Scalable Action Respecting Embedding

Michael Biggs and **Ali Ghodsi**
Department of Statistics and Actuarial Science
University of Waterloo
Waterloo, ON, Canada

Dana Wilkinson
David R. Cheriton School of Computer Science
University of Waterloo
Waterloo, ON, Canada

Michael Bowling
Department of Computing Science
University of Alberta
Edmonton, AB, Canada

Abstract

ARE is a non-linear dimensionality reduction technique for embedding observation trajectories, which captures state dynamics that traditional methods do not. The core of ARE is a semidefinite optimization with constraints requiring actions to be distance-preserving in the resulting embedding. Unfortunately, these constraints are quadratic in number and non-local (making recent scaling tricks inapplicable). Consequently, the original formulation was limited to relatively small datasets. This paper describes two techniques to mitigate these issues. We first introduce an action-guided variant of Isomap. Although it alone does not produce action-respecting manifolds, it can be used to seed conjugate gradient to implicitly solve the primal variable formulation of the ARE optimization. The optimization is not convex, but the Action-Guided Isomap provides an excellent seed often very close to the global minimum. The resulting Scalable ARE procedure gives similar results to original ARE, but can be applied to datasets an order of magnitude larger.

1 Introduction

Mapping is an important problem in a variety of areas in artificial intelligence (e.g., robotics, reinforcement learning, multi-agent games). A general description of the problem of mapping is to learn a useful representation of an environment from a stream of interactive experience—usually a collection of high-dimensional observation vectors interleaved with the actions that result in them.

One approach to this problem is to focus on the features of a representation that make it good or useful. One desirable feature is that it should be low-dimensional while still being sufficient for describing the observations. A second is that the actions associated with the environment should correspond to simple transformations in the representation. The first property corresponds to the well-known dimensionality reduction problem—indicating that good representations may be learned by modified dimensionality reduction algorithms. One such algorithm, Action Respecting Embedding (ARE), was derived from the Maximum Variance Unfolding algorithm and has been shown to learn very promising representations on relatively small streams of actions and observations. As the ARE algorithm is based on a semidefinite optimization, however, there are problems scaling it up to handle larger streams of experience.

In this paper, alternative procedures are developed for Action Respecting Embedding which are better suited to handling large datasets. The scaling issue is addressed through two new ideas. First, modifications are made to the well-known Isomap algorithm to make use of knowledge of actions. Although the actions can't impose hard constraints as they do with ARE, they do act as a guiding force. The new algorithm, Action-Guided Isomap, is only marginally successful at learning representations by itself, but it turns out to be more useful in combination with the second idea. A primal formulation of ARE is derived which attempts to directly learn the low-dimensional coordinates of the data points in the representation. This primal optimization no longer has a semidefinite constraint admitting the use of much speedier conjugate gradient solvers. Unfortunately, this new optimization is non-convex and so conjugate gradient solvers will also be subject to local minima. The primal optimization, though, can still be effective if it has a good starting seed. Hence, this optimization is coupled with Action-Guided Isomap for seeding. In practice this combination produces results on par with the original ARE algorithm. However, the new algorithm can handle data sets an order of magnitude larger.

After a review of Isomap, MVU and ARE in Section 2, Action-Guided Isomap is developed in Section 3. The primal version of ARE based on Procrustes operators is presented in Section 4. Section 5 shows comparisons of the new and old versions of ARE on smaller datasets as well as results of the new algorithm on much larger datasets while Section 6 concludes.

2 Background

This section reviews the Isomap and Maximum Variance Unfolding (MVU) techniques for dimensionality reduction as well as the recent Action Respecting Embedding algorithm for learning subjective representations. A classical technique for dimensionality reduction is Principal Components Analysis (PCA). It finds a linear subspace of a high-dimensional input space that preserves the most variance in the data points. This can be extended via the well known "kernel trick" to find non-linear subspaces by effectively performing the dual of PCA in an inner product

space defined by some kernel K. The construction of K can be specified in advance (e.g., polynomial or RBF kernels) or an attempt can be made to learn K from the initial data points. This second approach is the one taken in our approach and the dimensionality reduction methods summarized here. For more details on PCA and kernel PCA see, for example, (Scholkopf & Smola 2002; Shawe-Taylor & Cristianini 2004).

