Metric Learning for Estimating Psychological Similarities *

Jun-Ming Xu† , Xiaojin Zhu† , Timothy T. Rogers‡
†Department of Computer Sciences, ‡Department of Psychology
University of Wisconsin-Madison
{xujm, jerryzhu}@cs.wisc.edu, ttrogers@wisc.edu

Abstract

An important problem in cognitive psychology is to quantify the perceived similarities between stimuli. Previous work attempted to address this problem with multi-dimensional scaling (MDS) and its variants. However, there are several shortcomings of the MDS approaches. We propose Yada, a novel general metric learning procedure based on two-alternative forced-choice behavioral experiments. Our method learns forward and backward nonlinear mappings between an objective space in which the stimuli are defined by the standard feature vector representation, and a subjective space in which the distance between a pair of stimuli corresponds to their perceived similarity. We conduct experiments on both synthetic and real human behavioral datasets to assess the effectiveness of Yada. The results show that Yada outperforms several standard embedding and metric learning algorithms, both in terms of likelihood and recovery error.

1 Introduction

Since Fechner invented the field of psychophysics in [7], psychologists have been interested in the way our mind perceives similarity between objects. Computationally, each object (or stimulus) is usually represented by a feature vector $x \in \mathbb{R}^d$. This feature representation induces a convenient "objective distance" between two stimuli x_i, x_j through the ℓ_q norm:

$$\|x_i - x_j\|_q. \tag{1}$$

Often q is taken to be 2 for the Euclidean distance, or 1 for the Manhattan distance.

However, it is less clear how such objective distances relate to the "subjective distances", or the perceived similarities, among stimuli. For instance, many empirical studies of human learning employ Gabor patches shown in Figure 1. Such stimuli are thought to be especially useful because they appear to closely match the kinds of information extracted by neurons in early visual processing streams in mammalian cortex. The feature vector of a Gabor patch consists of the angle of the grating, its frequency, and its amplitude (i.e., degree of contrast). The problem is, it is not clear how these features relate to the subjective distances. Human perception is subject to a variety of nonlinear and configural processes that can make any such relationship extremely opaque. For instance, people appear to treat horizontal and vertical gratings as qualitatively distinct from oblique angles. Dimensions that are independent in principle, like the amplitude and frequency of a Gabor grating, are interdependent in perception, with some amplitudes being harder to see at higher frequencies. Even perception of individual dimensions for such stimuli can be highly nonlinear. And if these issues pose challenges even for simple stimuli like Gabor patches, one can imagine the challenges posed by real-world stimuli like faces or objects. Yet computational theories of human learning depend critically upon having subjective distances existing among the stimuli being learned. Essentially all such

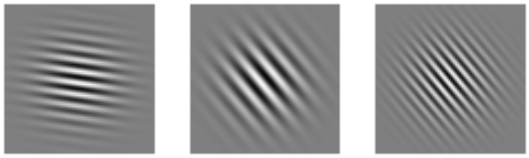


Figure 1: Examples of Gabor stimuli

theories invoke this subjective, psychological distance to explain how learning about an individual item will generalize to other items.

To address this issue, cognitive scientists have typically taken one of two approaches. The first is to make the simplifying assumption that the subjective distances are proportional to the objective distances, at least for restricted ranges for some kinds of stimuli. For instance, in studies with Gabor patches, the theorist might take care to employ stimuli with oblique orientations that are not too close to vertical or horizontal; and might further assume that, within this range, the true difference in orientation for a given pair of items is proportional to the perceived dissimilarity of the items. Such assumptions may seem intuitively reasonable but are difficult to formally justify; and moreover, it seems limiting to simply avoid "problematic" stimuli like vertical and horizontal gratings.

The second approach uses Multi-Dimensional Scaling (MDS) to find an embedding that relates to the subjective distances among some fixed set of stimuli [19]. There are a variety of different methods for collecting information about subjective similarity, and MDS has been applied to all of these. One of them, the confusion matrix, is widely accepted as yielding the most robust results. In the standard procedure, participants learn n classes, each containing a single unique item. Subsequently, they are tested by classifying each of the n items multiple times. From this data, a confusion matrix is constructed describing the number of times that item x_i was misclassified as item x_j. MDS can then be applied to this matrix to embed the n items into a low dimensional space, which is taken to be the psychological space in which the subjective distances are defined. This approach has the advantage that it can be applied to any arbitrary set of stimuli, and furthermore has yielded very robust results. Shepard, for instance, famously showed that the generalization of a learned category falls off exponentially when distances are measured in this way [20]. Nosofsky has capitalized on this method to develop a computational theory of identification and categorization that captures human performance with a high degree of accuracy [14, 15]. People have used other kinds of procedure to estimate subjective similarity, including similarity judgments and same/different judgments, but that these have well-known problems [13].

This MDS approach still has two important drawbacks, however. The first is that the learning and testing phase of the procedure is very labor-intensive, so that it is only possible to measure distances among a limited set of items. For example, much of Nosofsky's work has involved just $n = 16$ individual items. The second problem is that MDS only yields an embedding on training stimuli – it does not offer a way to map an unseen stimulus from its feature space to the psychological space.

This paper aims to address these problems by developing a novel method to translate objective distances to subjective distances, and vice versa. Our method is a general metric learning procedure that elicits perceived similarities in humans, and learns a continuous, nonlinear mapping from the feature space to the psychological space. This mapping (and its inverse mapping) can be applied to novel stimuli.

2 The Two-Alternative Forced Choice Match-to-Sample Procedure

In order to study subjective distances, one needs to measure some form of the perceived similarities among stimuli. As previously noted, past work in cognitive psychology has made extensive use of confusion matrices generated within an identity-recognition paradigm to measure such similarities. As also noted, however, such work is extremely labor-intensive and allows the psychologist to measure similarities amongst only a

small number of items. For these reasons, we have instead adapted a method frequently used to measure perceptual discriminability in psychophysics, specifically the two-alternative forced choice match-to-sample (2AFC) procedure.

In each trial of the 2AFC procedure, participants encounter 3 stimuli. The first stimulus (X) is displayed by its own on the screen. It disappears after a short while, and two stimuli A and B appear simultaneously on the screen. One of A or B is identical to X, while the other is different from X. The relative position of A and B is randomized in each trial.

The participant must judge whether the first stimulus X matched A or B. This procedure is sensitive to the perceptual similarity between A and B: as they grow more similar, performance drops to chance. This procedure does not require any category or identity learning and so can be applied to a much larger set of stimuli. Like identity-learning, the method produces a confusion matrix in which each cell indicates, for some pair of A and B, how frequently the participant chose the wrong match for X. More frequent incorrect choices indicate more perceptual confusability (and hence similarity) between A and B.

3 Problem Formulation

We now formally define our learning problem. Let $\mathcal{X} \subseteq \mathbb{R}^d$ be an *objective space*, in which items $x \in \mathcal{X}$ are represented as d-dimensional feature vectors. We assume that there is a corresponding *subjective space* $\phi(\mathcal{X}) \subseteq \mathbb{R}^p$ where potentially $p \neq d$. The function $\phi : \mathbb{R}^d \mapsto \mathbb{R}^p$ is an unknown nonlinear mapping from the objective space to the subjective space. We cannot directly observe $\phi()$. Nonetheless, its effects can be "felt" by the following means. For any two items $x_i, x_j \in \mathcal{X}$, we may observe a *comparison function* $f : \mathcal{X} \times \mathcal{X} \mapsto \mathbb{R}$

$$f(x_i, x_j) \equiv f(\|\phi(x_i) - \phi(x_j)\|). \tag{2}$$

We overload f here to declear that f depends only on the subjective distance $||\phi(x_i) - \phi(x_j)||$ between the two items. Notice that f is a stochastic function because it is intended to model a random outcome of a psychology experiment. The outcome may be affected by many factors, such as the subjects and experimental conditions. Given a set of items $x_1, \ldots, x_n \in \mathcal{X}$ and a limited number of comparisons using f, we want to estimate the mapping ϕ.

This task is more general than embedding, whose goal is to find the images of the training items: $\phi(x_1), \ldots, \phi(x_n)$. However, embedding usually does not aim to learn the mapping function ϕ and therefore has difficulty generalizing to the image of an unseen test item x^*. This task is also more general than traditional metric learning, whose goal is to find the subjective distances $\|\phi(x_i) - \phi(x_j)\|$ in our context. However, metric learning does not aim at producing an embedding or mapping.

