McGill University

Mechanism Design

for

Distance Metric Learning

Hossein Aboutalebi

Supervisor: Yang Cai

November 20, 2017

Abstract

Metric learning is an important area of machine learning, in which a similarity measure between sets of objects is learned from data. In a modern context, this data may come from crowd-sourcing problem instances to a set of users (or workers), who are likely to have different abilities and levels of commitment to the task. In this thesis, we first try to look over some of the main methods developed in this field of machine learning. Finally, we present a mechanism design approach for incentivizing workers to produce accurate data for a metric learning task. We show how to incorporate the data provided using this mechanism into a metric learning algorithm, and establish the theoretical properties of this approach. Results on some simulated problems show the promise of the algorithm.

résumé

L'apprentissage métrique est un domaine important de l'apprentissage automatique, sur lequel une mesure de similarité entre des ensembles d'objets est apprise à partir de données. Dans un contexte moderne, ces données peuvent provenir de certains cas de problèmes d'externalisation ouverte à un ensemble d'utilisateurs (ou de travailleurs), qui sont susceptibles d'avoir des capacités et des niveaux d'engagement différents pour la tâche. Dans cette thèse, nous essayons d'abord d'examiner quelques-unes des principales méthodes développées dans ce domaine de l'apprentissage automatique. Enfin, nous présentons une approche de conception de mécanisme pour inciter les travailleurs à produire des données précises pour la tâche d'apprentissage métrique. Nous présentons comment incorporer les données révélées en utilisant ce mécanisme dans un algorithme d'apprentissage métrique et établir les propriétés théoriques de cette approche. Les résultats sur certains problèmes simulés révèlent la promesse de l'algorithme.

Contents

1 Introduction **6**

1.1 Outline 8

2 Distance Metric **9**

2.1 Mahalanobis metric learning with application for clustering 11

2.2 Large margin nearest neighbor 13

2.2.1 Large margin nearest neighbor Loss Function 14

2.3 Information-theoretic metric learning 16

2.4 Empirical illustration of metric learning 19

2.5 Discussion 20

3 Mechanism Design **22**

3.1 Game Theory 22

3.1.1 Mechanism 24

4 Mechanism Design In Machine Learning **27**

4.1 Optimum statistical estimation with strategic data sources 27

4.2 Incentive compatible regression learning 30

5 Metric Learning with Strategic Workers **34**

5.1 Introduction 34

5.2 Our Mechanism 37
5.2.1 Incentives 37
5.2.2 Design of Our Mechanism 38
5.3 Metric Learning Algorithm 49
5.4 Empirical illustration 51

6 Conclusion **59**
6.1 Future Directions 60

1 Introduction

In many machine learning algorithms, such as k-means, nearest neighbor, outlier detection, etc., we need to assess the distance among instances. If the data contains different types of attributes (or features), simple metrics such as Euclidian distance might be insufficient. In such cases, it may be useful to learn a metric from data, which weighs differently the different attributes.

The metric learning problem was first discussed in [42], who showed that the results of algorithms such as k-means improve significantly if the metric is defined using a set of parameters learned from data. They used as input data in which pairs of instances are labeled as similar or dissimilar. Given this training data, a convex optimization problem can be formulated to find a Mahalanobis distance metric, defined as: $d_M(x_1, x_2) = \sqrt{(x_1 - x_2)^T M (x_1 - x_2)}$, where M is symmetric positive semi-definite matrix. The optimization requires that the similar instances are closer in metric space than dis-similar instances. The original work used a projected gradient optimization method to solve the optimization problem.

Since this initial work, several other approaches have been proposed for essentially the same metric learning problem. For example,[19] attempt to optimize the expected leave-one-out error of a stochastic nearest neighbor classifier. *Information theoretic metric learning* (ITML) work [16] uses the LogDet divergence to formulate the optimization problem. They also give a new interpretation of metric learning as minimizing the KL divergence of two multivariate Gaussian distribu-

tions. More recently, the *large margin nearest-neighbor* (LMNN) algorithm [41] extended the original metric learning approach, in the case in which labelled classification data is available, using the intuition that each instance in the training data should be surrounded by at least k instances with the same label; the resulting optimization problem can be solved by semi-definite programming. The work of *Pseudo-metric online learning algorithm* (POLA) [35] was the first metric learning algorithm for online tasks. [9] and Nonlinear *Neighbourhood component analysis* (NCA) [32] use nonlinear function approximators for metric learning. For more information about different models used in metric learning see, e.g., [4], [22], [43]. From now on, we will focus on learning Mahalanobis distances. In the first part of this thesis, we are going to review some of these methods.

In recent years, crowd sourcing has become an interesting way to obtain data for machine learning algorithms [17]. But, in such cases, participants may need to be incentivized in order to ensure that the obtained data set is of sufficiently good quality. Our goal in the second part of this thesis is to provide a mechanism for incentivizing workers that will produce data useful for metric learning. Our work builds directly on a mechanism designed for linear regression [8]. Their mechanism (which we denote OSE) is similar in flavor to VCG auctions [38, 10, 20], in that it makes the payment for a worker dependent on the quality of the data provided by the other workers. While other mechanisms for crowd sourcing have been proposed, eg. [36] for community sensing data collection, the OSE mechanism is most useful for us as a starting point, because its theoretical results apply to a broad range of regression problems (linear and polynomial regression, kernel regression, and some forms of regularization). In this thesis, we show that the main ideas of this mechanism also apply to metric learning. Then, we outline a metric learning algorithm using data provided by incentivized workers and illustrate its favorable accuracy-effort trade-off on some simulated tasks.

Throughout this thesis, to avoid any notation confusion, we have used capital P to denote the probability of variable and small p to denote its payment function.

1.1 Outline

The thesis structure is as follows: In chapter 2, we give a brief overview of the metric learning problem and explain some of the concepts used in this literature. We, then, look over three famous distance metric learning algorithms developed in this area: MMC [42], LMNN [41], ITML [16]. Finally, we compare these three methods with each other and Euclidean metric to see their impact on some real data sets.

In chapter 3, we give an introduction to the basic concepts of game theory and mechanism design.

In chapter 4, we discuss two research works OSE [8] and Incentive compatible regression learning (ICRL) [17] that have tried to combine mechanism design with machine learning algorithm for the task of crowd-sourcing.

In chapter 5, we propose our mechanism design method which is specifically designed for the task of crowd-sourcing in metric learning problems. At the end of this chapter, we provide some experimental results of our mechanism on some simulated environments.

Finally, in chapter 6, we provide a conclusion of the overall work in this thesis and give some guidelines for possible future works.

2 Distance Metric

In some problems, we need to calculate the distance between given inputs in order to do some analysis. These kinds of problems arise mostly in pattern recognition branches of machine learning such as k-NN, k-means [5],[14]. In k-NN, for example, we are going to classify the input based on the most popular label among its k nearest neighbors in the training set [14], [5], [34]. So, there is a need to find a distance metric between examples. In this regard, for instance, if our task is to classify images based on their quality and content, it does not sound sensible to use the same metric for both tasks. This is because the attributes that determine the quality of an image (like the number of pixels per index) might be very different from the attributes (like the distribution color of pixels) used to identify its content. The metric learning is a relatively new branch introduced in machine learning to address these problems. It was first formally proposed by [42]. Since then, it has received lots of attention and many different methods have been developed.

To define a valid distance metric, we use the same definition used in [41], [42]. Consider a vector space X. A mapping $D : X \times X \rightarrow R_0^+$ is called a metric if it satisfies the following expressions for an arbitrary vectors $\vec{x}_i, \vec{x}_j, \vec{x}_k \in X$ [41], [42]:

- **Triangular Inequality:** $D(\vec{x}_i, \vec{x}_j) + D(\vec{x}_j, \vec{x}_k) \geq D(\vec{x}_i, \vec{x}_k)$
- **Non-Negativity:** $D(\vec{x}_i, \vec{x}_j) \geq 0$

- **Symmetry:** $D(\vec{x}_i, \vec{x}_j) = D(\vec{x}_j, \vec{x}_i)$
- **Distinguishablity:** $D(\vec{x}_i, \vec{x}_j) = 0 \iff \vec{x}_i = \vec{x}_j$

If the metric does not satisfy Distinguishability but satisfies others, it is called pseudo-metric [41]. In most of the metric learning methods developed so far, they try to find a pseudo-metric. One well-known metric is the Euclidean Metric.

Definition: (Euclidean Metric): for a given vectors $\vec{x}_i, \vec{x}_j$ the Euclidean Metric is defined as follows:

$$D_{Euclidean}(\vec{x}_i, \vec{x}_j) = \sqrt{(\vec{x}_i - \vec{x}_j)^T(\vec{x}_i - \vec{x}_j)} = \|\vec{x}_i - \vec{x}_j\|_2 \tag{2.1}$$

It is easy to check that the Euclidean Metric is a valid metric as it satisfies all the four properties. Euclidean Metric is the ℓ_2-norm of the vector $\vec{x}_i - \vec{x}_j$.

The problem of using the Euclidean Metric for metric learning is that it assigns equal weights to all elements and ignores some statistical features that might exist in the data.

To better capture these features we may need to transform the original data with a linear transformation matrix L and we have [4], [19]:

$$\begin{aligned} D_L(\vec{x}_i, \vec{x}_j) &= \sqrt{(L\vec{x}_i - L\vec{x}_j)^T(L\vec{x}_i - L\vec{x}_j)} \\ &= \sqrt{(\vec{x}_i - \vec{x}_j)^T L^T L(\vec{x}_i - \vec{x}_j)} \\ &= \|L(\vec{x}_i - \vec{x}_j)\|_2 \end{aligned} \tag{2.2}$$

Note that D_L might not necessarily satisfy the distinguishability of the metric if the dimension of the kernel of a matrix is not equal to zero or $dim(ker(L)) \neq 0$.

In (2.2), if we substitute $L^T L$ with the matrix M we get:

$$D_M(\vec{x}_i, \vec{x}_j) = \sqrt{(\vec{x}_i - \vec{x}_j)^T M(\vec{x}_i - \vec{x}_j)} \tag{2.3}$$

where M is a positive semi-definite matrix ($M \succeq 0$) sometimes called the metric matrix. In (2.3), D_M is called Mahalanobis distance [4]. If we set M to be the identity matrix *I* we get back the Euclidean metric.

The goal of the metric learning is to find the best metric that discovers the underlying metric for a particular problem by using some initial information in the form of some training data. This training data might be a collection of sets containing similar or dis-similar pairs [42, 3, 16], it could be collections of triplets $\{(\vec{x}_i, \vec{x}_j, \vec{x}_k)\}_{i,j,k \in Train}$ such that $\vec{x}_i$ is more similar to $\vec{x}_j$ than $\vec{x}_k$ [4], [33] or it could be in any other general form.