2.1 Isomap

Isomap (Tenenbaum, de Silva, & Langford 2000) is a nonlinear dimensionality reduction technique which derives a kernel matrix designed to preserve certain local distances while using estimates of geodesic distance for the remaining distances. Isomap is similar to MDS, a common linear dimensionality reduction method which finds an embedding that best preserves distances between the high-dimensional points—usually provided in the form of a matrix D (Cox & Cox 2000). Isomap also takes a matrix D and first identifies a graph of local distances by running k-nearest-neighbors. The distances between neighbors are kept the same but non-neighbors have their distances replaced with the length of the shortest path between those two points in the neighborhood graph. This graph distance is intended to be an approximation of the true geodesic ("on the manifold") distance between points. In fact, as the density of the points increases the difference between the actual geodesic and the graph distance provably converges. Once D has been updated, it is converted into a kernel matrix K which can be used with kernel PCA to obtain an embedding.

2.2 Maximum Variance Unfolding

Maximum variance unfolding (MVU) (Weinberger, Sha, & Saul 2004) learns a kernel from data by solving a semidefinite program. As long as the learned matrix satisfies the positive semidefinite constraint then it is a valid kernel for kernel PCA which can be applied to extract the low-dimensional embedding. In addition to the semidefinite constraint, neighborhood constraints are used to preserve certain local distances, which, as with Isomap, are the neighbors in a k-nearest-neighbor graph. The objective function maximizes the trace of the kernel matrix which ensures maximal variance in the resulting embedding. This has the effect of pushing points as far apart as possible (subject to the distance constraints).

2.3 Action Respecting Embedding

Action Respecting Embedding (ARE) (Bowling, Ghodsi, & Wilkinson 2005; Bowling, Wilkinson, & Ghodsi 2006) is based on the observation that maps involve compression of information while ensuring actions have a consistent effect. Consequently, ARE is posed as a variation on standard dimensionality reduction. Instead of just a set of high-dimensional input vectors the algorithm receives a sequence of high-dimensional vectors, or observations, interleaved with discrete action labels that affected the environment's dynamics between each observation. The task is to learn an embedding that serves as a useful representation of the environment, both in terms of observations and actions. Not only must this embedding involve dimensionality reduction to capture sufficient statistics for predicting observations, it must also be one in which there exist consistent transformations that correspond to the action labels.

Algorithm 1 ARE

Require: n high-dimensional data points $z_i \in \mathbb{R}^d$
U a set of distinct action labels
$n-1$ action labels $u_i \in U$ where u_i is the action between z_i and z_{i+1}
distance metric $d(x,y) \approx \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}$

Create non-uniform neighborhood graph, η.

Maximize $\mathrm{Tr}(K)$ subject to:

$$
\begin{aligned}
& K \succeq 0, \\
& \sum_{ij} K_{ij} = 0, \\
\forall ij \quad & \eta_{ij} > 0 \vee [\eta^T \eta]_{ij} > 0 \\
\Rightarrow \quad & K_{ii} - 2K_{ij} + K_{jj} \leq ||z_i - z_j||^2, \\
\forall ij \quad & u_i = u_j \\
\Rightarrow \quad & K_{(i+1)(i+1)} - 2K_{(i+1)(j+1)} + K_{(j+1)(j+1)} \\
= \quad & K_{ii} - 2K_{ij} + K_{jj}
\end{aligned}
$$

Run kernel PCA with the resulting kernel K.

As the MVU algorithm from subsection 2.2 is posed as an optimization, it can be readily modified for this purpose by changing existing constraints and adding new constraints. In particular, the extra action label information is used in two ways. First, since the observations are assumed to be in a complete trajectory, a non-uniform neighborhood graph can be constructed by connecting every point to those that are joined to it by an action (as well as any closer points).