3.1 A Stochastic Comparison Function for the 2AFC Experiments

To fully define the problem, we need to instantiate the stochastic comparison function f for the 2AFC experiments. If two items x_i, x_j are identical, their subjective distance

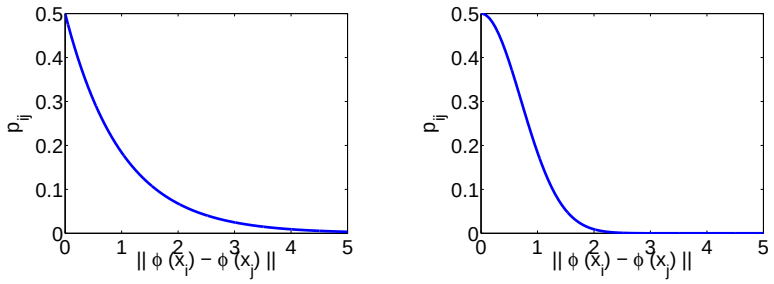
$$\|\phi(x_i) - \phi(x_j)\| \tag{3}$$

should be zero. In this case, in the 2AFC experiment a human participant would have chance probability $(\frac{1}{2})$ of picking out the target item. As the two items become gradually distinct, their subjective distance grows. The probability of the participant picking out the wrong item should decrease toward zero. This suggests that we can model the probability of participant error on the pair x_i, x_j in 2AFC with a monotonic function:

$$p_{ij} \equiv \psi\left(\|\phi(x_i) - \phi(x_j)\|\right). \tag{4}$$

The function $\psi : \mathbb{R}_+ \mapsto [0, 0.5]$ is closely related to the *choice model* in psychology. Following [13], we consider to function forms, see Figure 2. One is an exponential decay function

$$\psi_e(z) = \frac{1}{2}\exp\left(-z\right), \tag{5}$$



(a) Exponential decay ψ_e (b) Gaussian decay ψ_G

Figure 2: Two possible choice models

another is a Gaussian decay function

$$\psi_G(z) = \frac{1}{2}\exp\left(-z^2\right). \tag{6}$$

Of course, we do not observe p_{ij} but rather the binary outcomes of each 2AFC trial. Our stochastic comparison function f models this as independent Bernoulli random variables:

$$f(x_i, x_j) \sim \text{Bernoulli}(p_{ij}). \tag{7}$$

We are now ready to describe the 2AFC experiment formally. First of all, we aggregate all participants together and treat their behavioral data collectively. Altogether, there are n different items $x_1, \ldots, x_n$ involved in the experiments. Each pair (x_i, x_j) was shown n_{ij} times to the participants. The number n_{ij} could be zero for some pairs – in this case we do not have 2AFC behavioral data for those pairs. Each time (x_i, x_j) was shown, a participant makes a choice. The binary outcome of the stochastic comparison function $f(x_i, x_j)$ corresponds to whether the choice was incorrect or correct. Upon completion of the experiments, let a_{ij} be the number of times that the participants were incorrect on the pair (x_i, x_j),

$$a_{ij} = \sum_{1}^{n_{ij}} f(x_i, x_j) \tag{8}$$

and b_{ij} be the number of times that they were correct,

$$b_{ij} = n_{ij} - a_{ij}. \tag{9}$$

We further define the set of pairs on which we have 2AFC data

$$E = \{(i, j) \mid n_{ij} > 0\}. \tag{10}$$

3.2 Yada: Our Model

Given a set of items represented as feature vectors in the objective space $x_1, \ldots, x_n \in \mathbb{R}^d$, 2AFC experiment results $\{(a_{ij}, b_{ij}) \mid (i, j) \in E\}$, a choice model ψ, and a desirable dimensionality of subjective space p, our goal is to learn the mapping $\phi : \mathbb{R}^d \mapsto \mathbb{R}^p$ which relates the two spaces.

We introduce Yada (*Yet Another Dimension-reduction Algorithm*) for this purpose. We start with a simple linear mapping in the form

$$\phi(x) = Ax, \tag{11}$$

where A is a $p \times d$ matrix to be learned. Then the subjective distance between two items x_i and x_j is

$$\|\phi(x_i) - \phi(x_j)\| = \|A(x_i - x_j)\|. \tag{12}$$

Applying the choice model, we get the probability that a participant chooses the incorrect item in 2AFC experiments on that pair,

$$p_{ij} = \psi\left(\|A(x_i - x_j)\|\right). \tag{13}$$

As different subjects' choices can be treated as independent Bernoulli trials, the log-likelihood function is

$$\ell(A) \equiv \sum_{(i,j)\in E} a_{ij} \log p_{ij} + b_{ij} \log(1 - p_{ij}). \tag{14}$$

For stability, we add the squared Frobenius norm on A as a regularizer,

$$\|A\|_F^2 = \mathrm{Tr}(AA^\top), \tag{15}$$

and arrive at the linear Yada optimization problem,

$$\min_A -\ell(A) + \mu\|A\|_F^2, \tag{16}$$

where $\mu > 0$ is a regularization parameter.

The expressive power of this linear mapping is very limited. It is likely that the mapping ϕ is nonlinear in human cognition. Therefore, we enhance our model by the kernel trick. Note that the objective function is the sum of a loss function plus a monotonic function of Frobenius norm of A. By the representer theorem [16, 1], the solution A to the optimization problem (16) admits the following form

$$A = WX, \tag{17}$$

where W is a $p \times n$ matrix and $X = (x_1 \dots x_n)^\top$ is the $n \times d$ input matrix. The subjective distance between item x_i and x_j can be written as

$$\|\phi(x_i) - \phi(x_j)\| = \|WX(x_i - x_j)\|. \tag{18}$$

Let K be a $n \times n$ kernel Gram matrix on $x_1, \dots, x_n$, whose entries are $K_{ij} = \langle x_i, x_j \rangle$, and $\mathbf{k}_i$ be the i-th column of K. For nonlinear kernel, let $K_{ij} = \langle \sigma(x_i), \sigma(x_j) \rangle$, where σ is the feature mapping induced by kernel K. With the notation of K, the perceptual distance can be written as

$$\|\phi(x_i) - \phi(x_j)\| = \|W\sigma(X)\left(\sigma(x_i) - \sigma(x_j)\right)\| = \|W(\mathbf{k}_i - \mathbf{k}_j)\|. \tag{19}$$

Thus, the log-likelihood is now a function of W,

$$\ell(W) \equiv \sum_{(i,j)\in E} a_{ij} \log \psi(\|W(\mathbf{k}_i - \mathbf{k}_j)\|) + b_{ij} \log(1 - \psi(\|W(\mathbf{k}_i - \mathbf{k}_j)\|)). \tag{20}$$

The regularizer can be represented in terms of W too:

$$\|A\|_F^2 = \mathrm{Tr}(AA^\top) = \mathrm{Tr}(W\sigma(X)\sigma(X)^\top W^\top) = \mathrm{Tr}(WKW^\top). \tag{21}$$

Therefore, the kernelized Yada optimization problem is

$$\min_W -\ell(W) + \mu\mathrm{Tr}(WKW^\top). \tag{22}$$

It is easy to see that this problem is non-convex, because rotating W does not change the objective. In practice, though, we found that simple gradient methods are sufficient to solve (22) up to local optimum satisfactorily, and multiple local optima are not a serious issue if we are careful about initialization as discussed in Section 3.3.

The gradient of (22) is

$$\nabla_W = -\sum_{(i,j)\in E} \left(\frac{a_{ij}}{\psi(\|W(\mathbf{k}_i - \mathbf{k}_j)\|)} - \frac{b_{ij}}{1 - \psi(\|W(\mathbf{k}_i - \mathbf{k}_j)\|)} \right) \frac{\partial \psi(\|W(\mathbf{k}_i - \mathbf{k}_j)\|)}{\partial W} + 2\mu WK. \tag{23}$$

If we use the exponential choice model ψ_e (5),

$$\frac{\partial \psi_e(\|W(\mathbf{k}_i - \mathbf{k}_j)\|)}{\partial W} = -\frac{\psi_e(\|W(\mathbf{k}_i - \mathbf{k}_j)\|)W(\mathbf{k}_i - \mathbf{k}_j)(\mathbf{k}_i - \mathbf{k}_j)^\top}{\|W(\mathbf{k}_i - \mathbf{k}_j)\|}. \tag{24}$$

If we use the Gaussian choice model ψ_G (6),

$$\frac{\partial \psi_G(\|W(\mathbf{k}_i - \mathbf{k}_j)\|)}{\partial W} = -2\psi_G(\|W(\mathbf{k}_i - \mathbf{k}_j)\|)W(\mathbf{k}_i - \mathbf{k}_j)(\mathbf{k}_i - \mathbf{k}_j)^\top. \tag{25}$$

3.3 Initialization for Gradient Descent

It is well known in the machine learning community that informative initialization can alleviate the multiple local optima issue. Therefore, we propose the following initialization procedure. When the dimensions of the objective space and the subjective space are the same ($d = p$), we initialize W with W_0, where W_0 generates an embedding closest to the objective space, i.e.

$$W_0 = \underset{W}{\operatorname{argmin}} \|WK - X\|_F^2 + \mu \operatorname{Tr}(WKW^\top). \tag{26}$$

This initialization has a closed-form solution:

$$W_0 = XK^\top(\mu K + KK^\top)^{-1}. \tag{27}$$

If the dimension of the objective space is greater than that of the subjective space ($d > p$), we run principal component analysis (PCA) on X to project it down to a p-dimensional subspace, and then apply the initialization process (26). If the objective space has lower dimensionality than the subjective space ($d < p$), we augment x with $p - d$ extra dimensions and set these dimensions with random values sampled from $N(0, 1)$, then apply (26).

Besides informative initialization, we also experimented with multiple random starting points. We observed that, in most cases, there was no significant difference on the result between the two initialization methods. However, multiple random starting points method took much more computation resources than informative initialization, because it requires running optimization procedure multiple times.