Finally, in most papers of metric learning, there are two main paradigms. One paradigm begins with D_L in (2.2) as a metric and tries to find the best linear transformation of input to compute distance. The other paradigm starts with D_M in (2.3) as a metric and seeks to find the best metric matrix M. If the latter paradigm is used, usually an extra constraint of $M \succeq 0$ is considered to make sure that the final metric matrix is a valid metric. This is due to the fact that positive semi-definite constraint $M \succeq 0$ implicitly guarantees that the metric matrix has the non-negativity property:

Definition. (Positive semi-definite matrix) A matrix $M \in \mathcal{R}^{n \times n}$ is positive semi-definite if and only if [39]:

$$\vec{x}^T M \vec{x} \geq 0 \tag{2.4}$$

For all non-zero vectors $\vec{x} \in \mathcal{R}^n$. If the inequality (2.4) is strictly satisfied, the matrix is called positive definite matrix.

Usually, the methods that take the latter paradigm has been shown to be more successful like LMNN [4]. In the following sections, we are going to review some of the primary methods developed in metric learning.

2.1 Mahalanobis metric learning with application for clustering

The very first attempt to describe and solve the metric learning problems was *Mahalanobis metric learning with application for clustering* (MMC) done by [42]. In this section, we are going to take a look at their approach. They assume that initially the training set is provided in the form of two sets D, S containing

pairs of points from the input vector space X. The set S includes pairs of points that are assumed to be similar and D contains pairs of points that are dissimilar. Formally:

$$S = \{(\vec{x}_i, \vec{x}_j) | \vec{x}_i \in X, \vec{x}_j \in X, \vec{x}_i \text{ similar } to \ \vec{x}_j\} \tag{2.5}$$

$$D = \{(\vec{x}_i, \vec{x}_j) | \vec{x}_i \in X, \vec{x}_j \in X, (\vec{x}_i, \vec{x}_j) \notin S\} \tag{2.6}$$

MMC, then, tries to find a metric matrix M that is consistent with the data by assigning small distance to similar pairs and large distance on dissimilar ones. A trivial approach to satisfy these properties for M is to generate a kind of optimization problem which reflects these qualities. Indeed, MMC forms an optimization problem such that the sum of distances between the dissimilar pairs is maximized with respect to a constraint which keeps similar pairs distance lower than some threshold. The corresponding optimization problem is defined as follows:

$$g(M) = \max_{M} \sum_{\vec{x}_i, \vec{x}_j \in D} \sqrt{(\vec{x}_i - \vec{x}_j)^T M (\vec{x}_i - \vec{x}_j)} \tag{2.7}$$

$$s.t. \quad \sum_{\vec{x}_i, \vec{x}_j \in S} (\vec{x}_i - \vec{x}_j)^T M (\vec{x}_i - \vec{x}_j) \leq 1 \tag{2.8}$$

$$M \succeq 0 \tag{2.9}$$

The last constraint $M \succeq 0$ is to make sure that the output metric matrix is valid and satisfies the non-negativity property. As explained in [42], the choice of a constant c on the RHS of (2.8) is not essential, as it just changes the final M to $c^2 M$. The above optimization is convex as both the objective function and the constraints are convex. As a result, it is not going to be susceptible to fall in a locally optimum solution when trying to solve it. The optimization (2.7) is solved through projective gradient descent where a gradient step is followed by a projection onto the constraints (2.8), (2.9) respectively. Algorithm 1 is used in [42] to find M:

In Algorithm 1, C_1, C_2 corresponds to all the matrix M which satisfies the constraints (2.8), (2.9) respectively. To make sure the constraints of (2.7) are sat-

Algorithm 1 MMC Algorithm

input: Arbitrary Matrix M, α as our step size

output: MMC metric Matrix

repeat:

. **repeat**:

. M = $\arg\min_{M'}\{\|M' - M\|_F : M' \in C_1\}$

. M = $\arg\min_{M'}\{\|M' - M\|_F : M' \in C_2\}$

. **until convergence**

. M= M+$\alpha(\nabla_M g(M))$

until convergence

isfied, The gradient step is followed by a projection step to the space of solutions that are defined by equations (2.8), (2.9).

2.2 Large margin nearest neighbor

In this section, we are going to review *large margin nearest neighbor* (LMNN), one of the seminal works developed in metric learning by [41].

The MMC model tries to increase the distance between all the dissimilar pairs and imposes a constraint on all the similar pairs. As a result, MMC is imposing an a priori assumption that the clusters of points form a unimodal distribution [41]. Another problem with this approach is that the constraints are designed in a general way instead of considering the local properties of points. In this regard, in nonparametric tasks like k-NN where the labels of points are determined based on their local neighbors, MMC is unable to capture and exploit the characteristics of k-NN algorithm [41]. To solve this problem, LMNN [41, 40] takes a different approach by trying to form an optimization formulation which can reflect the specific non-parametric attributes of k-NN. LMNN also avoids making any specific assumptions on the distributions of clusters in order to provide a general solution. In this regard, LMNN proposed a new model which is designed explicitly for metric learning in k-NN classification tasks. They use a simple fact in k-NN which

assigns the label of a data point based on its k nearest neighbors. We know that if a data point's k nearest neighbors share a correct label with it, k-NN correctly classifies the data point.

Based on this simple observation, in the very beginning of their learning process, they determine a set of target neighbors for each input data $\vec{x}_i$ which remains unchanged in the future:

Definition (Target Neighbors): In k-NN, the target neighbors of a data point $\vec{x}_i$ are the k other points which are sought to become $\vec{x}_i$'s k nearest neighbor after running LMNN [41, 40].

Target neighbors of a point $\vec{x}_i$ should have the same class label with $\vec{x}_i$ so that k-NN correctly labels $\vec{x}_i$. Following the same notation used in [41], the notation $j \rightsquigarrow i$ is used to showing that $\vec{x}_j$ is the target neighbor of $\vec{x}_i$. The target neighbors of a point $\vec{x}_i$ might be given apriori; otherwise, the k target neighbors of each point is its closest k neighbors with the same class label in the ℓ_2-norm [41].

LMNN tries to push an input $\vec{x}_i$'s target neighbors close to it while keeping other inputs with different class label away. The target neighbors can be seen as creating a zone around $\vec{x}_i$ that input with different class labels should not enter [41]. In this model, to ensure that the learning process is robust against noisy input, a margin between the zone created by target neighbors and other inputs with different labels ("imposters") is maintained.

Finally, after running LMNN, we can hope that the local neighbors of points (inputs) have been filtered such that k local neighbors of a point share a similar label with it.

2.2.1 Large margin nearest neighbor Loss Function

To fulfill the mentioned goals, LMNN defines a loss function with two terms standing for the two qualities we want to see after running LMNN. One term makes imposters leave the target neighbor's zone with a margin. The other term moves target neighbors of a point (input) close to it. By following the same notation of [41], for two input data $\vec{x}_i, \vec{x}_l$ define y_{il} such that if their label is the same then $y_{il} = 1$ and $y_{il} = 0$ otherwise. Also, $[a]_+ = \max(0, a)$ denotes the hinge

loss.

LMNN defines the first term in the following way:

$$\epsilon_{push}(L) = \sum_{i,j \rightsquigarrow i} \sum_{l} (1 - y_{il})[1 + (\vec{x}_i - \vec{x}_j)^T M(\vec{x}_i - \vec{x}_j) - (\vec{x}_i - \vec{x}_l)^T M(\vec{x}_i - \vec{x}_l)]_+ \tag{2.10}$$

(2.10) imposes a force on imposters to have the margin of 1 from the zone established by each target neighbors. Note that due to the use of hinge loss in (2.10), this term only affects those differently labeled inputs that are within the zone of target neighbors plus the margin. This term is very efficiently designed in the sense that it does not make the whole optimization to deal with unnecessary differently labeled inputs when they are already outside the zone and margin. This unique property also makes LMNN distinct from MMC which deals with all the differently labeled input pairs at once.

LMNN's second term of its loss function is:

$$\epsilon_{pull}(L) = \sum_{j \rightsquigarrow i} (\vec{x}_i - \vec{x}_j)^T M(\vec{x}_i - \vec{x}_j) \tag{2.11}$$

Which pulls target neighbors of a point close to it. Again, this term only affects the target neighbors but not the whole sets of similar inputs which we saw in MMC.

The loss function of LMNN is defined as a combination of (2.10) and (2.11) with weight $w \in [0, 1]$:

$$\epsilon(L) = w\epsilon_{push}(L) + (1 - w)\epsilon_{pull}(L) \tag{2.12}$$

To solve (2.10), [41] first turns it into the form of a standard semidefinite programming (SDP) [6]. In (2.10), it contains a hinge loss in its objective; however, SDP problems should not have it. The trick used in LMNN is by introducing a slack variable ϵ for the first term of the loss function. The whole optimization model of LMNN is depicted below. Note that a slack variable is used in the loss

function of optimization instead of the second term.

$$\begin{aligned} min_M \;\; & w\sum_{j\rightsquigarrow i}(\vec{x}_i-\vec{x}_j)^T M(\vec{x}_i-\vec{x}_j)+(1-w)\sum_{i,j\rightsquigarrow i,l}\sum_{l}(1-y_{il})\epsilon_{ijl} \qquad (2.13)\\ s.t. \quad & (\vec{x}_i-\vec{x}_l)^T M(\vec{x}_i-\vec{x}_l)-(\vec{x}_i-\vec{x}_j)^T M(\vec{x}_i-\vec{x}_j)\geq 1-\epsilon_{ijl}\\ & \epsilon_{ijl}\geq 0\\ & M\succeq 0 \end{aligned}$$

Finally, we should mention that although LMNN has some clear advantage over MMC, it is especially designed for k-NN tasks.

2.3 Information-theoretic metric learning

We now discuss [16] method to find the metric matrix M. *Information-theoretic metric learning* (ITML) provides a new interpretation of finding metric matrix by viewing this problem as a problem of learning an optimal Gaussian distribution with a relative entropic objective function. One advantage of this method compared with others is that it does not need to compute eigenvalue decomposition of a matrix which might be computationally expensive. To bypass eigenvalue decomposition, ITML uses a special form of Bregman Divergence [7] (LogDet Divergence) in its objective function. We first take a look at Bregman Divergence and then discuss ITML.