Secondly, and more importantly, constraints are imposed which ensure that the actions are consistent in the resulting embedding. In particular, ARE enforces that the actions in the embedding are distance preserving, in other words are composed of translation and rotation, but not scaling. This set of transformations is called "distance preserving" because a transformation $f : X \to X$ is in the set if and only if,

$$\forall x_1, x_2 \in X \quad ||x_1 - x_2|| \quad = \quad ||f(x_1) - f(x_2)||. \quad (1)$$

Since the low-dimensional embedding only involves samples of the action operator being applied we only need to verify Equation 1 for these points:

$$a_i = a_j \Rightarrow ||x_i - x_j|| \quad = \quad ||x_{i+1} - x_{j+1}||$$

Such constraints on the effect of actions surprisingly can be written as direct constraints on the kernel matrix K. The details of these constraints and the rest of the algorithm is in Algorithm 1.

2.3.1 The Good Figure 1 shows a typical example of a representation learned by ARE contrasted by the embedding learned by MVU. The domain used is IMAGEBOT, where a robot is moving an observable image patch around a larger

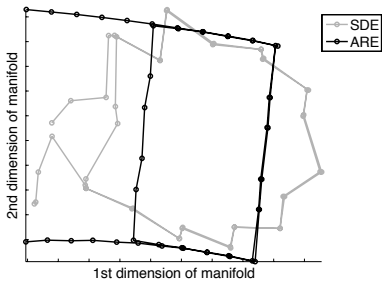

Figure 1: Results of Action Respecting Embedding (ARE) and Maximum Variance Unfolding (MVU) on extracting a representation from a single "A" shaped trajectory.

image. The actions correspond to North, South, East, and West, although the algorithm only sees semantic-less symbols. The robot's trajectory in objective coordinates corresponds to an "A" shape, which is largely mirrored in the ARE representation. In addition, ARE's representation is also able to capture that pairs of actions (North-South, East-West) are inverses and that the pairs are orthogonal, moving along independent dimensions. With dimensionality reduction alone, although a topologically correct representation is extracted, it is clear that finding a model of the effect of actions in this space would be nigh impossible. ARE's ability to capture the system's underlying state representation has led it be used as a basis for both subjective planning (Wilkinson, Bowling, & Ghodsi 2005) and subjective localization (Bowling *et al.* 2005).

2.3.2 The Bad The single most dramatic drawback of ARE is that at its core lies a semidefinite program with a large number ($O(n^2)$) of "action respecting" constraints. There are a number of general-purpose solvers, however they are all fairly expensive computationally. So, although the current results yielded by ARE are impressive, they are already unfortunately close to the limit on input size. A number of recent techniques have been proposed for scaling MVU to a larger number of points. Most of these techniques depend on the identification and embedding of a small number of landmarks using the computationally expensive semidefinite optimization. Because ARE's action respecting constraints are non-local, no small number of points can be embedded in a way that ensures that the actions constraints would be satisfied.

3 Action Guided Isomap

As described in Section 2.3, ARE extended the MVU algorithm to handle the case where additional action label information was available. Similarly, it is possible to modify the Isomap algorithm from Section 2.1. Recall that the core of Isomap was to build a distance matrix D, filling in neighbor entries with distances from the high-dimensional space and others with estimates of the geodesic distances. It is possible during this step to *attempt* to enforce the action-respecting constraints as they are constraints on distances. This is done through an iterative process. After computing D in the usual way, we investigate all entries D_{ij} where $a_i = a_j$. The constraints require $D_{ij} = D_{(i+1)(j+1)}$. If the two entries are not equal an edge is added to the original neighborhood graph to make those distances equal. Suppose $D_{ij} > D_{(i+1)(j+1)}$, then an edge is added to the neighborhood graph (or the length of an existing edge is reduced) between i and j with distance $D_{(i+1)(j+1)}$. This change could affect many other distances in D, so after adding an edge for each violated action constraint the entire matrix is recomputed. This whole procedure is then repeated, possibly many times, until either no distance constraint is violated or a set number of maximum iterations is reached. The complete details of the algorithm are in Algorithm 2.