3.4 Forward and Backward Mappings

After we solve the Yada problem (22), we can perform the "forward mapping" $\phi : \mathbb{R}^d \mapsto \mathbb{R}^p$ for any x (not limited to the training items) from objective space to subjective space:

$$\phi(x) = W\mathbf{k}_x, \tag{28}$$

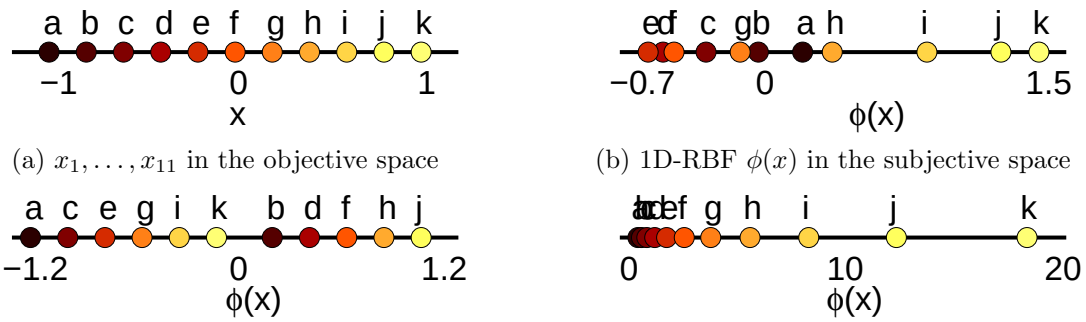
where $\mathbf{k}_x = (K(x_1, x), \ldots, K(x_n, x))^\top$.

In some circumstances, a "backward mapping" $\phi^{-1} : \mathbb{R}^p \mapsto \mathbb{R}^d$ is also of interest. Psychologists may be interested in generating a stimulus x that maps to a specific point in the subjective space. For example, they may wish to design a set of stimuli that lie on a regular grid in the subjective space to experiment with. These stimuli would have the desirable property that the subjective distance between neighboring items are normalized.

Since ϕ may map multiple points in $\mathbb{R}^d$ to the same point in the subjective space $\mathbb{R}^p$, the backward mapping ϕ^{-1} may be one-to-many. This backward mapping can be formulated as an optimization problem. Given $y \in \mathbb{R}^p$ in the subjective space, we want to find one of the points $x^* \in \mathbb{R}^d$ in the objective space such that $\phi(x^*)$ is the closest to y:

$$x^* = \underset{x \in \mathbb{R}^d}{\operatorname{argmin}} \|\phi(x) - y\|^2 \tag{29}$$

$$= \underset{x \in \mathbb{R}^d}{\operatorname{argmin}} -2y^\top W\mathbf{k}_x + \mathbf{k}_x^\top W^\top W\mathbf{k}_x. \tag{30}$$



(a) $x_1, \ldots, x_{11}$ in the objective space

(b) 1D-RBF $\phi(x)$ in the subjective space

(c) 1D-cos $\phi(x)$ in the subjective space

(d) 1D-exp $\phi(x)$ in the subjective space

Figure 3: 1D synthetic datasets

The optimization problem depends on the kernel K and may not be convex for some kernels. However, we can still use gradient descent to find a local minimum. To choose a good initial point, we consider a set of stimuli $\{\tilde{x}_1, \cdots, \tilde{x}_m\}$ on a regular grid in $\mathbb{R}^d$. The $\tilde{x}$ with the smallest subjective distance $\|\phi(\tilde{x}_i) - y\|$ is selected as the initial value for x in (30).

The complete Yada algorithm is given in Algorithm 1.

Algorithm 1: Yada

Input: Items $x_1, \ldots, x_n \in \mathbb{R}^d$, 2AFC results $\{(a_{ij}, b_{ij}) \mid (i,j) \in E\}$, choice model ψ, subjective space dimension p, kernel function K

Output: $\phi(x) : \mathbb{R}^d \mapsto \mathbb{R}^p$

$X \leftarrow (x_1, \ldots, x_n)^\top$;

if $p < d$ **then**

  $X \leftarrow$ embedding X in p-dimensional subspace by PCA;

else if $p > d$ **then**

  $X \leftarrow$ filling the extra $(p-d)$ dimension of X with random value sampled from $N(0,1)$;

end

Initialize W_0 with (26);

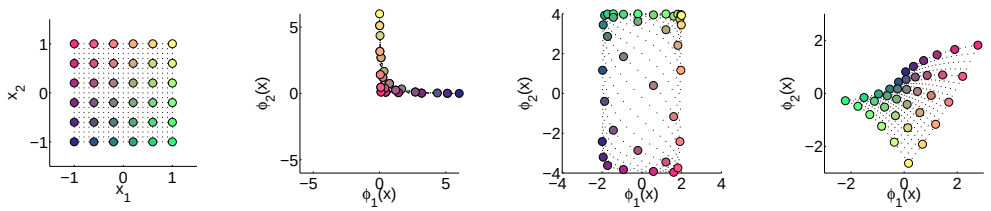
Solve W with gradient descent to optimize (22);

Forward mapping: $\phi(x) = W(K(x_1, x) \ldots K(x_n, x))^\top$ for any $x \in \mathbb{R}^d$;

Backward mapping: solve (30) for any $y \in \mathbb{R}^q$.

4 Experiments

In order to evaluate the effectiveness of Yada in learning the mapping ϕ, we conducted experiments on synthetic and real datasets. We compared Yada's performance to other existing algorithms, including non-metric multi-dimensional scaling and three other metric learning algorithms. In the following, we first introduce the datasets we experimented on. We then describe our experiment procedure and evaluation criteria. We found that compared to other algorithms, Yada recovers the mapping ϕ better in both evaluation criteria considered.



(a) $x_1 \dots x_{36}$ (b) 2D-RBF (c) 2D-tanh (d) 2D-poly

Figure 4: 2D synthetic datasets

4.1 The Data

4.1.1 Synthetic Datasets

One advantage of synthetic datasets is that the true mapping ϕ is available and we know the true pairwise distances in the subjective space. Therefore, we can compare how well different algorithms recover the pairwise distances. We created three 1D datasets and three 2D datasets with different characteristics. As in real 2AFC experiments, we first selected a set of items $x_1, x_2 \dots, x_n \in \mathbb{R}^d$ in the objective space. For 1D datasets, we chose 11 items in the objective space on a regular grid between -1 and 1 with stepsize 0.2, i.e. $x = \{-1, -0.8, -0.6, \dots, 1\}$, see Figure 3(a). For 2D datasets, we chose 36 items on a regular grid between -1 and 1 over both axis with stepsize 0.4, see Figure 4(a). For each objective dataset we created three different subjective datasets as discussed below.

1. *1D-RBF.* In all our experiments, we let K be the RBF kernel. Therefore, we created a 1D synthetic dataset whose true mapping ϕ is within the family representable by linear combination of RBF kernel functions in the form

$$\phi(x) = \sum_{i=1}^{n} w_i \exp(-\|x - x_i\|^2) \tag{31}$$

   where x_i's are the training items. In particular, we set

$$\phi(x) = 2\exp(-(x+0.6)^2) - 3\exp(-(x+0.2)^2) + 2\exp(-(x-0.8)^2). \tag{32}$$

   On one hand, this is an "easy" dataset for Yada because there is no model mismatch. On the other hand, as shown in Figure 3(b), the mapping is nonlinear and the order of items is not preserved.

2. *1D-cos.* The mapping is

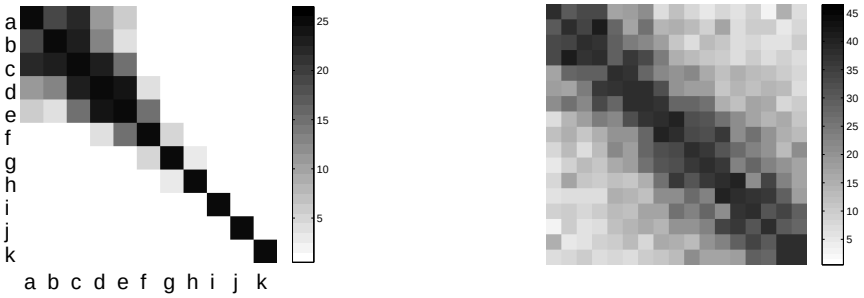
$$\phi(x) = 0.6\cos(5\pi x) + 0.5x. \tag{33}$$

   This produces a highly nonlinear, shuffled mapping as shown in Figure 3(c). It is also outside the family of mappings representable by RBF kernels.

3. *1D-exp.* The mapping is an exponential function of x, see Figure 3(d):

$$\phi(x) = \exp(2(x+1))/3. \tag{34}$$

   The few right-most points are far from each other and other items. Therefore, according to the choice model, with overwhelming probability they are not confusable. This poses a challenge for the 2AFC experiment because as long as the subjective distance is large enough, a_{ij} is expected to be zero. That is to say, a metric learning algorithm has little information on the actual subjective distance – all it knows is that the distance is large.