Definition (Bregman Divergence): Let ϕ be a real-valued strictly convex function with domain $S = dom(\phi) \in R^m$ where S is a convex set and ϕ is differentiable on the relative interior of S. The Bregman divergence of ϕ for any two vector $\vec{x},\vec{y}\in R^m$ is [23], [37]:

$$D_\phi(\vec{x},\vec{y})=\phi(\vec{x})-\phi(\vec{y})-(\vec{x}-\vec{y})^T\nabla\phi(\vec{y}) \qquad (2.14)$$

Note that in (2.14), if we use $\phi(\vec{x}) = \vec{x}^T\vec{x}$, we get $D_\phi(\vec{x},\vec{y}) = \|\vec{x}-\vec{y}\|_2^2$. As described in [23], The Bregman divergence can be extended to take matrices as inputs. Assuming here a strictly convex function $\phi(\mathbf{X})$ over matrix space which is also differentiable, we have:

$$D_\phi(\mathbf{X},\mathbf{Y})=\phi(\mathbf{X})-\phi(\mathbf{Y})-trace((\nabla\phi(\mathbf{Y})^T(\mathbf{X}-\mathbf{Y})) \qquad (2.15)$$

where $trace(X) = \sum_i x_{i,i}$, the sum of the diagonal elements of a given matrix. The following example shows an important usage of Bregman divergence:

Example. Consider a matrix $\mathbf{X} \in \mathcal{R}^{n \times n}$. Assume λ_i denotes the ith eigenvalue of $\mathbf{X}$. Defines $\phi(\mathbf{X}) = -\sum_i \log \lambda_i$. The Bregman divergence with respect to this ϕ is (λ_i, λ'_i denotes eigenvalue $\mathbf{X}, \mathbf{Y}$ respectively):

$$\begin{aligned} D_\phi(\mathbf{X}, \mathbf{Y}) &= trace(\mathbf{Y}^{-1}(\mathbf{X} - \mathbf{Y})) - \sum_i \log \lambda_i + \sum_i \log \lambda'_i \\ &= trace(\mathbf{X}\mathbf{Y}^{-1}) - \sum_i \log(\frac{\lambda_i}{\lambda'_i}) - n \\ &= trace(\mathbf{X}\mathbf{Y}^{-1}) - \log\left(\prod_i \frac{\lambda_i}{\lambda'_i}\right) - n \\ &= trace(\mathbf{X}\mathbf{Y}^{-1}) - \log(det(\mathbf{X}\mathbf{Y}^{-1})) - n \end{aligned} \tag{2.16}$$

In the third line of (2.16) we have used the $\log a + \log b = \log ab$ property. (2.16) is called LogDet divergence. Assuming that Y is a positive definite matrix, the LogDet divergence is finite if and only if X is also positive definite [4]. So, using LogDet divergence as a regularization term in the objective function of an optimization automatically opens a path to bypass the positive definite constraint in the optimization. In ITML, it is assumed that we are first given an initial metric matrix M_0 and a batch of training data consisting of similar and dissimilar pairs. The goal of ITML method is to find a metric matrix M such that M stays close to some initial given matrix M_0, while keeping a distance lower than a parameter u on similar training pairs and higher than a parameter v on dissimilar pairs.

ITML measures a distance between two matrices M and M_0 implicitly through exploiting KL-divergence [24] of information theory. Suppose we want to model an unknown distribution $P(x)$ with the distribution $Q(x)$ in order to transmit data x to some client. Then, the average additional information (bits) needed due to the use of the distribution $Q(x)$ can be shown to be equal to KL-divergence between the distributions $P(x)$ and $Q(x)$ [5]:

Definition (KL Divergence): The KL-divergence between the distributions

$P(x)$ and $Q(x)$ is an asymmetric relation defined by:

$$KL(P(x)||Q(x)) = \int P(x) \log \frac{P(x)}{Q(x)} dx \tag{2.17}$$

In addition to its information theoretic interpretation, (2.17) is defining a measure of closeness between two distributions $P(x), Q(x)$ [5]. In this regard, consider two Gaussian distributions $\mathcal{N}(C, M), \mathcal{N}(C, M_0)$ with equal mean C and covariance matrix M, M_0 respectively. As their mean is identical, their distance should be dependant on their covariance matrices M, M_0. As a result, we can apply (2.17) to find the distance between M, M_0 [16]. ITML uses the same idea to measure distance for different metric matrix by defining their corresponding Gaussian distributions. In fact, it uses the following KL-divergence function as its loss function to keep M close to M_0 subject to constraints on similar and dissimilar pairs:

$$KL(\mathcal{N}(\mu, M_0)||\mathcal{N}(\mu, M)) = \int \mathcal{N}(\mu, M_0) \log \frac{\mathcal{N}(\mu, M_0)}{\mathcal{N}(\mu, M)} dx \tag{2.18}$$

Though (2.18) might be a reaonable objective function, it might be very hard to solve its corresponding optimization. In this regard, it is shown in [15] that

$$KL(\mathcal{N}(\mu, M_0)||\mathcal{N}(\mu, M)) = \frac{1}{2} D_{LogDet}(M, M_0)$$

As a result, ITML minimizes the $D_{LogDet}(M, M_0)$ subject to a similar constraint on similar and dissimilar pairs. Note that in ITML there is no need to do the projection step with eigenvalue decomposition due to the use of Bregman divergence.

2.4 Empirical illustration of metric learning

After explaining MMC, LMNN, ITML algorithms, it's worth investigating whether they make any improvement compared with using simple Euclidean metric on real data sets. Here, we also want to compare the effectiveness of these algorithms on real datasets.

We applied metric learning algorithms described in the previous section on the k-NN tasks for wine quality, abalone mushroom, breast cancer and diabetic patients datasets from UCI repository [25] (Figure 1.1). Here is a brief description of these datasets (In all of these datasets we have assumed by reformatting the last attribute is the target value):

- **Wine quality:** This dataset contains two files corresponding to white and red wine. We use white wine file here. In both files, the dataset contained 12 attributes where the last attribute is the measure of the quality of the corresponding wine. The measurement is done based on physicochemical tests. For more information, please refer to [13].

- **Abalone mushroom:** This dataset contains 8 attributes where the last one is the number of rings of the corresponding mushroom which gives information about its age. For more information, please refer to [25].

- **Breast cancer:** This dataset contains 10 attributes where the last one is the state of the corresponding breast cancer which can be malignant or benign. For more information, please refer to [26].

- **Diabetic patients:** This dataset contains 20 attributes where the last one is the signs of DR. For more information, please refer to [1]

In these experiments, we chose $k = 5$ for k-NN algorithms. The test set had the size of 100 for all datasets which we separated it initially from the training data. Figure 1.1 shows the results. The error bar on figure is for five times repetition of each experiment. The x-line for the bars shows the size of the training set for learning metric matrix by LMNN, ITML, MMC receptively. The y-line of Figure

1.1 shows the number of mislabeled test data by k-NN algorithm after learning the metric matrix. Formally:

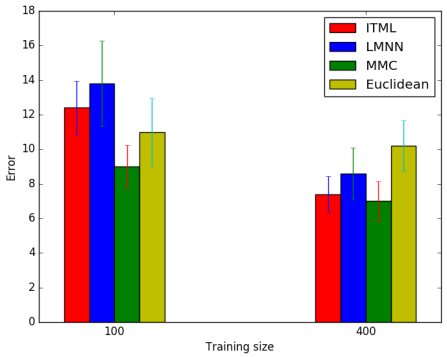
$$Error = \sum_{i \in T} \mathbf{1}_{y_i \neq \hat{y}_i}$$

Where T denotes the test set, $y_i, \hat{y}_i$ denote the actual label and the output label of k-NN for the ith test data and **1** is the indicator function which has value 1 if its condition holds and zero otherwise.

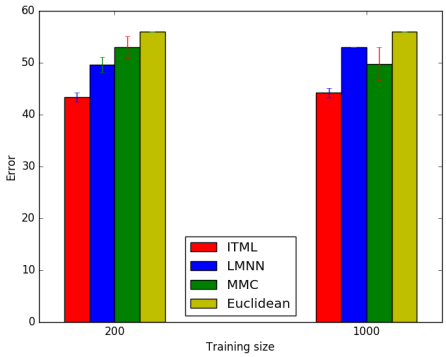
As it can be seen, when the training size increases the error decreases in almost all cases. In Figure 1.1a for Breast cancer dataset, we can see that MMC method had better performance compared with other methods. MMC also outperforms other methods in Figure 1.1c for Abalone mushroom though ITML also has competitive results on this dataset. For Figure 1.1b, d for Diabetic patients and Wine datasets, we see that ITML had better performance than others. In all the figures, we can say that all methods outperformed Euclidean metric.

2.5 Discussion

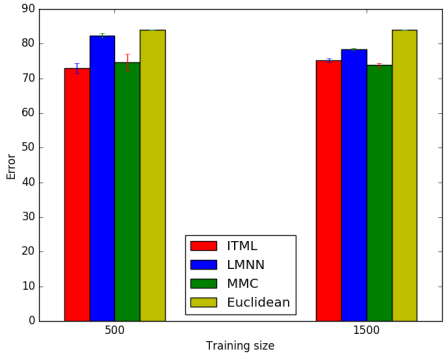
So far, we have explored some of the distance metric learning problems and concepts in machine learning area. We also discussed three of the well-known algorithms developed to address these metric problems. In the previous section, we compared the results of applying these algorithms to real data sets and found that they, indeed, have enough capacity to improve learning accuracy compared with using traditional Euclidean metric. In the next two chapters, we will explore game theory concepts and mechanism design and review some of the efforts made to combine mechanism design with machine learning algorithms in crowd-sourcing.



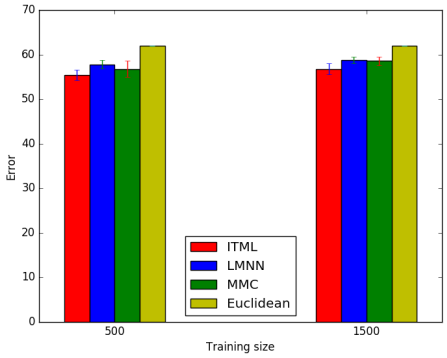
(a) Breast cancer dataset



(b) Diabetic patient dataset



(c) Abalone mushroom data set



(d) Wine data set

Figure 2.1: Summary of results of applying ITML, LMNN, MMC versus Euclidean metric on data sets. The yellow bar in figures corresponds to the Euclidean metric

3 Mechanism Design

Mechanism design is an attempt to develop a protocol such that the self-interested participants (called rational agents) maximize (or minimizes) a general objective. This area is a subfield of game theory study. In this section, we first review some fundamental notions of game theory. Then we give a formal definition of mechanism design.

3.1 Game Theory

Game theory is ”the study of mathematical models of conflict and cooperation between intelligent, rational decision-makers”[28]. By using mathematical models, it provides insights into social situations and behaviors. This social condition may involve two or many individuals such that their interactions affect their welfare (utility) [28, 31]. Game theory has been widely used in economics, political science, psychology, logic, and computer science [30, 11, 2, 28].

In this section, we are going to review some important basic notions of game theory that we are going to use later:

Definition (Game): A game is an interaction between at least two players or agents with non-identical objectives [29].

These players' behaviors are the center of focus in the game theory. The primary assumption used in the game theory context is that the players (agents) involved in a game are rational decision makers [28].