Algorithm 2 Action Guided Isomap

Require: n high-dimensional data points $z_i \in \mathbb{R}^d$
U a set of distinct action labels
$n-1$ action labels $u_i \in U$ where u_i is the action between z_i and z_{i+1}
distance metric $d(x,y) \approx \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}$
Create non-uniform-neighborhood graph, η.
Create action-respecting distance matrix D:
if i and j are neighbors in η **then**
$D_{i,j} = d(z_i, z_j)$
else
$D_{i,j} = \infty$
end if
for $k = 1 \ldots n$ **do**
$\forall i, j \quad D_{i,j} = \min\{D_{i,j}, D_{i,k} + D_{j,k}\}$
end for
while constraints violated **do**
$\forall i, j$
if $u_i = u_j$ and $D_{i,j} \neq D_{i+1,j+1}$ **then**
set $D_{i,j} = D_{i+1,j+1}$
recompute D
end if
end while
Convert D to kernel matrix $K = -HSH/2$
where $S_{i,j} = D^2_{i,j}$ and H is the centering matrix.
Run kernel PCA with the resulting kernel K.

Theorem 1 proves that the action constraint corrections will converge, and so the final D matrix will satisfy all the action constraints. However, a configuration of points satisfying these distances may not be embeddable in a Euclidean space; and in particular, for the desired target dimensionality. The next section describes how the Action-Guided Isomap can be used as a seed to obtain a low-dimensional distance-preserving embedding.

Theorem 1 *AG-Isomap terminates in a finite number of action constraint iterations.*

Proof. Consider the distance matrix D whose entries are being updated on each iteration of checking the action constraints (the **while** loop in Algorithm 2). We first demonstrate the invariant that ever entry in the matrix is a sum of edge distances from the generating neighborhood graph. The initial computation computes path lengths between

points and so clearly satisfies the invariant. Iterations of the **while** loop replace entries with smaller entries (maintaining the invariant) and then recomputes path lengths (again, maintaining the invariant), thus proving the invaraint. Now, consider the longest path in the generating neighborhood graph. There is a finite number of possible edge sums less than this longest path length. Therefore, by the invariant, there is a finite number of possible values each entry in the distance matrix can ever assume, and as a consequence, there is a finite number of distance matrices that can ever be formed. On every iteration the distance matrix changes (or the algorithm terminates), and the matrix can never return to the same value (since the entries are monotonically decreasing), so the matrix must remain unchanged after a finite number of iterations (because there's only a finite number of possible distance matrices). Hence, AG-Isomap terminates.

4 Procrustes Gradient Correction

In Section 2.3 the ARE algorithm was posed as an optimization to learn an appropriate kernel. This is really the dual of the actual problem of interest which is to find the positions of the low dimensional points on a manifold. Alternatively, the primal problem could be solved directly; that is, by optimizing the positions of a collection of points subject to constraints on distances and the constraint that actions are distance-preserving. This would eliminate the constraint that a matrix be positive semidefinite, and so a semidefinite solver is no longer necessary. Instead, a standard conjugate gradient descent algorithm could be used.

A naive approach would be to minimize the loss function that directly corresponds to the ARE optimality condition. i.e.

$$L_{i,j} = \|x_i - x_j\|_2^2 - \|x_{i+1} - x_{j+1}\|_2^2 \text{ and} \quad (2)$$

$$M_{i,j} = \|x_i - x_j\|_2^2 - D_{i,j}^2. \quad (3)$$

Where L represents loss based on action-respecting constraints and M represents loss based on the preservation of local distances from some neighborhood graph. Then the total loss function to be minimized is:

$$\Phi = \sum_{a \in A} \sum_{i,j \in S_a} L_{i,j}^2 + M_{i,j}^2 \quad (4)$$

where A is the set of distinct action labels and S_a is the set of pairs of points joined by action label a.

Unfortunately, the portion of the loss function from Equation 2 contains terms for all possible $O(n^2)$ action-respecting constraints. This means that the function and its derivative need to be computed for every iteration of the conjugate gradient, which is computationally very expensive. We present an alternative approach however which is $O(n + k)$ where k is the number of edges in the neighborhood graph. This approach will be computationally much more efficient than solving the primal form directly.