(a)Synthesized confusion matrix on 1D-exp, $n_{ij} = 50$

(b) Actual confusion matrix from 2AFC experiments on Lines, $n_{ij} = 76$

Figure 5: The confusion matrix $[a_{ij}]$ in synthetic and real datasets

4. *2D-RBF*. We constructed a 2D dataset within the function family our algorithm searches in:

$$\phi_1(x) = 6\exp\left(-(x_1+1)^2-(x_2+1)^2\right) \quad (35)$$
$$\phi_2(x) = 6\exp\left(-(x_1-1)^2-(x_2-1)^2\right). \quad (36)$$

   This mapping collapses a 2D grid into an L-shaped region, see Figure 4(b).

5. *2D-tanh*. This 2D dataset has a mapping that is dense near the boundaries while sparse in the center, see Figure 4(c).

$$\phi_1(x) = 2\tanh(x_1+2x_2+0.1) \quad (37)$$
$$\phi_2(x) = 4\tanh(2.5x_1-x_2+0.8). \quad (38)$$

6. *2D-poly*. This 2D dataset has a polynomial function for the true mapping, see Figure 4(d).

$$\phi_1(x) = 0.1(-0.3x_1+0.5x_2+1.5)^4 - 0.08(0.5x_1-0.3x_2+1.5)^4 \quad (39)$$
$$\phi_2(x) = 0.08(-0.6x_1+0.3x_2+1.5)^4 - 0.1(0.3x_1+0.5x_2+1.5)^4. \quad (40)$$

   It is yet another nonlinear mapping from the objective space to the subjective space.

Given the items $x_1 \dots x_n$ and the true mapping function ϕ, we sampled the $\{(a_{ij}, b_{ij})\}$ counts as follows. We computed the pairwise subjective distances $\|\phi(x_i) - \phi(x_j)\|_2$. We then used the Gaussian decay choice model ψ_G (6) to compute the error probabilities p_{ij}. We simulated $n_{ij} = 50$ subjects participated in each 2AFC experiment, which is a reasonable number in real experiments. Because different subjects' choices are modeled by independent Bernoulli trails, we sampled $a_{ij} \sim \text{Binomial}(50, p_{ij})$ as the number of subjects choosing the wrong items in the 2AFC experiment for pair (x_i, x_j), and $b_{ij} = 50 - a_{ij}$ is the number of subjects choosing the correct selections. Figure 5(a) shows the confusion matrix $[a_{ij}]$ generated by the above procedure for the 1D-exp dataset. We label and order the items as they appear in Figure 3(d). Items on the left (e.g., a, b, c, d, e) are easily confused because their pairwise subjective distances are small. On the other hand, for items on the right (e.g., j and k) most $a_{ij} = 0$.

In reality, the number of (x_i, x_j) pairs a human participant can work on is limited too. We simulated this by controlling the size of E for synthetic datasets. For 1D dataset, as there are only 11 stimuli and therefore 55 distinct pairs, it is realistic to expect participants to examine all pairs. Therefore, for all 1D datasets we let $E = \{(i,j) \mid 1 \le i < j \le n\}$. However, there are 630 distinct pairs in each 2D dataset. We chose informative pairs with small objective distance, specifically $E = \{(i,j) \mid i < j \ \wedge \ \|x_i - x_j\|_2 \le 0.8\}$. This results in $|E| = 158$ pairs for each 2D dataset.

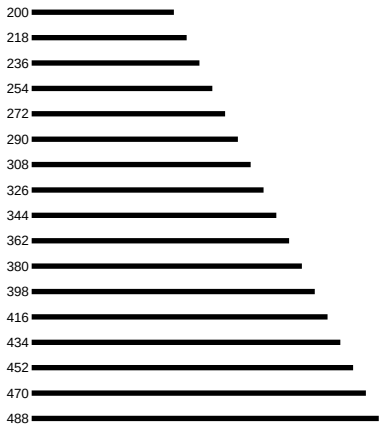


Figure 6: The line stimuli and their x values

4.1.2 Real Datasets

We conducted three real 2AFC experiments on different types of stimuli. The participants are undergraduate students from the University of Wisconsin-Madison, participating for partial course credit.

1. *Lines*. The stimuli consist of line segments with different lengths, see Figure 6. The lengths range from 200 to 488 pixels in steps of 18 pixels, resulting in $n = 17$ distinct stimuli. All $|E| = 136$ pairs are tested. 76 participants responded to all pairs, thus $n_{ij} = 76$ for all $(i, j) \in E$. Figure 5(b) shows its confusion matrix $[a_{ij}]$, where the stimuli are ordered in increasing lengths.

2. *Blobs*. The stimuli are novel computer-generated shapes parametrized by a single parameter $x \in \mathbb{R}$. The shapes change with x smoothly in several aspects simultaneously. Figure 7 shows a few blobs and their x values. The range of x in the experiment is [-2.00, 2.00] and the difference of x between two neighbor stimuli is 0.25. There are $n = 17$ distinct stimuli and $|E| = 136$ pairs. 54 participants

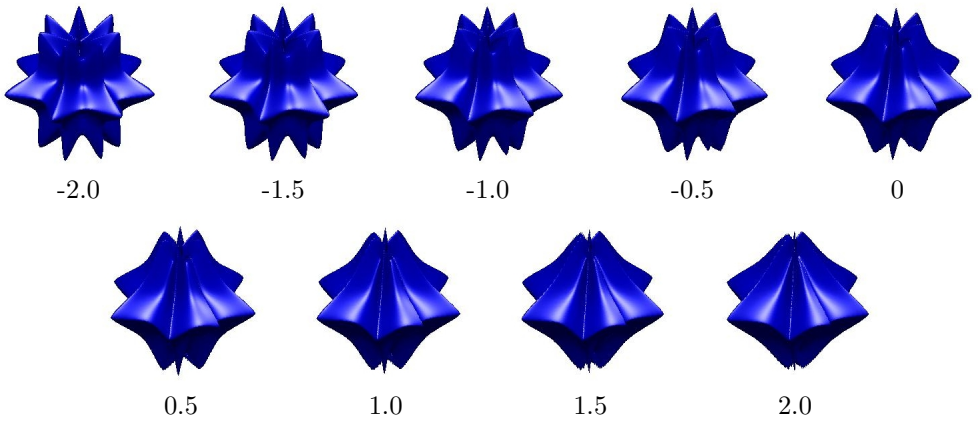


Figure 7: Selected blob stimuli and their x values

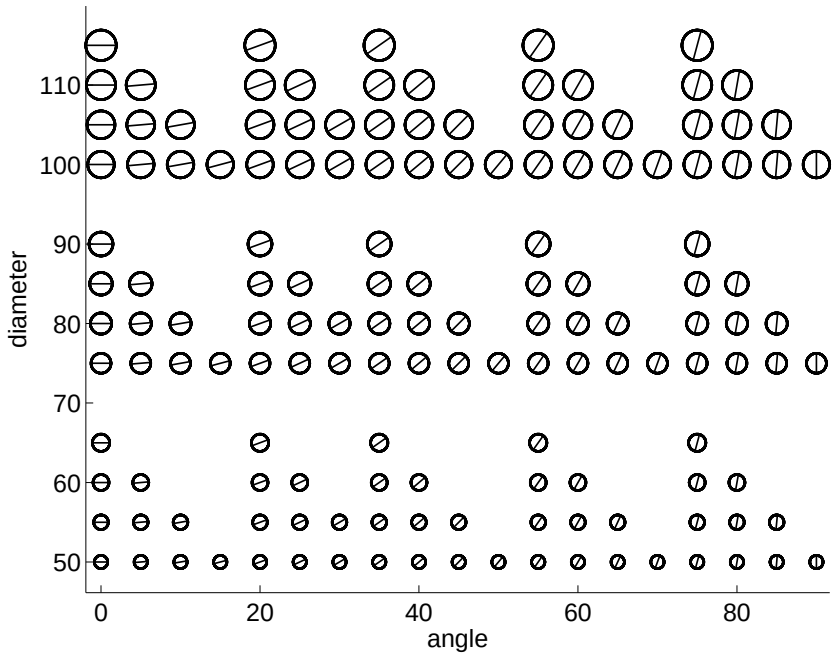


Figure 8: The screw stimuli and their $(diameter, angle)$ values

responded to all pairs ($n_{ij} = 54$).

3. *Screws.* Each stimulus consists of a circle and a line inside, and can be described by two parameters $(diameter, angle)$, the diameter of the circle and the angle of the line. Thus, the objective space is $\mathbb{R}^2_+$. To reduce total number of pairs, we selected $n = 147$ distinct stimuli to experiment with. Figure 8 shows all the screw stimuli and their $(diameter, angle)$ values used in the experiment. 15 of these stimuli were selected as anchors with $diameter \in \{50, 75, 100\}$ and $angle \in \{0, 20, 35, 55, 75\}$. For each anchor (d, a), 9 other stimuli were compared against it. These 9 stimuli were selected by $\{(d', a') \mid (d' \geq d) \wedge (a' \geq a) \wedge (d' - d + a' - a \leq 15)\}$. Therefore, there are $|E| = 135$ pairs of stimuli tested. 68 participants responded to all 135 pairs ($n_{ij} = 68$).