Company 1 \ Company 2	A	B
A	(-1,-1)	(10,3)
B	(3,10)	(6,6)

Table 1: showing the pay off company 1,2. (p_i,p_j) of the ith row and jth column correponds to company 1 getting pay off p_i and company 2 getting pay off p_j

Definition (rational agent): An agent is called rational if and only if it consistently chooses a decision that maximizes his objective/utility function [28, 29, 31].

In a game, it is assumed that a player has different strategies (options) $S = \{s_1, s_2, ..., s_k\}$ to play and each of these strategy may lead to an outcome in a set $O = \{o_1, o_2, ..., o_n\}$ [28, 29, 31]. The agent usually has some sort of preference over possible outcomes in a game. The sets O and S can be continuous or discrete based on the nature of the given game. In order to be able to distinguish between different strategies that an agent can use, a function is defined such that it assigns a value to the strategy chosen by the agent and this value might be dependant on the strategy chosen by other agents. This function is called utility function [29, 31]. To make these concepts more clear let's consider the following example:

Example 1. Consider two companies 1, 2 that can invest either in project A or B. Assume that project A is more profitable than project B if just one of the companies invests and creates a captive market. In this case, the marginal profit for the company investing in A is 10 versus 3 for the other company investing in project B. If both companies invest in project A due to the high manufacturing cost they will lose -1. Finally, if both companies invest in project B they will get marginal profit 6. Table 1 depicts the different strategy with its corresponding pay off for company 1 and 2. We can see that under this scheme, there is a sort of competition between companies 1 and 2 in which they try to increase their profit. Here, if both companies choose the strategy to invest in B, it would be an unstable solution as both companies have motives to go for project A. This is because they are going to get a higher pay off if they change their strategy. We can say a similar

thing when both of them choose to invest in project A. On the other hand when one of them choose to invest in A and the other one is investing in project B, we have a stable solution. This is because if any of them changes its strategy, it gets less pay off (lower utility function value) which is not a rational behavior.

A stable solution in Game Theory is called a Nash Equilibrium. We will give a more formal definition about it later.

3.1.1 Mechanism

Unlike Example 1, in the real world, we may encounter situations where agents do not have complete information about a game. Specifically, they do not have enough information about other agent's behavior. This behavior consists of private information of agents about their utility function and their preferences [29]. To model such games, we will follow the same model presented in the seminal work [29] (a similar model is also shown in [31]). In such games, it is assumed that no probabilistic information is provided to any agents about others.

Definition. A strict incomplete game with n agents consists of [29]:

- Each agent i has a set of actions X_i.
- Each agent i has a set of types T_i. it is regarded as its private information
- a utility function u_i is defined for the agent i with type set T_i with formulation $u_i : T_i \times X_1 \times X_2 \times ... \times X_n$. $X_1, ..., X_n$. (This definition reminds that the utility function of an agent is dependent on the action of other agents.)

Based on the above definition, we can give a more formal definition of strategy. The strategy of an agent is just a mapping from its type to his action set [29]. In essence, we can see strategy as a value function in machine learning which assigns numerical values to different options of an agent except in game theory a rational agent always follow an option with the highest value.

Definition (Nash Equilibrium): if we have for all agents i with $t_i \in T_i$ and all $x_i \in X_i$:

$$u_i(t_i, s_i(t_i), s_{-i}(t_{-i})) \geq u_i(t_i, x_i, s_{-i}(t_{-i})) \tag{3.1}$$

then we say the set of strategies $s_1, .., s_n$ are in Nash Equilibrium [27, 29, 31].

The notation $\{\bullet\}_{-i}$ in (3.1), refers to all other agents except i. The stable solutions we showed in Example 1 are the examples of Nash Equilibrium. When the game is in the Nash Equilibrium state, no rational agent has any incentive to change its strategy as it is not going to improve its utility.

In general, we may have the case where under any circumstances, following a specific strategy for an agent always leads to the maximum utility value. This strategy is called a dominant strategy for the agent.

Definition. (Dominant Strategy): We say a strategy s_i for agent i is a dominant strategy if for all t_i, all x_{-i} and x_i we have [27, 29, 31]:

$$u_i(t_i, s_i(t_i), x_{-i}) \geq u_i(t_i, x_i, x_{-i}) \tag{3.2}$$

Note that the main difference of Dominant Strategy with Nash Equilibrium is that (3.2) holds for any strategy chosen by other agents, while (3.1) only holds for a fixed set of strategy.

In most mechanisms, people try to give a solution which can hold in a Nash equilibrium or a dominant strategy of players (agents).

When agents take actions, the action usually has some sort of costs for them (e.g., paying an amount of money). While, in return for taking action, they will get some value from this outcome $o \in O$ (e.g., receiving payment). This value should be clearly dependent on the type of the agent which reflects agent's vote, viewpoint, choice, etc. In this regard, two functions $p_i : X_1 \times X_2 \times ... \times X_n \rightarrow \mathcal{R}$, $v_i : T_i \times O \rightarrow \mathcal{R}$ are defined to account for the cost and value functions of agent i respectively [29]. It is easy to check that the utility function of agent i is [29]:

$$u_i(t_i, x_i, x_{-i}) = v_i(t_i, o) - p_i(x_1, ..., x_n) \tag{3.3}$$

Now, under this scheme, the mechanism design is about specifying the rule of a game such that when agents take their actions what is going to be the expected outcome. A function $f : T_1 \times T_2 \times ... \times T_n \rightarrow O$ is usually used to determine the output. This function is sometimes called the social choice function or social welfare function [29, 17]. In essence, social choice function reflects the aggregate

type of agents participating in a game. As discussed in [17] we can define a notion of efficiency for social choice function in the following way:

Assume the set $T = T_1 \times T_2 \times ... \times T_n$, for a social choice function $f : T \to O$ if for all $T_i \in T$:

$$\alpha \sum_{i=1}^{n} v_i(f(T), T_i) \geq \mathbf{max}_{o \in O} \sum_{i=1}^{n} v_i(o, T_i) \tag{3.4}$$

then the social choice function is α efficient [17].

4 Mechanism Design In Machine Learning

After explaining the notions of mechanism design, in this section, we are going to see some of the attempts to interweave mechanism design and machine learning problems. The main thing addressed in these works is that they propose a way to get probably a better training data for a class of machine learning function approximations.

4.1 Optimum statistical estimation with strategic data sources

One way to improve function approximation methods is to try to get more accurate training data. In this regard, *optimum statistical estimation with strategic data sources* (OSE) [8] solves this problem by compensating data sources (workers) via monetary incentives. OSE assumes that a statistician wants to approximate a function f through a set of training data $< \vec{x}_i, y_i >_{i=1}^{n}$ provided by some workers. Each worker i is given a vector $\vec{x}_i$, and would return the corresponding label y_i with some noise. As usual, under this scheme, when the amount of effort spent by a worker increases, the accuracy of the label increases as well. A game is then developed such that each worker tries to get a better utility function for himself by exerting the optimal effort level. These workers are strategic in the sense that they try to maximize their utility function.

Formally, given a tuple $\vec{x_i}$, worker i will produce $y(e)$ by exerting effort e such that [8]:

$$y(e) = f(\vec{x}) + \delta \tag{4.1}$$

where $f(\vec{x})$ is a true value, and δ is a noise with mean equal to zero and variance $\sigma(e)^2$. The utility function of workers is dependent on the effort level they exert:

$$u_i(e) = p(\vec{x}, y(e)) - e \tag{4.2}$$

Clearly, workers aim to use the effort level that maximizes their expected utility. The payment function $p : D \times \mathcal{R} \to \mathcal{R}$ (D is the domain of input vector space $\vec{x}$) is determined by the mechanism.

OSE proposes the following objective for the statistician who wants to minimize it:

$$\mathbf{E}_{x^*, \vec{y}(\vec{e}^*)} \left[\left(\hat{f}_{\vec{y}}(x^*) - f(x^*) \right)^2 + \gamma \sum_{i \in W} p_i(\vec{x}, \vec{y}) \right] \tag{4.3}$$

In (4.3), γ is a regularizer factor, x^* is the test data drawn from some distribution over a domain. $\hat{f}$ is the approximation of the true function f which is obtained via the training data $< \vec{x}, \vec{y} >_{i=1}^{n}$.

(4.3) tries to combine the accuracy of the approximation function with the payment to make a trade-off between high accuracy and low commitment to workers. (4.3) consists of two terms inside the expectation. In order to compute the minimum of (4.3) we need to be able to have some explicit formula for both terms. The second term, p_i, the payments are determined by mechanism and it should be easy to compute. However, the first term of the expectation, $\left(\hat{f}_{\vec{y}}(x^*) - f(x^*) \right)^2$, might be tricky to compute. It turns out that for some function approximation method like linear regression, this term can be computed. These estimators are called well-behaved by OSE.

Definition (Well-Behaved Approximator): "Assume $\mathcal{H}$ is a family of functions f. An approximation $\hat{f}$ for $\mathcal{H}$ is well-behaved if and only if there exists some function g such that, for all distributions F over D our input space, functions $f \in \mathcal{H}$, and vectors $\vec{x}$ drawn from the input space and $\vec{\sigma} \in \mathbb{R}_+^*$:" [8]:

$$\mathbf{E}_{x^*,\vec{y}(\vec{e^*})}\left[\left(\hat{f}_{\vec{x},\vec{y}}(x^*) - f(x^*)\right)^2\right] = g(\vec{x}, F, \vec{\sigma}) \tag{4.4}$$

Here, x^* is the test data drawn from the distribution F. As explained previously, each y_i in (4.4) is deviated from its value by a Gaussian noise with mean 0 and the variance σ_i^2. Note that the well-behaved requirement is necessary here otherwise we are unable to compute the minimum of (4.3). As we are going to see later, the condition of well behaved approximator in [8] is kind of restrictive as we cannot in general use it in all cases. Later, we are going to mitigate this condition in our method.

Now, assuming that the approximator is well behaved, OSE proposes a mechanism where the dominant strategy of workers is to apply the optimal effort e^* which minimizes (4.3). If any of the workers applies less effort, he or she will have lower utility gain. As a result, he or she cannot get the maximum utility which is in contradiction with the definition of strategic worker. It is proven in [8] that (4.3) can provide this dominant strategy framework if the payment is defined in the following way:

$$p_i(\vec{x}, \vec{y}) = c_i - d_i(y_i - \hat{f}_{(\vec{x},\vec{y})_{-i}}(\vec{x_i}))^2 \tag{4.5}$$

where $\hat{f}_{(\vec{x},\vec{y})_{-i}}$ is the approximation function we get by eliminating the data of the ith worker from training data. c_i, d_i are the constants defined in the following way to assure the existence of dominant strategy:

$$\begin{aligned} d_i &= \frac{-1}{2\sigma_i(e^*)\sigma_i'(e^*)} \\ c_i &= d_i(\sigma_i(e_i)^2 + g(\vec{x}_{-i}, F, \vec{\sigma}_{-i}(\vec{e}_{-i}))) + e_i \end{aligned} \tag{4.6}$$

Where the $\sigma_i'(e^*)$ is the derivative of σ_i function at the point e^*. The proof of why this payment setting provide such dominant strategy is given in [8].