4.1 An alternative primal formulation

Recall that ARE learns a representation with explicit constraints that the actions correspond to distance-preserving transformations in that representation. For each unique action u there is a collection of data point pairs (x_i, x_{i+1}) which are connected by that action. Another way of thinking of this is that there is a function f_u where $f_u(x_i) = x_{i+1}$, and such a function needs to be learned for each action. Because of the distance-preserving constraints, f_u can be represented as: $f_u(x_i) = A_u x_i + b_u = x_{i+1}$. Transformations of the above form encode translation in the b_u vector, and rotation and scaling in the A_u matrix. A_u and b_u could be learned using simple linear regression but scaling is not distance preserving so there is the additional constraint that A_u does not scale, i.e., $A_u^T A_u = I$. It turns out that this is similar to the extended orthonormal Procrustes problem (Schoenemann & Carroll 1970), but without allowing for a global scaling constant and can be solved in closed form (Wilkinson, Bowling, & Ghodsi 2005).

This suggests a loss function that measures the difference between the position of a point and its prediction as estimated by the application of the appropriate transformation to the previous point.

$$L_i = \|x_{i+1} - A_{u_i} x_i - b_{u_i}\|_2^2$$

where now there are only $O(n)$ terms (M is the same as before). Note that each point will appear twice, once as x_i and again as x_{i+1} and therefore 5 and 6 can be combined for the full gradient of the function.

$$\frac{\delta L_i}{\delta x_i} = -2A_i^T x_{i+1} + 2A_i^T A_i x_i + 2A_i^T b_{u_i} \quad (5)$$

$$\frac{\delta L_i}{\delta x_{i+1}} = 2x_{i+1} - 2A_i x_i - 2b_i \quad (6)$$

5 Results

We now examine the ability of the proposed algorithm at learning low-dimensional action-respecting representations. Results are in the IMAGEBOT domain introduced in Section 2.3. We demonstrate the effectiveness of Action-Guided Isomap on a simple dataset, where its usefulness as a seed for a primal optimization is contrasted with regular Isomap. Both of the conjugate gradient methods discussed in Section 4 are compared on a few datasets of varying orders of magnitude. Finally, the new procedures are compared to the original ARE algorithm, and their empirical running-times are presented.

For these results, ARE refers to the original ARE algorithm, and Isomap refers to the original Isomap algorithm. AG-Isomap refers to the Action Guided Isomap described in Section 3.

PG-ARE is a two step algorithm. First AG-Isomap is used to generate a starting point which is then used with the Procrustes conjugate gradient optimization described in Section 4. CG-ARE is a naive conjugate gradient algorithm which uses AG-Isomap to generate a starting point but then solves a conjugate gradient to explicitly optimize the primal form of the problem.

For these experiments, IMAGEBOT is always viewing a 200 by 200 patch of a 1914 by 1960 image. Six distinct actions are used: four translation actions, and two rotation actions. The translations are 'north', 'south', 'east' and 'west',

each by 10 pixels. The rotation actions are 'turn clockwise' and 'turn counter-clockwise', each by $22\frac{1}{2}^\circ$. As an example, Figure 2 shows an 'A'-shaped trajectory that IMAGEBOT followed, superimposed on a portion of the image that is IMAGEBOT's environment.

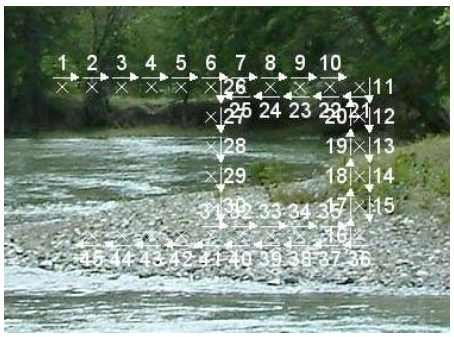

Figure 2: A sample IMAGEBOT trajectory.