4.2 Evaluation Criteria

It is impossible to directly measure how close the learned mapping $\hat{\phi}$ is to the ground truth mapping ϕ on real datasets. Therefore, we define two criteria to evaluate different algorithms:

1. Normalized log-likelihood ℓ. If the learned mapping $\hat{\phi}$ is close to the ground truth, it should achieve high log likelihood on the observed data. The normalized log-likelihood on a set of pairs E is defined as

$$\ell(\hat{\phi}) \equiv \frac{1}{|E|} \sum_{(i,j)\in E} a_{ij} \log \hat{p}_{ij} + b_{ij} \log(1 - \hat{p}_{ij}) \tag{41}$$

where

$$\hat{p}_{ij} = \psi(\|\hat{\phi}(x_i) - \hat{\phi}(x_j)\|) \tag{42}$$

and E denote a set of pairs we evaluate on. In the following experiments, we often split the available pairs into training, tuning and test sets, and E refers to the appropriate set that should be obvious from context.

2. Recovery error $\mathcal{E}$ of pairwise distances in the subjective spaces. One expects the learned subjective distance $\|\hat{\phi}(x_i)-\hat{\phi}(x_j)\|$ to be close to the true subjective distance $\|\phi(x_i)-\phi(x_j)\|$. However, comparing them is only feasible and sensible on synthetic datasets that were produced with the Gaussian choice model ψ_G (6). For these synthetic datasets, we define the recovery error of pairwise distances as

$$\mathcal{E}(\hat{\phi}) \equiv \frac{1}{|E|} \sum_{(i,j)\in E} (\|\phi(x_i) - \phi(x_j)\| - \|\hat{\phi}(x_i) - \hat{\phi}(x_j)\|)^2. \tag{43}$$

4.3 Experiment Procedure

We used ten-fold cross-validation to evaluate Yada and baseline algorithms (see Section 4.4). Since some baseline algorithms cannot handle out-of-sample items, all stimuli in the test set must also appear in the training set. Therefore, we did not split the items $x_1, \ldots, x_n$. Instead, we randomly split E into 10 disjoint folds. Nine of the folds formed E_{trn} and the remaining one E_{tst}. We trained each algorithm on E_{trn} and evaluated it by $\ell(\hat{\phi})$ (41) and $\mathcal{E}(\hat{\phi})$ (43) on E_{tst} (test set performance) respectively when applicable. We report the average results over 10 folds. We also plot the embedding produced by different algorithms.

Many algorithms have parameters to be set. We used five-fold cross-validation on the first training set E_{trn} to tune these parameters. We selected the parameters with the highest normalized log-likelihood on the tuning set, and applied them to all subsequent splits for that dataset. For Yada, there are two parameters: the regularization parameter μ and the RBF kernel bandwidth. We searched μ and the bandwidth jointly over the grid $\{10^{-7}, 10^{-6}, \ldots, 10^{2}\} \times \{2^{-2}, 2^{-1.5}, \ldots, 2^{3}\}$.

We implemented Yada in Matlab. We solved the optimization problems by calling *fminunc* in Matlab with the default setting except that the maximum number of iterations was increased from 400 to 700. We supplied the gradient of the objective function to speed up *fminunc*.

We conducted experiments with two different choice models ψ_e (5) and ψ_G (6). Some algorithms do not take choice models into account, therefore their outcomes with different choice models are the same. However, the choice model may affect the log-likelihood. Since we tune parameters based on log-likelihood, the optimal parameters depend on the choice model. For different choice models, we carried out the above experiment procedure separately.

4.4 Baseline Algorithms

We compared our method with several embedding and metric learning algorithms [4, 17, 27, 9]. Due to the lack of an appropriate way to assign the pairwise distances to the pairs with $a_{ij} = 0$, we did not include metric MDS as a comparison method. We briefly discuss these baseline algorithms below.

4.4.1 Non-metric MDS (NMMDS)

Non-metric MDS takes the ranking of subjective distances as input. To generate the ranking, we sorted all pairs by a_{ij}/n_{ij} in descending order. As we believe the choice model is monotonic, the order of a_{ij}/n_{ij} reflects the order of subjective distances. The pair with the smallest a_{ij}/n_{ij} value was set as rank 1, the second smallest one as rank 2, and so on. We tested NMMDS with the Matlab implementation *mdscale*.

NMMDS also requires initial coordinates in the subjective space $\mathbb{R}^p$. The default in Matlab of initializing NMMDS with metric MDS does not apply for the reason stated earlier. Instead, we initialized NMMDS in the same way as we initialize Yada, as discussed in Section 3.3.

The output of NMMDS is scale invariant because its goal is to find an embedding which preserves the ranking of distances. If NMMDS produces $\phi(x_1), \ldots, \phi(x_n)$, then any scaled version $\tau\phi(x_1), \ldots, \tau\phi(x_n)$ (with $\tau > 0$) is equivalent in terms of ranking. However, the scaling factor τ affects the pairwise distances and, in turn, the normalized log likelihood (41) and recovery error (43). To give NMMDS maximum advantage, we optimized τ by maximizing training set log-likelihood (41).

4.4.2 Schultz and Joachims (S&J)

One may compare the subjective distances in the form $\|\phi(x_i) - \phi(x_j)\| < \|\phi(x_i) - \phi(x_k)\|$ on a triple of items x_i, x_j, x_k. S&J takes such Boolean judgment results as input to learn a metric [17]. In order to turn our 2AFC experiment outcomes into this form, we transformed them as follows. For any two pairs $(i,j),(i,k) \in E$ that share one common item x_i, we compared a_{ij}/n_{ij} and a_{ik}/n_{ik}. If $\frac{a_{ij}}{n_{ij}} > \frac{a_{ik}}{n_{ik}}$, we declared that item x_i is closer to x_j than x_k, and vice versa. Comparisons with such simple ratios turn out to be effective, and we report these results here. We also explored using Fisher's exact test on $a_{ij}, b_{ij}, a_{ik}, b_{ik}$ and only considered the comparisons achieving significant level 0.05. The results were very similar and we do not report them here.

As a metric learning algorithm, S&J produces a distance metric but not an embedding in $\mathbb{R}^p$ or a mapping ϕ to that space. We followed the authors in [17] and applied MDS on the learned distances to produce an embedding. Because S&J does not involve a choice model, to make the comparison fair we also maximized normalized log-likelihood (41) by finding an optimal scaling factor τ with the same procedure as in NMMDS. As in [17], we ran S&J with SVM-light and set the parameter C at default value 1. We used the RBF kernel and the same bandwidth parameter tuning procedure as in Yada.

4.4.3 Xing et al. (XNJR)

One may make a Boolean judgment on whether two items x_i, x_j are "similar." The XNJR algorithm [27] takes such judgments and learns a metric. To convert our 2AFC experiment outcomes into such judgments, we used the following procedure. Intuitively, if most participants failed to distinguish a pair of items, these two items are similar to each other. Therefore, we threshold a_{ij}/n_{ij} – if it is greater than a threshold, the pair of items x_i and x_j are similar. Otherwise, they are dissimilar. If $(i,j) \notin E$, the relation between x_i and x_j is unknown. This procedure depends critically on the threshold. In our experiments, we selected the threshold from $\{0.05, 0.1, \ldots, 0.5\}$ with cross validation.

In [27], the algorithm has not been kernelized and will not handle our synthetic datasets. To make a fair comparison, we followed the idea of kernel PAC, compute the Gram matrix in advance, and take the columns as input feature vectors to the XNJR algorithm. We also optimized the scaling factor τ as we did for NMMDS to improve its performance. We used the same RBF kernel and tuned the threshold and kernel parameter jointly as we did for Yada.

4.4.4 Globerson & Roweis (G&R)

G&R [9] is another embedding algorithm that works on the same similarity judgment data as XNJR. We converted our 2AFC data in the same way. There are both parametric and non-parametric embedding methods in [9]. Since our goal is to learn the mapping, we compared with the parametric version KPSDE. There are three parameters to be set in the G&R algorithm and, similar to XNJR, a threshold is needed to transform the data. We tuned them by cross validation. First, we fixed the weight of the trace λ and the weight of penalty on constraint violations β to 1, and tuned the threshold and kernel bandwidth with 5-fold cross validation. Then, we fixed the two parameters and tuned λ and β from the candidate set $\{10^{-4}, 10^{-2}, \ldots, 10^{4}\}$ by 5-fold cross validation. We also used the same scaling method as we did for NMMDS to maximize its performance.

4.5 Results

The experiment results in this section demonstrate the advantage of Yada over the baseline algorithms in learning an objective-space-to-subjective-space mapping from 2AFC data. We present two kinds of results: Tables 1, 2, 3 show the test-set (E_{tst}) performance of all algorithms on synthetic and real datasets, while Figures 9, 10, 11 visualize the learned mappings of these algorithms. In all tables, the best algorithm for each dataset (i.e., each row) is marked with boldface.

Table 1 reports the normalized log likelihood $\ell(\hat{\phi})$ on the test set E_{tst}. Yada achieves the highest test-set log-likelihood on all synthetic and real datasets (with ties). The baselines NMMDS, S&J, XNJR, and

Table 1: Main results: test-set log-likelihood $\ell(\hat{\phi})$. Algorithms here use the Gaussian choice model ψ_G, which matches the generative process of the synthetic datasets.

