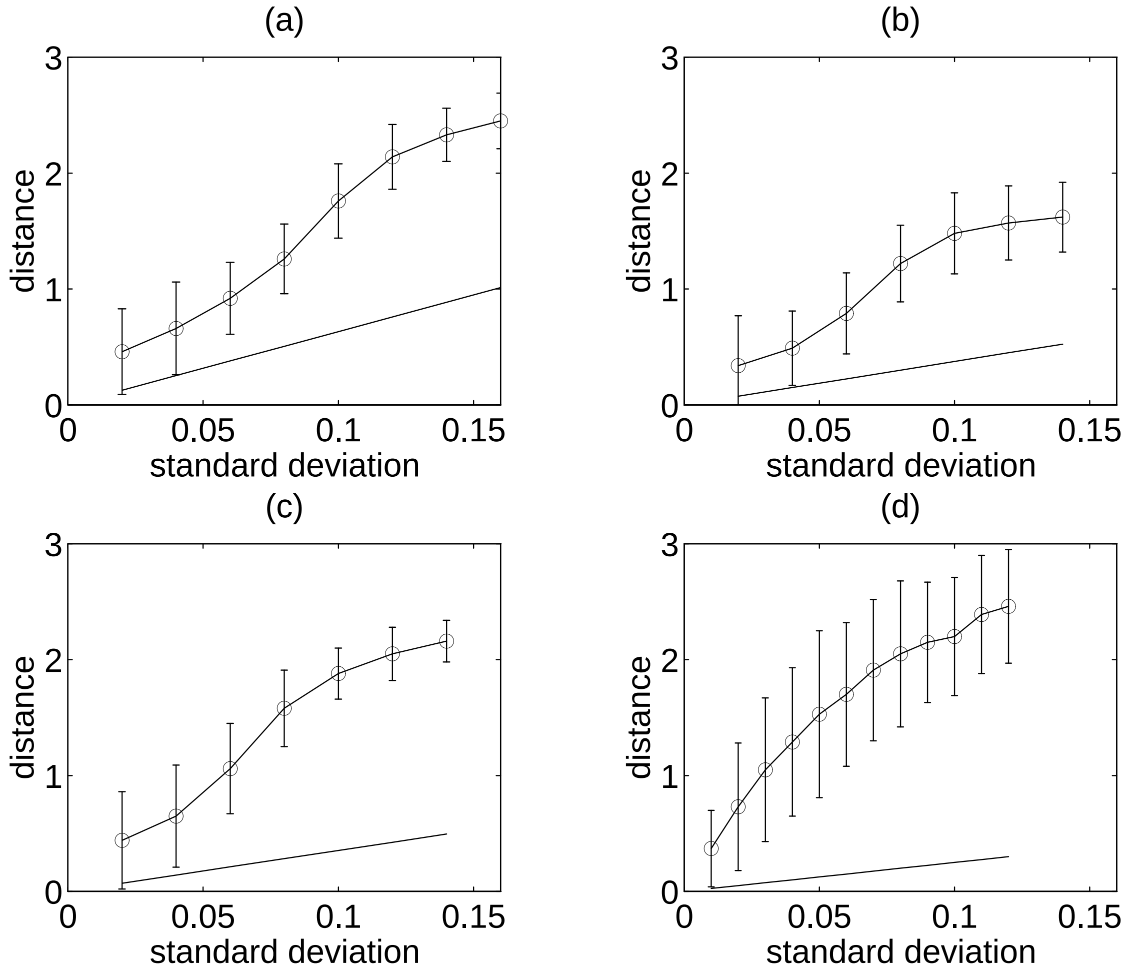

Figure 3: (a): 10 clusters, 100 point sets, 20 points each, scale, rotation, translation, 120 trials. (b): 4 clusters, 64 point sets, 15 points each, affine, 10 % deleted, 5 % spurious, 140 trials. (c): 8 clusters, 256 point sets, 20 points each, affine, 10 % deleted, 5 % spurious, 70 trials. (d): 4 clusters, 64 graphs, 10 nodes each, 360 trials.