4.2 Incentive compatible regression learning

Incentive compatible regression learning (ICRL) [17], uses incentive compatible concept in its design of the mechanism for regression learning. In this setting, workers are holding some private information about the label of input data. This private information reflects the viewpoint of the worker over the input data. This input data plus the labels of workers constitute the training set which is going to be used later for regression learning to build a hypothesis function. The main problem here is that these workers may have a conflicting viewpoint on the inputs [17]. Thus, they might have the incentive to lie so as to make the final hypothesis function incline toward their view. As described in ICRL, the main goal is to construct a hypothesis function which reflects the common viewpoint of workers. The following example provides a good motivation for the whole problem.

Example. Consider a company asks a group of experts to give a score for a newly manufactured car. In the end, the company wants to build a score function to evaluate its future products. Now, an expert who is interested in sports cars may give a meager score for a car that has relatively moderate acceleration. The score, however, might be very unfair if the car is a family car for having moderate acceleration. As a result, we need to incentivize experts to tell the truth.

It is also possible that some subset of workers forms a coalition to deceive the whole mechanism in favor of their viewpoints. The problem with the definition of incentive compatible is that it does not cover the case where the workers may form a coalition to deceive the mechanism. A new concept is needed to cover these cases as well. This problem is also addressed in ICRL. In this framework, [17], first, defines an extension of the incentive compatible concept and then tries to build a mechanism that follows this expansion.

Definition (ϵ-group strategyproof): For an $\epsilon \geq 0$, suppose for any group $C \subset \{1, 2, ..., n\}$ of workers who want to form a coalition in order to jointly deviate from their right type T_i to get a higher pay off. Now suppose a mechanism that imposes the following condition:

All the members of C can get at least ϵ higher pay off only if all of them get exactly ϵ higher pay off .

This mechanism is called ϵ-group strategyproof in dominant strategy equilibrium [17]. Formally, for $\hat{T} \in T$ if $\hat{T}_j = T_j$ if $j \notin C$. The mechanism is called ϵ-group strategyproof if for all $i \in C$ and all $C \subset \{1, 2, ..., n\}$ the following inequality hold:

$$v_i(f(\hat{T}, T_i)) - p_i(\hat{T}) \geq v_i(f(T, T_i)) - p_i(T) + \epsilon$$

then we have:

$$v_i(f(\hat{T}, T_i)) - p_i(\hat{T}) = v_i(f(T, T_i)) - p_i(T) + \epsilon$$

for all $i \in C$ and all $C \subset \{1, 2, ..., n\}$ [17].

If the $\epsilon = 0$ then the 0-(group) strategyproof and (group) strategyproof is used interchangeably. In general, it might be even impossible to avoid any coalition of agents in a mechanism. However, by introducing the ϵ factor, The ϵ-group strategyproof provides a flexible extension to the notion of incentive compatibility which also considers the coalition. Although this new concept might not avoid coalition (when the $\epsilon > 0$), it at least provides a safety margin ϵ when it is satisfied. Note that if a mechanism is an incentive compatible, it does not necessarily mean that it is also group strategyproof. On the other hand, the reverse always holds.

Now, ICRL formulates the whole problem as follow: The general goal is to formulate some real-valued function $f : X \to R$ with domain X which reflects the average attitude of workers. As usual, $f \subset F$ where F is a hypothesis space of real-valued functions. Let the worker set $N = \{1, 2, ..., n\}$. Each worker $i \in N$ holds a private function $o_i : X \to R$ which reflects its viewpoint on the input data. Furthermore, each agent has some probability distribution ρ_i which reflects the relative importance of the points in the input space X for the agent. The accuracy of predicted function f can be its average loss over the whole domain. However, this global definition cannot be computed, as it is defined over the whole domain. Instead, ICRL works with an empirical estimate of the global risk $\hat{R}_N(f)$ as follows. For each worker i, it samples m points and asks the corresponding labels from i to form the training set $S_i = \{(\mathbf{x}_{i,j}, y_{i,j})\}_{j=1}^m$. The global training set is $S = \cup_{i=1}^n S_i$. Then, ICRL defines the empirical estimate of the global risk

$\hat{R}_N(f)$:

$$\hat{R}(f, S) = \frac{1}{|S|} \sum_{(\mathbf{x},y)\in S} l(f(\mathbf{x}), y) \tag{4.7}$$

(4.7) can be seen as the social welfare where we want to minimize it to reflect the common viewpoint of workers better. Finally, for the training set S, a learning method can return the hypothesis function which minimizes the empirical risk. This method is called ERM (Empirical Risk Minimization) [17]. As discussed in ICRL, note that under this scheme workers still have the incentive to lie about the true label of a given training input $\mathbf{x}$ when reporting y in order to minimize their own risk function:

$$R_i(f) = \mathbf{E}_{x\sim\rho_i}[l(f(\mathbf{x}), o_i(\mathbf{x}))]$$

ICRL has proved some interesting results on the ways to design the group strategyproof mechanism. We are going to explain the main result here.

The main result applies to the case where the ρ_i for all i consists of only one point which means each worker is just interested in only one point in the domain of its inputs. Furthermore, assume the loss function is the ℓ_1-norm:

$$l(a, b) = |a - b| \tag{4.8}$$

Under these circumstances, if we apply Algorithm 2 we have:

Algorithm 2 ERM with Tie Breaking Algorithm

input: set S, convex hypothesis space F

output: function $\tilde{f}$

. $r = \mathbf{min}_{f\in F}\hat{R}(f, S)$

. $\tilde{f} = \underset{f\in F,\hat{R}(f,S)=r}{\arg\min} \int f^2(x)dx$ *(This integral is over training data)*

The mechanism using Algorithm 2 is group strategyproof [17].

Here, we will not go over the whole proof of the above claim, as it might be beyond the scope of this thesis. However, we provide some highlights over the approach taken by the author of ICRL to prove their claim. First, given two training set S, S' on the same set of points, ICRL applies Algorithm 2 on each

of them to get real-valued functions f, f'. They show that if $f \neq f'$, then there exists some input $x_i \in S, S'$ such that $y_i' \neq y_i$ where y_i' is the label of this input in S' and y_i is its label in S. Also, the loss function with respect to y label for this input using f function is less than the same loss function using f'. On the next step, they use this fact to show that any coalition of agents to lie about their true type is doomed to decrease their final value function. As a result, Algorithm 2 is group strategy proof.

In this chapter, we tried to provide a review of some of the effort done to use game theory on machine learning algorithm. In the next chapter, we will take advantage of OSE method to develop a mechanism for metric learning algorithms.

5 Metric Learning with Strategic Workers

5.1 Introduction

In the previous chapter, we saw how game theory is being used in machine learning problems. The main trend here is how to elevate the accuracy of training data by making workers to tell the truth or put more effort on their task. [8], [17], [18], [36]

Our mission here is to use the similar idea in metric learning problems specifically. We want to see in metric learning whether we can use better training data by incentivizing workers. It turns out that due to the complexity of metric learning problems, we need to relax some of the notions we have explained before. In this regard, we found that the model used in OSE is a suitable model as a baseline and we are going to extend this model to metric learning problems.

Following the model used in OSE, we consider a set of strategic workers $\mathcal{W}$, who are going to provide the training data, by exerting different levels of effort. In order to be consistent with this model, we assume that given a pair of instances, a worker will provide a real-valued measure of their similarity, instead of a discrete "similar" or "dis-similar" label. While this is different from standard metric learning approaches, many datasets (e.g. opinion surveys, psychometric data) contain this type of information; we further discuss how this assumption may be lifted in conclusion.

Upon being presented with a pair of instances $\vec{x}_{i,1}$, $\vec{x}_{i,2}$ whose true distance is

y_i^*, a worker w_i will produce a label y_i such that

$$y_i = y_i^* + \epsilon_i, \tag{5.1}$$

where ϵ_i is a random variable drawn iid from a known distribution. For the following developments, we assume (as usual in regression) that this distribution is Gaussian with zero mean and standard deviation $\sigma_i(e_i)$, where e_i is the effort exerted by the worker. As in OSE, $\sigma_i(e_i)$ is a convex, monotonically decreasing function of e_i: as a worker increases her effort level, the result should be closer to the true value of the metric. Moreover, $\{\sigma_i(\cdot)\}_{i\in\mathcal{W}}$ is known to the mechanism designer.

Each worker i has a utility function dependent on e_i, which she attempts to maximize:

$$u_i = p_i - e_i \tag{5.2}$$

where $p_i(\cdot)$ denotes the payment commitment for the i-th worker. This payment could depend on all workers' instances and labels. As in OSE, workers are rational, so they will participate only if $u_i \geq 0$. From now on, for simplicity we will use x_i to denote the difference $\vec{x}_{i,1} - \vec{x}_{i,2}$ between a pair of instances $\vec{x}_{i,1}, \vec{x}_{i,2}$ that is assigned to worker i, so: $d_M(\vec{x}_{i,1}, \vec{x}_{i,2}) = \sqrt{(\vec{x}_{i,1} - \vec{x}_{i,2})^T M (\vec{x}_{i,1} - \vec{x}_{i,2})} = \sqrt{x_i^T M x_i}$.

The payment commitments need to incentivize workers to produce distance estimates that allow matrix M to be sufficiently accurate, while also keeping the total cost reasonable. The following definition formally formulates our problem.