	Yada	NMMDS	S&J	XNJR	G&R
1D-RBF	**-0.52**	-0.55	**-0.52**	-0.53	**-0.52**
1D-cos	**-0.49**	-0.63	-0.64	-0.63	-0.62
1D-exp	**-0.13**	-0.21	-0.14	-0.16	-0.15
2D-RBF	**-0.34**	-0.36	-0.36	-0.40	-0.36
2D-tanh	**-0.27**	-0.31	-0.34	-0.36	-0.31
2D-poly	**-0.52**	-0.53	-0.54	-0.56	-0.55
Lines	**-0.55**	**-0.55**	**-0.55**	**-0.55**	**-0.55**
Blobs	**-0.44**	-0.45	**-0.44**	**-0.44**	**-0.44**
Screws	**-0.57**	-0.61	-0.62	-0.60	-0.61
Average	**-0.43**	-0.47	-0.46	-0.47	-0.46

Table 2: Test-set recovery error $\mathcal{E}(\hat{\phi})$ on synthetic datasets. Algorithms use the Gaussian choice model ψ_G.

	Yada	NMMDS	S&J	XNJR	G&R
1D-RBF	**0.01**	0.06	**0.01**	0.03	**0.01**
1D-cos	**0.01**	0.41	0.50	0.46	0.44
1D-exp	**5.49**	11.11	11.06	9.74	9.92
2D-RBF	**0.01**	0.10	0.17	0.31	0.12
2D-tanh	**0.13**	0.53	1.82	1.92	1.17
2D-poly	**0.00**	0.03	0.06	0.11	0.09
Average	**0.92**	1.99	2.22	2.05	1.89

Table 3: Test-set log-likelihood $\ell(\hat{\phi})$ is not very sensitive to mismatch in the choice model. Algorithms here use the exponential choice model ψ_e instead of ψ_G.

	Yada	NMMDS	S&J	XNJR	G&R
1D-RBF	**-0.54**	-0.56	**-0.54**	**-0.54**	**-0.54**
1D-cos	**-0.56**	-0.60	-0.61	-0.61	-0.61
1D-exp	**-0.14**	-0.19	-0.15	-0.15	-0.15
2D-RBF	**-0.35**	-0.38	-0.37	-0.40	-0.37
2D-tanh	**-0.27**	-0.31	-0.34	-0.35	-0.32
2D-poly	**-0.53**	-0.54	-0.54	-0.54	-0.55
Lines	**-0.53**	**-0.53**	**-0.53**	**-0.53**	-0.54
Blobs	**-0.42**	**-0.42**	**-0.42**	**-0.42**	**-0.42**
Screws	**-0.57**	-0.61	-0.61	-0.59	-0.61
Average	**-0.43**	-0.46	-0.46	-0.46	-0.46

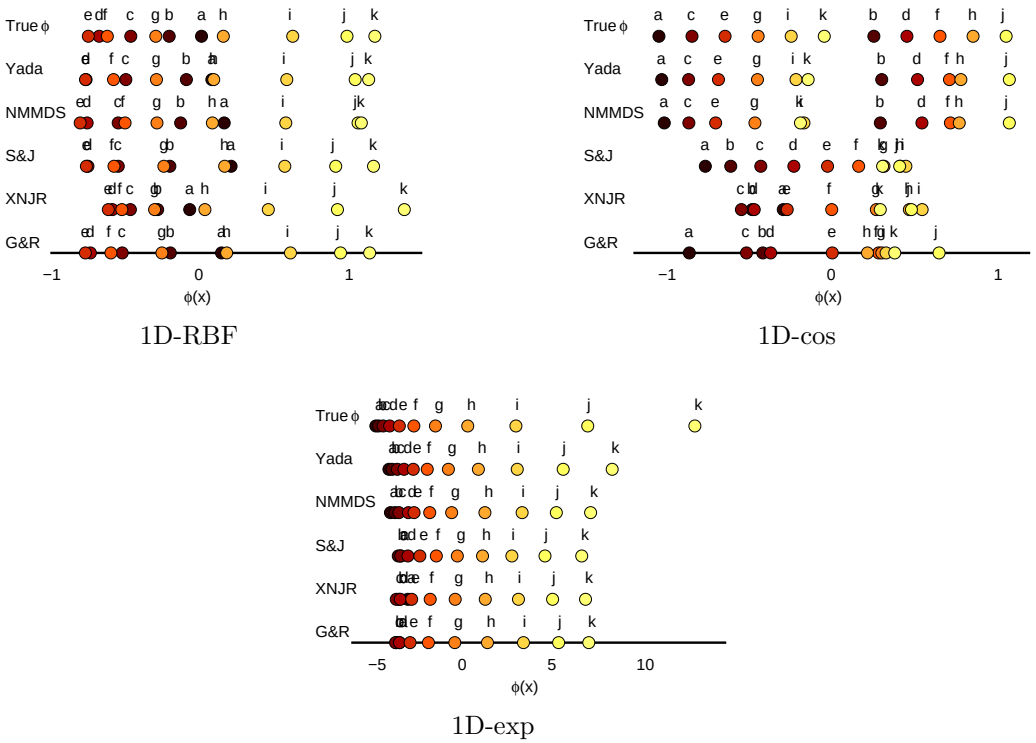


Figure 9: The learned mapping $\hat{\phi}(x_1), \ldots, \hat{\phi}(x_n)$ for 1D synthetic datasets

G&R are consistently worse than Yada, most noticeably on the synthetic datasets. We speculate that the inferior performance of these algorithms is caused by their inability to exploit all information from the 2AFC experiments. For example, S&J only considers the Boolean similarity judgments among triples. XNJR and G&R lose even more information by only considering pairwise Boolean judgments.

Table 2 reports the other evaluation criterion, recovery error $\mathcal{E}(\hat{\phi})$, on the test sets of the synthetic datasets. The table does not include the real datasets because for them the ground truth is unknown. Again, Yada achieves lower recovery error than all other methods, often very significantly. This is to be expected, given that the way the synthetic datasets were generated matches the model assumption behind Yada. Nonetheless, it serves as validation of the Yada algorithm. It is worth pointing out that all algorithms have difficulty with the synthetic dataset 1D-exp: the recovery error is very large. This can be explained by recalling that in Figure 3(d) the right-most items are far from all other items. In an 2AFC experiment, based on the Gaussian choice model ψ_G (6) these items are easily distinguished from any other items. However, this poses a problem for any algorithm using the confusion counts a_{ij}, b_{ij} directly or indirectly, because a_{ij} would always be zero in practice. Thus, the algorithms do not have enough information on precisely how far apart they should be placed.

Table 3 shows that Yada is not very sensitive to the assumption of the choice model ψ. This table is the same as Table 1 except that we used the exponential choice model ψ_e instead of the Gaussian choice model ψ_G during learning and computing the normalized log likelihood $\ell(\hat{\phi})$. Recall that the synthetic datasets were generated using the Gaussian choice model. Therefore, this represents a mismatch between data and learning models. Fortunately, the results in Table 3 suggest that the performance of Yada and baseline algorithms on synthetic datasets is not very sensitive to the choice model. This provides some assurance on applying Yada to real datasets, even though we do not know the true choice model in humans, and it is likely that neither ψ_e nor ψ_G is the correct one.