5.1 Manifold Learning

The first row of Figure 3 shows the results of Isomap and AG-Isomap applied to data from the trajectory in Figure 2. Isomap fails to generate a manifold in which actions have a simple interpretation. AG-Isomap captures the true topology of the trajectory, but fails to produce an action respecting embedding on its own.

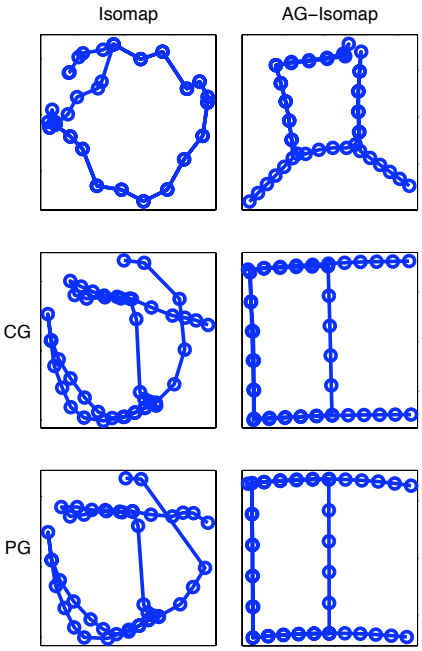

Figure 3: Results of Isomap (left) and Action-Guided Isomap (right) on the A trajectory

The second and third rows show the result of using these embeddings as a seed for each of the conjugate gradient methods described in Section 4. The second row demonstrates the naive conjugate gradient approach, where the ARE loss function is optimized directly, and the third row shows the more efficient Procrustes method from Section 4.1. In both cases, the Isomap seed is insufficient to produce a meaningful result, whereas the AG-Isomap seed allowed the primal search methods to uncover the "letter A" trajectory.

Figure 4 continues to compare the two conjugate gradient methods on a few different datasets. In each case, AG-Isomap is used to seed the optimization. The first row presents the "letter A" dataset as used previously. The second is an IMAGEBOT trajectory that spells the letters "ISAIM", and the third row shows a simple pattern constructed from very large rectangular paths. Note that PG-ARE enforces that actions apply consistent transformations in the learned embedding, whereas this condition is not maintained so strictly with CG-ARE.

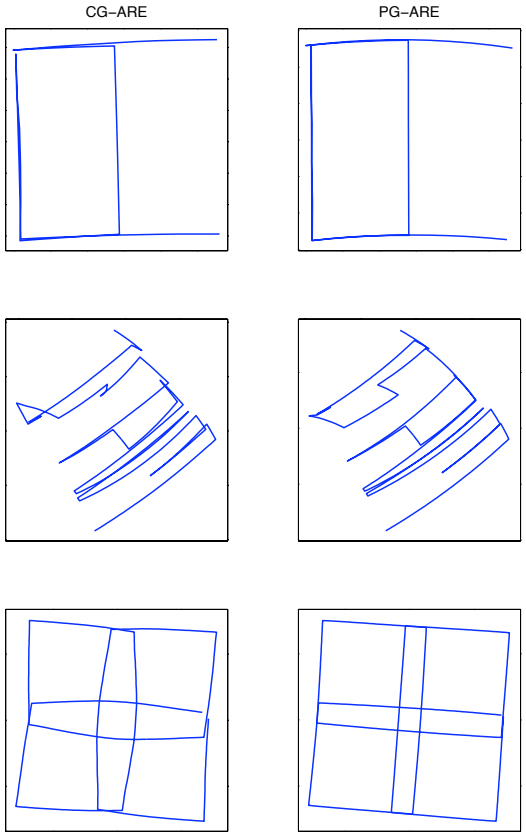

Figure 4: Results of CG-ARE (left) and PG-ARE (right) on three datasets.