Definition 1 (Metric Learning with Strategic Workers (MLSW))**.** *Suppose that we are given:*

- *access to a set $\mathcal{W}$ of strategic workers for some* unknown *metric, where each worker $i \in \mathcal{W}$ has a known function σ_i mapping effort to accuracy; we also assume that all workers' estimations are independent;*
- *a pool of data points $T \subseteq \mathbb{R}^d$;*
- *a distribution F over $\mathbb{R}^d$ (the distribution of the difference between the test pair $\bar{x} = \bar{x}_1 - \bar{x}_2$).*

Our goal is to:

1. *choose how to assign data points to workers; if we assign worker i the difference between two data points $x_{i,1}$ and $x_{i,2}$ from T, worker i should return an estimated distance between $x_{i,1}$ and $x_{i,2}$; if we assign worker i a null input $\perp$, it means that we exclude worker i from the task* [1] *and s/he does not need to return anything;*

2. *commit to a payment function p_i to each $i \in \mathcal{W}$, where p_i is a (potentially) randomized mapping $p_i : (x_i, y_i)_{i\in\mathcal{W}} \mapsto \mathbb{R}$, which may depend not only on the estimate produced by worker i but also the estimates produced by the other workers.*

Given labelled data points $(x_i, y_i)_{i\in\mathcal{W}}$, we apply an ERM algorithm to compute a metric. In particular, let $\hat{M}$ be the solution [2] *of the following mathematical program.*

$$\min_{M\succeq 0} h(M) = \min_{M\succeq 0} \sum_{i\in\mathcal{W}} (y_i - x_i^T M x_i)^2 \tag{5.3}$$

Subject to our decisions in 1 and 2, we are looking to minimize a weighted average of the mean-square error of our estimation and the expected payments made to the workers, namely:

$$\mathbb{E}_{\bar{x},\boldsymbol{y}(\boldsymbol{e})} \left[\left(\bar{x}^T M^* \bar{x} - \bar{x}^T \hat{M} \bar{x} \right)^2 + \alpha \sum_{i\in\mathcal{W}} p_i \left((x_i, y_i(e_i))_{i\in\mathcal{W}} \right) \right] \tag{5.4}$$

for some $\alpha > 0$, where the expectation is taken with respect to all the randomness in the setting: the randomness in $\bar{x} \sim F$, the randomness in the outputs $\boldsymbol{y}(\boldsymbol{e}) \triangleq \{y_i(e_i)\}_{i\in\mathcal{W}}$ produced by the workers, and the randomness in the payment functions. For (5.4) *to be a well-defined objective, we need to be able to predict the efforts $\{e_i\}_{i\in\mathcal{W}}$ that the workers will exert given our decisions for 1*

[1] More specifically, whenever we write $i \in \mathcal{W}$, we only consider the workers whose inputs are not $\perp$.

[2] We first assume that we can solve this problem exactly. We will later discuss the case where only an approximate solution can be obtained.

and 2. We discuss how this can be achieved in Section 5.2.1 below. At the very least, our prediction needs to satisfy the individual rationality constraint of Definition 3, i.e. that the expected payment to each worker i is at least as large as e_i, otherwise the worker would not participate.

5.2 Our Mechanism

In this section, we show how to design a mechanism for the MLSW problem. We demonstrate in Section 5.2.2 that for any effort levels $\{e_i\}_{i\in\mathcal{W}}$, we can design an individually rational mechanism such that there is a unique dominant strategy equilibrium where every worker i exerts effort e_i. In Section 5.2.2, we provide guidance on how to assign data points to workers with access to any exact or approximate algorithm for solving the mathematical program (5.3). First, let us formally define these game theoretic solution concepts.

5.2.1 Incentives

Definition 2 (Unique Dominant Strategy Equilibrium)**.** *A solution to MLSW — comprising queries $\{x_i\}_{i\in\mathcal{W}}$, and payment commitments $\{p_i\}_{i\in\mathcal{W}}$—induces a* unique dominant strategy equilibrium (UDSE) $\{e_i^*\}_{i\in\mathcal{W}}$ *iff, for all $i \in \mathcal{W}$ and all $\{e_j\}_{j\in\mathcal{W}}$:*

$$\mathbb{E}\left[p_i\left((x_i, y_i(e_i^*)), (x_j, y_j(e_j))_{j\in\mathcal{W}\setminus\{i\}}\right)\right] - e_i^* \geq \mathbb{E}\left[p_i\left((x_j, y_j(e_j))_{j\in\mathcal{W}}\right)\right] - e_i,$$

where the expectation is with respect to everything that is random, with equality only if $e_i = e_i^*$. In words, no matter what effort levels the other workers choose, the unique optimal effort level of every worker i is e_i^*.

If a game has a UDSE, then it is fairly straightforward for the players to decide their strategies. All mechanisms considered in this paper have a UDSE. Notice that it is rather rare in game theory for a game to have such an outcome, which poses a significant constraint on our design. Furthermore, since the workers are assumed strategic and their participation is voluntary, they should not be making a loss when participating. This is captured by the following definition, adding an additional requirement to our solutions to MLSW.

Definition 3 (Individual Rationality)**.** *Given a solution to MLSW—comprising queries* $\{x_i\}_{i\in\mathcal{W}}$*, and payment commitments* $\{p_i\}_{i\in\mathcal{W}}$*, a collection of efforts* $\{e_i^*\}_{i\in\mathcal{W}}$ *satisfies* individual rationality (IR) *iff, for all workers* $i \in \mathcal{W}$*,*

$$\mathbb{E}\left[p_i\left(\left(x_j, y_j(e_j^*)\right)_{j\in\mathcal{W}}\right)\right] - e_i^* \geq 0.$$

5.2.2 Design of Our Mechanism

In this section, we show how to design our mechanism. We first argue that for any collection of efforts $\{\hat{e}_i\}_{i\in\mathcal{W}}$, there is a payment commitment such that every worker exerting effort level $\hat{e}_i$ is an individually rational and unique dominant strategy equilibrium. Then we provide some guidance on how to assign the tasks to workers so that our mechanism has high overall performance.

Our Payment Commitments

In this section, we show how to design the payment commitments in our mechanism. Indeed, our approach is applicable to a much broader setting. We prove that for any learning task as long as there exists a *well-behaved* estimator, for any effort level $\{\hat{e}_i\}_{i\in\mathcal{W}}$, one can set up a payment commitment scheme so that (i) every worker exerting effort level $\hat{e}_i$ is an individually rational and unique dominant strategy equilibrium, and (ii) extracts optimal worker surplus – the expected utility of every worker is 0. First, we formally define what is a well-behaved estimator.

Definition 4 (Well-behaved Estimator)**.** *Let* $\mathcal{H}$ *be a family of functions* $f : \mathcal{D} \mapsto \mathbb{R}$*, where* $\mathcal{D} \subseteq \mathbb{R}^n$*. An estimator for* $\mathcal{H}$ *takes as input a collection* $(x_i, y_i)_{i=1}^k$ *of examples* $(x_i, y_i) \in \mathcal{D} \times \mathbb{R}$ *and produces an* estimated function $\hat{f}_{(x_i,y_i)_{i=1}^k}$ *mapping from* $\mathcal{D}$ *to* $\mathbb{R}$*. An estimator* $\hat{f}$ *for* $\mathcal{H}$ *is* well-behaved *iff there exists some function* g [3] *such that, for all distributions* F *over* $\mathcal{D}$*, functions* $f \in \mathcal{H}$*, and vectors* $\boldsymbol{x} \in \mathcal{D}^*$

[3] Intuitively, an estimator $\hat{f}$ is *well-behaved* iff its expected mean square error does not depend on f.

and $\boldsymbol{\sigma} \in \mathbb{R}_+^*$ *(of the same dimension as* $\boldsymbol{x}$*):*[4]

$$\mathbb{E}_{\boldsymbol{y},x^*}\left[\left(\hat{f}_{(\boldsymbol{x},\boldsymbol{y})}(x^*) - f(x^*)\right)^2\right] = g(\boldsymbol{x}, F, \boldsymbol{\sigma}),$$

where for the purposes of the expectation on the left hand side $x^* \sim F$ *and, independently for all* i, $y_i = f(x_i) + \epsilon_i$*, where each* ϵ_i *is sampled from an arbitrary distribution of mean* 0 *and variance* σ_i^2*, (and when* $\boldsymbol{x}$ *is such that* $\hat{f}_{(\boldsymbol{x},\boldsymbol{y})}$ *is well-defined).*

Note that several common estimators, such as linear regression, polynomial regression, finite-dimensional kernel estimation, are well-behaved. Our definition is a relaxation of the definition used by [8], as we do not restrict the estimator to only produce functions from $\mathcal{H}$. This relaxation is crucial for us to apply our result to metric learning.

Theorem 1. *Suppose the underlying truth is generated by some function* f *from some hypothesis class* $\mathcal{H}$*. Given any query* x*, a worker* i *with effort* e_i *can produce a label* $y = f(x) + \epsilon_i$*, where* ϵ_i *is a random variable with zero mean and variance* $\sigma_i(e_i)^2$*. If* $\hat{f}$ *is a well-behaved estimator for* $\mathcal{H}$*, for any queries* $\{x_i\}_{i\in\mathcal{W}}$ *and any collection of efforts* $\{\hat{e}_i\}_{i\in\mathcal{W}}$*, there exists a collection of real numbers* $(a_i, b_i)_{i\in[n]}$ *such that if the payment commitment is* $a_i - b_i\left(y_i(e_i) - \hat{f}_{(\boldsymbol{x},\boldsymbol{y}(\boldsymbol{e}))_{-i}}(x_i)\right)^2$ *for worker* i*,*

1. *every worker exerting effort level* $\hat{e}_i$ *is an individually rational and unique dominant strategy equilibrium, and*

2. *the expected utility for each worker is* 0 *at this equilibrium.*

In particular, we can choose b_i *to be* $\frac{-1}{2\sigma_i(\hat{e}_i)\sigma_i'(\hat{e}_i)}$ *and* a_i *to be* $b_i \cdot (\sigma_i(\hat{e}_i)^2 + g(\boldsymbol{x}_{-i}, \mathbf{1}_{x_i}, \boldsymbol{\sigma}_{-i}(\hat{\boldsymbol{e}}_{-i}))) + \hat{e}_i$*, where the function* g *is defined in Definition 4.*

Proof. First, we prove that this is a UDSE. From worker i's perspective, she wants to find an effort level e_i that maximizes her expected payment minus her effort:

$$\max_{e_i} \mathbb{E}_{y_i(e_i)}\left[a_i - b_i(y_i(e_i) - \hat{f}_{(\boldsymbol{x},\boldsymbol{y}(\boldsymbol{e}))_{-i}}(x_i))^2\right] - e_i.$$

[4]We use the shorthand $\mathcal{D}^* \triangleq \bigcup_{i=1}^{\infty} \mathcal{D}^i$, and similarly for $\mathbb{R}_+^*$.

Since a_i is independent of our choice e_i, so we can ignore this term. On the other hand

$$\begin{aligned}
&\mathbb{E}_{y_i(e_i)}\left[b_i\left(y_i(e_i)-\hat{f}_{(\boldsymbol{x},\boldsymbol{y}(\boldsymbol{e}))_{-i}}(x_i)\right)^2\right]\\
=&b_i\cdot\mathbb{E}_{y_i(e_i)}\left[((y_i(e_i)-y_i^*)+(y_i^*-\hat{f}_{(\boldsymbol{x},\boldsymbol{y}(\boldsymbol{e}))_{-i}}(x_i)))^2\right]\\
=&b_i\cdot\sigma_i(e_i)^2+b_i\cdot g(\boldsymbol{x}_{-i},\mathbf{1}_{x_i},\boldsymbol{\sigma}_{-i}(\hat{\boldsymbol{e}}_{-i}))
\end{aligned}$$

Notice that the second term does not depend on e_i, so we can ignore this term when finding the utility maximizing effort level for i. After the simplification, to maximize the expected utility, worker i needs to choose an effort level that maximizes $-b_i\cdot\sigma_i(e_i)^2-e_i$. As $\sigma_i(\cdot)$ is a convex decreasing function, it is not hard to see that $-b_i\cdot\sigma_i(e_i)^2-e_i$ is a concave function and thus has a unique optimal solution. Suppose e_i^* is the optimum, then taking derivative over e_i and setting it to 0 gives the following condition for e_i^*: $2\sigma_i'(e_i^*)\cdot\sigma_i(e_i^*)=-1/b_i$. Due to the aforementioned properties of $\sigma_i(\cdot)$, $e_i^*=\hat{e}_i$ is the unique solution of this equation.