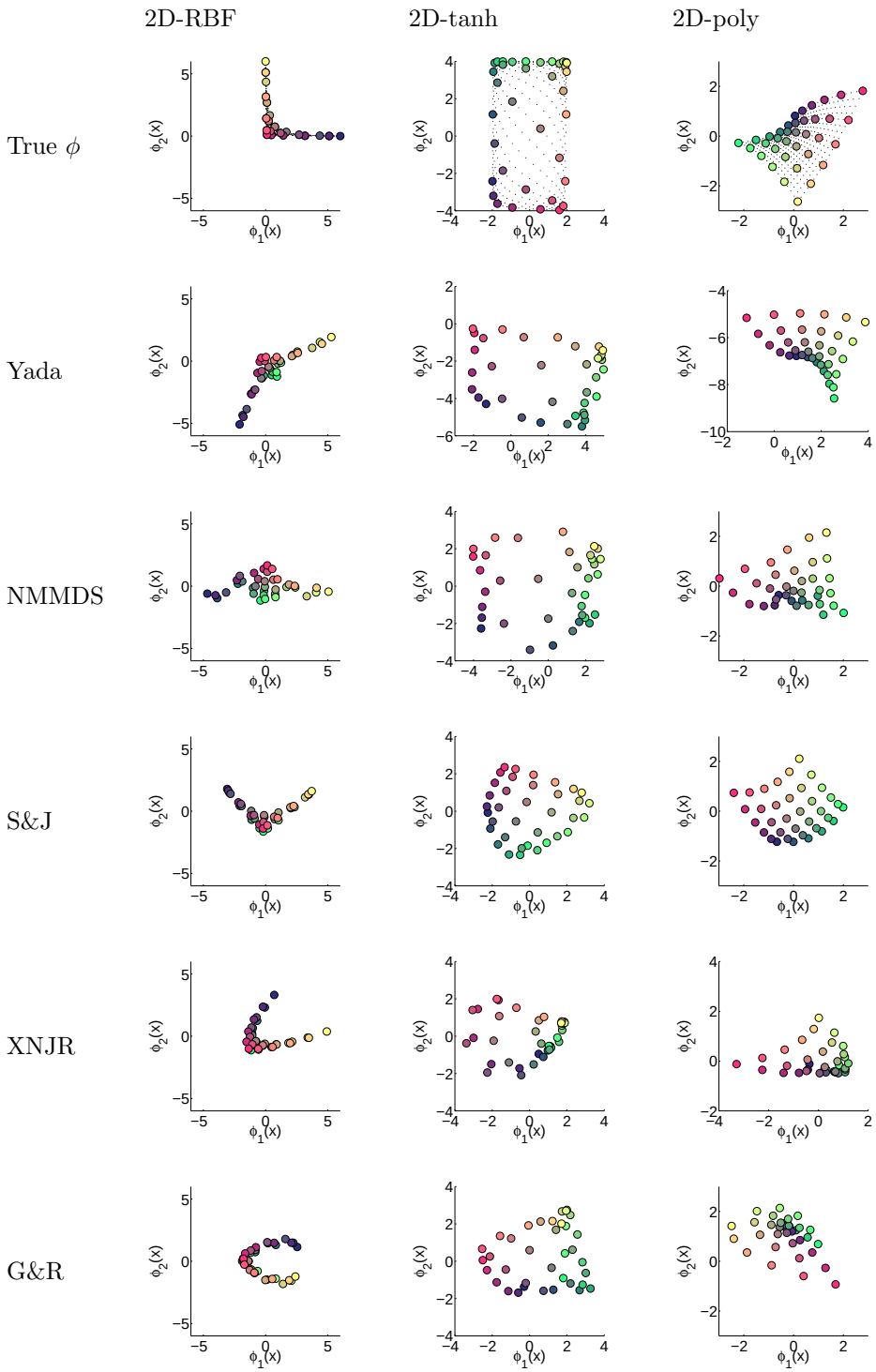


Figure 10: The learned mapping $\hat{\phi}(x_1), \ldots, \hat{\phi}(x_n)$ for 2D synthetic datasets

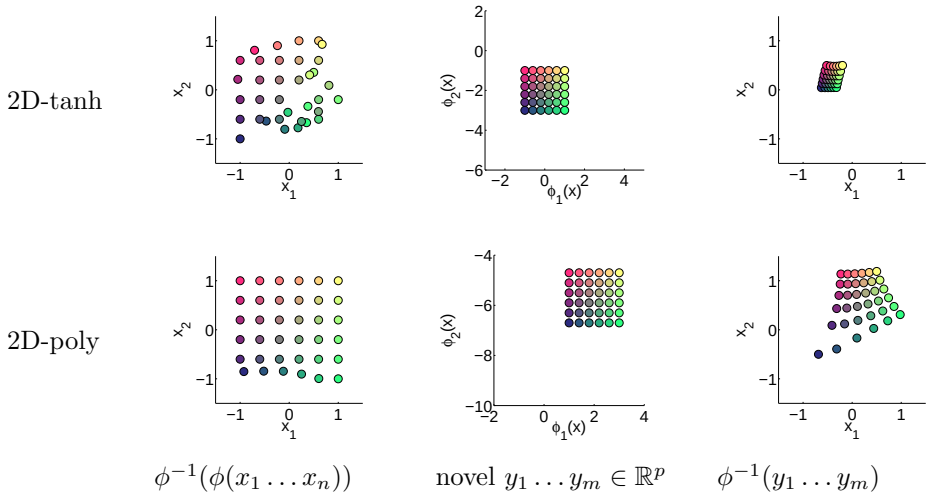


Figure 11: Inverse Mapping

Figures 9, 10 visualize the learned nonlinear mapping on the six synthetic datasets. Each plot shows the position of the training points in the subjective space $\hat{\phi}(x_1), \dots, \hat{\phi}(x_n)$ as produced by Yada and the baseline algorithms. Because the mapping $\hat{\phi}$ is shift and rotation invariant, for 1D results, we centered them $\sum_i \hat{\phi}(x_i) = 0$ and also flipped the direction when appropriate which is equivalent to rotating the mapping by 180 degrees, to make comparison easier. We did not shift or rotate the 2D results. All training items are coded by the same colors as in Figure 3 and Figure 4, so the reader can track their positions.

On all the synthetic datasets, Yada succeeds in qualitatively recovering the structure of the mapping. Most of them are close to the ground truth. On the particularly difficult 1D-exp dataset, Yada shrinks the range of items and does not place the far-right items far enough apart. However, all algorithms have the same issues due to the lack of information. We can still observe the increasing gap in Yada's result, which is not obvious for the baseline algorithms. The baseline algorithms in general produce more distorted mappings. For example, NMMDS fails to recover the L-shape on 2D-RBF; S&J, XNJR, and G&R do not recover the shuffled mapping on 1D-cos.

We are also interested in the inverse mapping $\phi^{-1}(y)$. Figure 11 shows some inverse mapping results produced by Yada on synthetic datasets 2D-tanh and 2D-poly. The first column shows the forward-then-backward mapping on the training items: $\phi^{-1}(\phi(x_1)), \dots, \phi^{-1}(\phi(x_n))$. Let $\hat{x}_i = \phi^{-1}(\phi(x_i))$. Ideally, we would like to see $\hat{x}_i = x_i, i = 1 \dots 36$ and thus a regular grid like that in Figure 4(a). But this is not always the case in Figure 11. We attribute it to the fact that ϕ may be (nearly) many-to-one and the inverse mapping can have difficulty finding a unique pre-image. Indeed, we are able to verify that $\|\phi(\hat{x}_i) - \phi(x_i)\| \approx 0, i = 1 \dots 36$.

As mentioned earlier, an application of $\phi^{-1}(y)$ is for psychologists to design a set of perceptually normalized stimuli. In other words, one starts with stimuli $y_1 \dots y_m$ on a regular grid *in the subjective space*. Examples of this is shown in the second column of Figure 11. One must compute their inverse mappings $x^* = \phi^{-1}(y)$ in order to actually create the stimuli. For instance, for the Screws x_1^* specifies the angle and x_2^* the diameter, while y, being in the subjective space does not directly specify such physical properties. The third column of Figure 11 shows that, as expected, the inverse mapping reverses the nonlinear effects shown on the first row of Figure 10. These stimuli are indeed perceptually equidistant when mapped back to the subjective space.

5 Related Work

Multi-dimensional scaling (MDS) aims at embedding a set of items into a Euclidean space such that pairwise distances are preserved. There are many varieties of the classical MDS [4]. Since it is not always possible to find an embedding in $\mathbb{R}^p$ to reproduce all pairwise distances, metric MDS will approximate the given distances as closely as possible. Unfortunately, pairwise distances are not always available in real-word applications. For example, in the 2AFC experiments such distances are not directly available.

Instead of preserving the actual pairwise distances, non-metric MDS (NMMDS) seeks an embedding to preserve their rankings. There are two major differences between our Yada model and NMMDS. First, NMMDS only embeds training items. When there are new items in the objective space, we have to re-run non-metric MDS with all available items and pairwise distances to obtain the embedding for the new items. In contrast, Yada produces a mapping function $\hat{\phi}$ which can be used to embed new items. Second, NMMDS only preserves the ranking but not the exact distances. Yada uses the choice model to convert 2AFC count data into distances.

Metric learning is another line of related work. Many metric learning algorithms aim at learning a Mahalanobis Distances $\|(x_i - x_j)^\top M(x_i - x_j)\|$ induced by a positive semi-definite matrix $M \succeq 0$ [11]. Once M is learned, it is possible to define a linear mapping with $M^{1/2}x$, or a rank q approximation to it. There are many variants of metric learning algorithms that take in different input. One kind of metric learning algorithms are purely unsupervised. There is no additional information other than the feature vectors. In this case, algorithms have been proposed to learn a mapping which preserves local distances and local structures, such as structure preserving embedding [18] and maximum variance unfolding [26]. The information to be preserved are inferred from the original feature spaces without user input. Another kind of methods take advantages of side information, other than class labels, to learn a better metric [6, 21]. For example, users may want to put some pairs closer and some pairs far away. We discuss this kind of algorithms in the following based on the type of side information.

One type of input is Boolean similarity / dissimilar judgments on a set of pairs E. A distance metric respecting such relationships, which assigns small distances to similar pairs and large distances to dissimilar ones, is preferred. Xing *et al.* formulated this problem by minimizing the sum of distances between similar pairs, with the constraint that distances between dissimilar pairs are greater than a constant [27]. Later, Bilenko *et al.* proposed a more general semi-supervised learning framework to exploit this side-information in unsupervised learning [3]. With a similar input, Globerson and Roweis proposed both non-parametric and parametric embedding algorithms to visualize the binary pairwise similarity measurements [9]. They formulated it as a semi-definite programming problem. Davis *et al.* took an information-theoretic approach by introducing a new regularizer and constraints [5].

Another type of input is the knowledge that some items are in the same class, although their actual class labels are unknown. Bar-Hillel *et al.* proposed Relevant Component Analysis to learn metric from such equivalence constraints [2]. Tsang *et al.* kernelized this algorithm in [24]. Under the supervised setting, Goldberger *et al.* [10] and Weinberger & Saul [25] learned a Mahalanobis distance to maximize the performance of nearest neighbor classifications. With the same intuition, Globerson and Roweis proposed learning a metric to collapse classes, which maps all items in the same class to a single point and pushes other items infinitely far away [8].