The original ARE algorithm is compared to the new CG-ARE and PG-ARE methods in Figure 5. Results are very similar, which shows that CG-ARE and PG-ARE are appropriate alternatives to the original ARE algorithm. The second "circle" data is an IMAGEBOT trajectory of 16 counter-

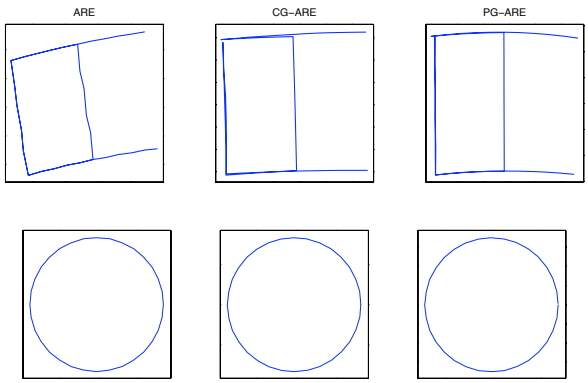

Figure 5: Results of ARE (left), CG-ARE (middle), and PG-ARE (right) on two small trajectories.

clockwise steps of rotation and translation.

CPU running times for the various algorithms is provided in Table 1. For the smaller data sets, ARE, CG-ARE, and PG-ARE have similar running times. It is worth mentioning that PG-ARE is somehow slower than CG-ARE on small data sets due to its overhead for computing Procrustes operators. On the larger data sets, however, PG-ARE is noticeably faster than CG-ARE as expected. Note that the times for PG-ARE and CG-ARE both include the computation of AG-Isomap. Furthermore, ARE is infeasible for the larger datasets and so its running time cannot be compared.

Dataset	Letter A	Circle	ISAIM	Rectangles
# points	46	33	517	761
AG-Isomap	0.13 s	0.11 s	214 s	1265 s
CG-ARE	3.29 s	1.48 s	675 s	1826 s
PG-ARE	5.97 s	1.69 s	287 s	1352 s
ARE	4.35 s	1.95 s	n/a	n/a

Table 1: Algorithm run-times. Times shown for CG-ARE and PG-ARE include the AG-Isomap computation.

6 Conclusion

ARE is a promising way of learning representations from a raw stream of experience. Due to the semidefinite optimization at its core, there are scaling problems. The combination of solving the primal problem of directly learning the representation with a non-convex conjugate gradient method seeded with representations learned by action-guided Isomap proves successful. Not only are results quite similar to those of ARE obtained on smaller data sets, but good representations are learned for problems whose input sizes are an order of magnitude larger.

References

Bowling, M.; Wilkinson, D.; Ghodsi, A.; and Milstein, A. 2005. Subjective localization with action respecting embedding. In *ISRR 2005*. The International Symposium of Robotics Research.

Bowling, M.; Ghodsi, A.; and Wilkinson, D. 2005. Action respecting embedding. In Raedt, L. D., and Wrobel, S., eds., *ICML 2005*, 65–72. The 22nd International Conference on Machine Learning.

Bowling, M.; Wilkinson, D.; and Ghodsi, A. 2006. Subjective mapping. In *New Scientific and Technical Advances in Research (NECTAR) at AAAI 2006*, 1569–1572. The 21st National Conference on Artificial Intelligence.

Cox, T. F., and Cox, M. A. A. 2000. *Multidimensional Scaling, Second Edition*. Chapman & Hall/CRC.

Schoenemann, P. H., and Carroll, R. 1970. Fitting one matrix to another choice of a central dilation and a rigid motion. *Psychometrika* 2(35).

Scholkopf, B., and Smola, A. 2002. *Learning with Kernels*. MIT Press.

Shawe-Taylor, J., and Cristianini, N. 2004. *Kernel Methods for Pattern Analysis*. Cambridge University Press.

Tenenbaum, J. B.; de Silva, V.; and Langford, J. C. 2000. A global geometric framework for nonlinear dimensionality reduction. *Science* 290(5500):2319–2323.

Weinberger, K. Q.; Sha, F.; and Saul, L. K. 2004. Learning a kernel matrix for nonlinear dimensionality reduction. In *Proceedings of the Twenty First International Conference on Machine Learning (ICML-04)*, 839–846.

Wilkinson, D.; Bowling, M.; and Ghodsi, A. 2005. Learning subjective representations for planning. In *IJCAI 2005*. The 19th International Joint Conference on Artificial Intelligence.