Next, we argue that $\{\hat{e}_i\}_{i\in\mathcal{W}}$ is individually rational under our payment commitments. By our choice of a_i, it is not hard to see that the expected utility for bidder i is 0. Thus, $\{\hat{e}_i\}_{i\in\mathcal{W}}$ is individually rational and extracts full surplus from all workers. □

This theorem was implicitly stated and proved in Theorem 9 of [8]. Their result only showed that for the optimal effort levels $\{e_i^*\}_{i\in\mathcal{W}}$, such a payment commitment scheme induces $\{e_i^*\}_{i\in\mathcal{W}}$ as a UDSE. We extend their result to arbitrary effort levels and also allow the estimator $\hat{f}$ to produce functions outside $\mathcal{H}$. Next, we apply Theorem 1 to metric learning.

Corollary 1.1. *Let* $\mathcal{H}=\left\{f(x)=x^TMx\;\middle|\;M \text{ is a PSD matrix in } \mathbb{R}^{d\times d}\right\}$. *The* quadratic regression *estimator* $\hat{f}$ *is a well-behaved estimator for* $\mathcal{H}$*, and for every collection of efforts* $\{\hat{e}_i\}_{i\in\mathcal{W}}$*, we can use* $\hat{f}$ *to design payment commitments as described in Theorem 1 so that (i) every worker exerting effort level* $\hat{e}_i$ *is an individually rational and unique dominant strategy equilibrium, and (ii) the expected utility for each worker is* 0 *at this equilibrium.*

Proof. As any function in $\mathcal{H}$ can be represented as a quadratic function over x, the quadratic regression estimator $\hat{f}$ is clearly a well-behaved estimator for $\mathcal{H}$. Then the claim follows from Theorem 1. □

One major challenge for solving the MLSW problem is that we cannot directly control how much effort each worker exerts, but can only affect their effort levels through payments. In general, it is difficult to characterize how the workers react to different payment commitments. Surprisingly, Corollary 1.1 shows that through a particular format of payment commitments, we can indeed incentivize the workers to exert *any levels of effort* we want and we extract *full surplus* from the workers meaning the payment for any worker equals the amount of effort exerted by that worker. Equipped with Corollary 1.1, we can reduce the MLSW problem into a pure optimization problem, that is, how do we assign data points to workers and how do we choose the effort levels for the workers so that the following expression is minimized:

$$\mathbb{E}_{\bar{x}\sim F,\boldsymbol{y}(\boldsymbol{e})}\left[(\bar{x}^T M^* \bar{x} - \bar{x}^T \hat{M} \bar{x})^2 + \alpha \sum_{i\in\mathcal{W}} e_i\right] \tag{5.5}$$

Note that in expression (5.5) we have replaced the sum of payments in expression (5.4) with the sum of the workers' effort levels. It is not hard to see that expression (5.5) is a lower bound of expression (5.4) for any IR mechanism. Also, for any assignment of the data points and any collection of effort levels, if we adopt the payment scheme provided by Corollary 1.1, expression (5.4) has the same value as expression (5.5). Therefore, minimizing expression (5.5) is equivalent to minimizing expression (5.4). In the next section, we provide some guidance on how to minimize expression (5.5).

Choosing the Task Assignment and Effort Levels

In this section, we discuss how to choose the task assignment and effort levels to minimize expression (5.5). An astute reader might have already realized that we may not even have all the necessary information to optimize expression (5.5). If the metric induced by $\hat{M}$ is well-behaved, $\mathbb{E}_{\bar{x},\boldsymbol{y}(\boldsymbol{e}^*)}\left[(\bar{x}^T M^* \bar{x} - \bar{x}^T \hat{M} \bar{x})^2\right]$ only

depends on the task assignment and the effort levels. However, to the best of our knowledge, we are not aware of any algorithm that produces a $\hat{M}$ whose induced metric is well-behaved [5]. As a consequence, $\mathbb{E}_{\bar{x},\boldsymbol{y}(\boldsymbol{e}^*)}\left[(\bar{x}^T M^* \bar{x} - \bar{x}^T \hat{M} \bar{x})^2\right]$ may depend on the true underlying metric, which we do not have any information about. To proceed, we take a different approach. Instead of directly optimizing expression (5.5), we provide a relaxation of expression (5.5) that is independent of the underlying metric. Then we choose the task assignment and effort levels to optimize this relaxation.

First, let us fix some notations. We use $\mathcal{X}_i \in \mathbb{R}^{d\times d}$ to denote $x_i \cdot x_i^T$, $V_C \in \mathbb{R}^{d^2}$ to denote the vectorization[6] of a $d \times d$ matrix C, and $\mathcal{V}$ to denote $\sum_{i=1}^{k} V_{\mathcal{X}_i} \cdot V_{\mathcal{X}_i}^T$. The following Lemma provides an upper bound for $||\hat{M} - M^*||_2$.

Lemma 1. *If $\mathcal{V}$ has rank $\frac{d(d+1)}{2}$,*

$$||\hat{M} - M^*||_2^2 \leq 4||Q^+||_2^2 \cdot \left|\left|\left(\sum_{i\in\mathcal{W}} \delta_i \cdot V_{\mathcal{X}_i}^T\right) Q^+\right|\right|_2^2,$$

where Q is a $d^2 \times d^2$ PSD matrix such that $Q^T Q = \mathcal{V}$ with Q^+ being the pseudoinverse of Q, and $\delta_i = y_i - y_i^*$. Also, for any $\{e_i\}_{i\in\mathcal{W}}$, $\mathbb{E}_{\boldsymbol{y}(\mathbf{e})}\left[||M^* - \hat{M}||_2^2\right] \leq 4||Q^+||_2^2 \cdot \sum_{i\in\mathcal{W}} \sigma_i(e_i)^2 \cdot ||V_{\mathcal{X}_i}^T \cdot Q^+||_2^2$.

Proof. As defined in Definition 1, $\hat{M}$ is the optimal solution of the following optimization problem $\min_{M\succcurlyeq 0} h(M)$ where $h(M) = \sum_{i\in\mathcal{W}} \left(y_i - V_M^T \cdot V_{\mathcal{X}_i}\right)^2$. Our goal is to bound $||\hat{M} - M^*||_2^2$, which is the same as bounding $||V_{\hat{M}} - V_{M^*}||_2$. Next, we will use the condition that $h(\hat{M}) \leq h(M^*)$ to derive an upper bound of

[5] In Corollary 1.1, we used the quadratic regression as a well-behaved estimator, but we cannot use the quadratic regression here, as it is not guaranteed to produce a metric.

[6] By appending the column vectors of C into a column vector with d^2 entries.

$||V_{\hat{M}} - V_{M^*}||_2^2$.

$$\begin{aligned}
0 \geq & h(\hat{M}) - h(M^*) \\
= & \sum_{i \in \mathcal{W}} \left(y_i - {V_{\hat{M}}}^T \cdot V_{\mathcal{X}_i}\right)^2 - \sum_{i \in \mathcal{W}} \left(y_i - {V_{M^*}}^T \cdot V_{\mathcal{X}_i}\right)^2 \\
= & 2 \left(\sum_{i \in \mathcal{W}} y_i \cdot V_{\mathcal{X}_i}^T \right) \cdot (V_{M^*} - V_{\hat{M}}) + V_{\hat{M}}^T \cdot \mathcal{V} \cdot V_{\hat{M}} - V_{M^*}^T \cdot \mathcal{V} \cdot V_{M^*} \\
= & 2 \left(\sum_{i \in \mathcal{W}} y_i \cdot V_{\mathcal{X}_i}^T - V_{M^*}^T \cdot \mathcal{V} \right) \cdot (V_{M^*} - V_{\hat{M}}) + (V_{M^*} - V_{\hat{M}})^T \cdot \mathcal{V} \cdot (V_{M^*} - V_{\hat{M}})
\end{aligned} \tag{5.6}$$

Notice that:

$$V_{M^*}^T \cdot \mathcal{V} = \sum_{i \in \mathcal{W}} V_{M^*}^T \cdot V_{\mathcal{X}_i} \cdot V_{\mathcal{X}_i}^T = \sum_{i \in \mathcal{W}} \left(x_i^T \cdot M^* \cdot x_i\right) \cdot V_{\mathcal{X}_i}^T = \sum_{i \in \mathcal{W}} y_i^* \cdot V_{\mathcal{X}_i}^T$$

Hence the above inequality can be simplified to

$$2 \left(\sum_{i \in \mathcal{W}} -\delta_i \cdot V_{\mathcal{X}_i}^T \right) \cdot (V_{M^*} - V_{\hat{M}}) \geq (V_{M^*} - V_{\hat{M}})^T \cdot \mathcal{V} \cdot (V_{M^*} - V_{\hat{M}}) \tag{5.7}$$

Observe that $\mathcal{V}$ is a PSD matrix, so there exists another PSD matrix Q such that $Q^T Q = \mathcal{V}$, and the RHS of the inequality can be rewritten as $||Q \cdot (V_{M^*} - V_{\hat{M}})||_2^2$. If $\mathcal{V}$ is full rank, then so is Q, and we can rewrite the LHS as $2 \left(\sum_{i \in \mathcal{W}} -\delta_i \cdot V_{\mathcal{X}_i}^T\right) Q^{-1} \cdot Q \cdot (V_{M^*} - V_{\hat{M}})$. Now both sides have $Q \cdot (V_{M^*} - V_{\hat{M}})$ and we can cancel it out. However, it turns out $\mathcal{V}$ is never full rank, as all $V_{\mathcal{X}_j}$ are vectorization of symmetric matrices. Indeed, it is not hard to argue that $\mathrm{rank}(\mathcal{V})$ is at most $d(d+1)/2$, as the space of $d \times d$ symmetric matrices has dimension $d(d+1)/2$. We will proceed with the pseudoinverse Q^+ of Q, but we first argue that $Q^+ \cdot Q \cdot V_M = V_M$ for any symmetric matrix M. Before we proceed, let's prove the following lemma:

Lemma 2. *Let Q be a PSD matrix such that $Q^T Q = \mathcal{V}$ and Q^+ be the pseudoinverse of Q, then as long as $rank(\mathcal{V}) = \frac{d(d+1)}{2}$, $Q^+ \cdot Q \cdot V_M = V_M$ for any symmetric matrix M.*