Yet another type of input is relative distance judgments. For example, a relative judgment over a triple has the form that item x_j is more similar to x_i than x_k is to x_i [17]. Such comparison information is used as constraints which the learned distance metric should obey. As another example, McFee and Lanckriet proposed a parametric embedding algorithm with multiple kernels on quadruples, with the form that the distance between x_i and x_j is smaller than the one between x_k and x_l [12]. Tamuz *et al.* introduced an learning algorithm which adaptively chooses triplet-based relative-similarity queries to learn similarity with as few queries as possible, since judgments are usually expensive to obtain.

From this perspective, Yada also produces a Mahalanobis distance. However, one major difference between Yada and the other algorithms is that Yada incorporates a natural rank-q constraint. In addition, Yada is designed to take the 2AFC outcomes as input. The other algorithms would have to undergo lossy preprocessing in order to work with the 2AFC outcomes. Furthermore, Yada provides a simple method to

compute the backward mapping ϕ^{-1}.

6 Conclusions and Future Work

In this paper we presented Yada, a novel algorithm for finding a mapping from an objective to a subjective similarity space, or vice versa, which is a central problem in psychology. We formulated Yada as a general metric learning problem by maximizing a regularized log-likelihood on 2AFC experiment outcomes. Yada not only learns the relation between objective distances and subjective distances, but also produces forward and backward mappings between the two spaces. These mappings provide convenient tools to help psychologists understand human cognitive process like learning and categorization, and to design behavioral experiments assessing these abilities. We conducted experiments on synthetic and real datasets to compare the performance of Yada with several baseline algorithms. The results show that Yada can best utilize the information from 2AFC experiments and recover the underlying mapping ϕ.

One future direction to enhance our algorithm is to generate one-to-one mappings. For some types of stimuli, we expect the mappings to be one-to-one. Currently, the mapping generated by Yada may be many-to-one. Furthermore, we are also interested in mappings with monotonic properties. For example, if length is one of the features in the objective space, we may want the mapping to be monotonic in length. This extra constraint may help us avoid overfitting and produce more interpretable results.

Some prior work attempted to infer a metric back from MDS solutions. One example is the ISOMAP algorithm [23], where a neural network was trained to perform the mapping from the input space to the embedded space, once coordinates in the embedded space had been identified. The ISOMAP approach is capable of generating "subjective" distances for novel items situated in an "objective" space. In future work it would be interesting to compare the behavior of ISOMAP to our approach.

Since human participants can only make a limited number of judgments, carefully choosing informative pairs to experiment with is very important. Currently, our method passively accepts 2AFC experiment results designed by psychologists as input. If it can adaptively choose informative pairs as in [22], the algorithm may achieve better performance with the same number of pairs.

7 Acknowledgments

We would like to thank Joseph Harrison, Bryan Gibson, and Charles Kalish for their assistance in conducting the human 2AFC experiments. We are grateful to Thorsten Joachims, Amir Globerson, and Eric P. Xing for their kind instructions and source code for implementing their algorithms. We thank the anonymous reviewers for their extremely helpful comments. The work is support in part by NSF IIS-0916038, NSF IIS-0953219, and AFOSR FA9550-09-1-0313.

References

[1] A. Argyriou, C. Micchelli, and M. Pontil. When is there a representer theorem? Vector versus matrix regularizers. *Journal of Machine Learning Research*, 10(Nov):2507–2529, 2009.

[2] A. Bar-Hillel, T. Hertz, N. Shental, and D. Weinshall. Learning a mahalanobis metric from equivalence constraints. *Journal of Machine Learning Research*, 6(Jun):937–965, 2005.

[3] M. Bilenko, S. Basu, and R. J. Mooney. Integrating constraints and metric learning in semi-supervised clustering. In *Proceedings of the 21st International Conference on Machine Learning*, ICML '04, pages 11–19, New York, NY, 2004. ACM.

[4] I. Borg and P. J. F. Groenen. *Modern Multidimensional Scaling: Theory and Applications*. Springer, New York, NY, second edition, 2005.

[5] J. V. Davis, B. Kulis, P. Jain, S. Sra, and I. S. Dhillon. Information-theoretic metric learning. In *Proceedings of the 24th International Conference on Machine Learning*, ICML '07, pages 209–216, New York, NY, 2007. ACM.

[6] T. De Bie, M. Momma, and N. Cristianini. Efficiently learning the metric with side-information. In R. Gavald, K. Jantke, and E. Takimoto, editors, *Algorithmic Learning Theory*, volume 2842 of *Lecture Notes in Computer Science*, pages 175–189. Springer Berlin / Heidelberg, 2003.

[7] G. Fechner. *Elemente der Psychophysiks (Elements of Psychophysics)*. 1860.

[8] A. Globerson and S. Roweis. Metric learning by collapsing classes. In Y. Weiss, B. Schölkopf, and J. Platt, editors, *Advances in Neural Information Processing Systems 18*, pages 451–458. MIT Press, Cambridge, MA, 2006.

[9] A. Globerson and S. Roweis. Visualizing pairwise similarity via semidefinite programming. In *Proceedings of the 11th International Conference on Artificial Intelligence and Statistics*, pages 139–146, San Juan, Puerto Rico, 2007.

[10] J. Goldberger, S. Roweis, G. Hinton, and R. Salakhutdinov. Neighbourhood components analysis. In L. K. Saul, Y. Weiss, and L. Bottou, editors, *Advances in Neural Information Processing Systems 17*, pages 513–520. MIT Press, Cambridge, MA, 2005.

[11] P. Mahalanobis. On the generalized distance in statistics. In *Proceedings of the National Institute of Science, Calcutta*, volume 12, pages 49–55, 1936.

[12] B. McFee and G. Lanckriet. Partial order embedding with multiple kernels. In *Proceedings of the 26th Annual International Conference on Machine Learning*, ICML '09, pages 721–728, New York, NY, 2009. ACM.

[13] R. M. Nosofsky. Overall similarity and the identification of separable-dimension stimuli- a choice model analysis. *Perception and Psychophysics*, 38:415–432, 1985.

[14] R. M. Nosofsky. Similarity scaling and cognitive process models. *Annual Review of Psychology*, 43(1):25–53, 1992.

[15] R. M. Nosofsky. The generalized context model: An exemplar model of classification. In E. M. Pothos and A. J. Willis, editors, *Formal Approaches in Categorization*. Cambridge University Press, Cambridge, UK, 2011.

[16] B. Schölkopf, R. Herbrich, and A. J. Smola. A generalized representer theorem. In *Proceedings of the 14th Annual Conference on Computational Learning Theory*, COLT '01, pages 416–426, London, UK, 2001. Springer-Verlag.

[17] M. Schultz and T. Joachims. Learning a distance metric from relative comparisons. In S. Thrun, L. Saul, and B. Schölkopf, editors, *Advances in Neural Information Processing Systems 16*. MIT Press, Cambridge, MA, 2004.

[18] B. Shaw and T. Jebara. Structure preserving embedding. In *Proceedings of the 26th Annual International Conference on Machine Learning*, ICML '09, pages 937–944, New York, NY, USA, 2009. ACM.

[19] R. N. Shepard. Stimulus and response generalization: Tests of a model relating generalization to distance in psychological space. *Journal of Experimental Psychology*, 55(6):509–523, 1958.

[20] R. N. Shepard. Toward a universal law of generalization for psychological science. *Science*, 237:1317–1323, 1987.

[21] L. Song, A. Smola, K. Borgwardt, and A. Gretton. Colored maximum variance unfolding. In J. Platt, D. Koller, Y. Singer, and S. Roweis, editors, *Advances in Neural Information Processing Systems 20*, pages 1385–1392. MIT Press, Cambridge, MA, 2008.

[22] O. Tamuz, C. Liu, S. Belongie, O. Shamir, and A. Kalai. Adaptively learning the crowd kernel. In L. Getoor and T. Scheffer, editors, *Proceedings of the 28th International Conference on Machine Learning*, ICML '11, pages 673–680, New York, NY, USA, June 2011. ACM.

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

[24] I. Tsang, P. Cheung, and J. Kwok. Kernel relevant component analysis for distance metric learning. In *Proceedings of 2005 IEEE International Joint Conference on Neural Networks*, volume 2, pages 954–959. IEEE, 2005.

[25] K. Weinberger and L. Saul. Distance metric learning for large margin nearest neighbor classification. *Journal of Machine Learning Research*, 10(Feb):207–244, 2009.

[26] K. Q. Weinberger, F. Sha, and L. K. Saul. Learning a kernel matrix for nonlinear dimensionality reduction. In *Proceedings of the 21st International Conference on Machine earning*, ICML '04, pages 106–113, New York, NY, USA, 2004. ACM.

[27] E. P. Xing, A. Y. Ng, M. I. Jordan, and S. Russell. Distance metric learning with application to clustering with side-information. In S. T. S. Becker and K. Obermayer, editors, *Advances in Neural Information Processing Systems 15*, pages 505–512. MIT Press, Cambridge, MA, 2003.