Proof. Let UAU^T be the eigendecomposition of $\mathcal{V}$, where $U = [u_1, \ldots, u_{d^2}]$ and A is a diagonal matrix. Moreover, u_j is the j-th eigenvector of $\mathcal{V}$ and $A_{jj} = \lambda_j$

is the corresponding eigenvalue. Since $rank(\mathcal{V}) = \frac{d(d+1)}{2}$, WLOG we assume $\lambda_j > 0$ if and only if $j \leq \frac{d(d+1)}{2}$. Since $\lambda_j u_j = \mathcal{V} \cdot u_j = \sum_{k \in \mathcal{W}} \left(V_{\mathcal{X}_k}^T \cdot u_j\right) \cdot V_{\mathcal{X}_k}$, u_j is also the vectorization of some symmetric matrix for all $j \leq \frac{d(d+1)}{2}$. As the space of $d \times d$ symmetric matrices has dimension $d(d+1)/2$, the top $d(d+1)/2$ eigenvectors of $\mathcal{V}_i$ form a basis for the space of the vectorization of all $d \times d$ symmetric matrices. In other words, for any symmetric matrix M, V_M is in the linear span of $\{u_1, \ldots, u_{d(d+1)/2}\}$. Notice that $Q = UA^{1/2}U^T$ and $Q^+ = UB^{1/2}U^T$ where B is a diagonal matrix with $[1/\lambda_1, \ldots, 1/\lambda_{d(d+1)/2}, 0, \ldots, 0]$ on the diagonal. Since V_M lies in the span of the top $d(d+1)/2$ eigenvectors so $Q^+Q \cdot V_M = V_M$. □

Now, we can rewrite the LHS of Inequality (5.7) as

$$2\left(\sum_{i \in \mathcal{W}} -\delta_i \cdot V_{\mathcal{X}_i}^T\right) Q^+ \cdot Q \cdot (V_{M^*} - V_{\hat{M}})$$

and RHS of Inequality (5.7) as

$$||Q \cdot (V_{M^*} - V_{\hat{M}})||_2^2$$

Next, we apply the Cauchy-Schwarz inequality to relax the LHS to obtain the following inequality:

$$2\left|\left|\left(\sum_{i \in \mathcal{W}} \delta_i \cdot V_{\mathcal{X}_i}^T\right) Q^+\right|\right|_2 \cdot ||Q \cdot (V_{M^*} - V_{\hat{M}})||_2 \geq ||Q \cdot (V_{M^*} - V_{\hat{M}})||_2^2$$

Dividing $||Q \cdot (V_{M^*} - V_{\hat{M}})||_2$ on both sides gives us

$$2\left|\left|\left(\sum_{i \in \mathcal{W}} \delta_i \cdot V_{\mathcal{X}_i}^T\right) Q^+\right|\right|_2 \geq ||Q \cdot (V_{M^*} - V_{\hat{M}})||_2$$

Finally, we multiply $||Q_i^+||_2$ on both sides and apply the Cauchy-Schwarz inequality to relax the RHS.

$$\begin{aligned} 2||Q^+||_2 \cdot \left|\left|\left(\sum_{i \in \mathcal{W}} \delta_i \cdot V_{\mathcal{X}_i}^T\right) Q^+\right|\right|_2 &\geq ||Q^+||_2 \cdot ||Q \cdot (V_{M^*} - V_{\hat{M}})||_2 \\ &\geq ||Q^+Q \cdot (V_{M^*} - V_{\hat{M}})||_2 \\ &= ||V_{M^*} - V_{\hat{M}}||_2 \end{aligned} \tag{5.8}$$

Hence, $||V_{M^*} - V_{\hat{M}}||_2^2 \leq 4||Q^+||_2^2 \cdot \left|\left|\left(\sum_{i\in\mathcal{W}} \delta_i \cdot V_{\mathcal{X}_i}^T\right) Q^+\right|\right|_2^2$.

If we let $y_i = y_i(e_i)$, we have:

$$\begin{aligned}\mathbb{E}_{\boldsymbol{y}(\mathbf{e})}\left[||V_{M^*} - V_{\hat{M}}||_2^2\right] \leq & \mathbb{E}_{\boldsymbol{y}(\mathbf{e})}\left[4||Q^+||_2^2 \cdot \left|\left|\left(\sum_{i\in\mathcal{W}} \delta_i \cdot V_{\mathcal{X}_i}^T\right) Q^+\right|\right|_2^2\right] \\ = & 4||Q^+||_2^2 \cdot \sum_{i\in\mathcal{W}} \sigma_i(e_i)^2 \cdot ||V_{\mathcal{X}_i}^T \cdot Q^+||_2^2\end{aligned} \tag{5.9}$$

□

We have obtained an upper bound of $||\hat{M} - M^*||_2$ when $\hat{M}$ is the exact solution of (5.3). What if we only have access to an approximate algorithm? Suppose our algorithm outputs a solution $\tilde{M}$ such that $||\nabla h(\tilde{M})||_2 \leq \gamma$ [7], where γ is a parameter of the algorithm. In the next Lemma, we show how to obtain an upper bound for $||\tilde{M} - M^*||_2$.

Lemma 3. *Suppose $\hat{M}$ is a minimizer for function $h(\cdot)$ and $\tilde{M}$ is a $d \times d$ matrix such that $||\nabla h(\tilde{M})||_2 \leq \gamma$. If $\mathcal{V}$ has rank $\frac{d(d+1)}{2}$,*

$$||\tilde{M} - M^*||_2^2 \leq 8||Q^+||_2^2 \cdot \left|\left|\left(\sum_{i\in\mathcal{W}} \delta_i \cdot V_{\mathcal{X}_i}^T\right) Q^+\right|\right|_2^2 + 8\gamma^2||Q^+||_2^4$$

where δ_i, Q and Q^+ are defined in Lemma 1. Also, for any $\{e_i\}_{i\in\mathcal{W}}$,

$$\mathbb{E}_{\boldsymbol{y}(\mathbf{e})}\left[||M^* - \tilde{M}||_2^2\right] \leq 8||Q^+||_2^2 \cdot \sum_{i\in\mathcal{W}} \sigma_i(e_i)^2 \left|\left|V_{\mathcal{X}_i}^T \cdot Q^+\right|\right|_2^2 + 2\gamma^2||Q^+||_2^4$$

.

Proof. First observe that

$$||\tilde{M} - M^*||_2^2 \leq 2||\hat{M} - M^*||_2^2 + 2||\tilde{M} - \hat{M}||_2^2$$

. Hence, we only need to upper bound $||\tilde{M} - \hat{M}||_2^2$. Similar to Inequality (5.6), we can show that

$$2\left(\sum_{i\in\mathcal{W}} y_i \cdot V_{\mathcal{X}_i}^T - V_{\tilde{M}}^T \cdot \mathcal{V}\right) \cdot (V_{\hat{M}} - V_{\tilde{M}}) \geq (V_{\hat{M}} - V_{\tilde{M}})^T \cdot \mathcal{V} \cdot (V_{\hat{M}} - V_{\tilde{M}})$$

[7] Note that $h(M) = \sum_{i\in\mathcal{W}}(y_i - x_i^T M x_i)^2$.

Notice that $-2\left(\sum_{i\in\mathcal{W}} y_i \cdot V_{\mathcal{X}_i}^T - V_{\tilde{M}}^T \cdot \mathcal{V}\right) = \nabla h(\tilde{M})$. By Lemma 2, we can rewrite the inequality above as

$$-\nabla h(\tilde{M}) Q^+ Q \cdot (V_{\hat{M}} - V_{\tilde{M}}) \geq ||Q \cdot (V_{\hat{M}} - V_{\tilde{M}})||_2^2$$

After relaxing the LHS with the Cauchy-Schwarz inequality, we have

$$||\nabla h(\tilde{M})||_2 ||Q^+||_2 ||Q \cdot (V_{\hat{M}} - V_{\tilde{M}})||_2 \geq ||Q \cdot (V_{\hat{M}} - V_{\tilde{M}})||_2^2$$

Multiply $\frac{||Q^+||_2}{||Q\cdot(V_{\hat{M}} - V_{\tilde{M}})||_2}$ on both sides and apply Cauchy-Schwarz Inequality to relax the RHS, we have

$$||\nabla f_i(V_{\tilde{M}})||_2 ||Q_i^+||_2^2 \geq |V_{\hat{M}} - V_{\tilde{M}}||_2$$

Hence,

$$|V_{\hat{M}} - V_{\tilde{M}}||_2^2 \leq \gamma^2 ||Q_i^+||_2^4$$

Putting everything together, we can prove our claims:

$$||\tilde{M} - M^*||_2^2 \leq 8||Q^+||_2^2 \cdot \left|\left|\left(\sum_{i\in\mathcal{W}} \delta_i \cdot V_{\mathcal{X}_i}^T\right) Q^+\right|\right|_2^2 + 8\gamma^2 ||Q^+||_2^4$$

□

Theorem 2. *For any MLSW problem, any task assignment $\{x_i\}_{i\in\mathcal{W}}$, and any effort levels $\{e_i\}_{i\in\mathcal{W}}$, there exists a payment commitment such that:*

1. *every worker exerting effort level e_i is an individually rational and unique dominant strategy equilibrium, and*
2. *the expected utility for each worker is 0 at this equilibrium.*

Moreover, if $\mathcal{V} = \sum_{i\in\mathcal{W}} V_{\mathcal{X}_i} V_{\mathcal{X}_i}^T$ has rank $\frac{(d+1)d}{2}$, expression (5.4) *is upper bounded by*

$$4\mathbb{E}_{\bar{x}}[||\bar{x}||_2^4] \cdot ||Q^+||_2^2 \cdot \sum_{i\in\mathcal{W}} \sigma_i(e_i)^2 \cdot ||V_{\mathcal{X}_i}^T \cdot Q^+||_2^2 + \alpha \sum_{i\in\mathcal{W}} e_i$$

when $\hat{M}$ is an exact solution of the mathematical program (5.3)*, and is upper bounded by*

$$\mathbb{E}_{\bar{x}}[||\bar{x}||_2^4] \cdot \left(8||Q^+||_2^2 \cdot \sum_{i \in \mathcal{W}} \sigma_i(e_i)^2 \cdot ||V_{\mathcal{X}_i}^T \cdot Q^+||_2^2 + 2\gamma^2 ||Q^+||_2^4 \right) + \alpha \sum_{i \in \mathcal{W}} e_i \tag{5.10}$$

when $\hat{M}$ *is an approximate solution of the mathematical program* (5.3) *and* $||\nabla h(\hat{M})||_2 \leq \gamma$.

Proof. The first two properties follow from Corollar 1.1. Since every worker has utility 0, expression (5.4) equals to expression (5.5). Notice that

$$\begin{aligned} \mathbb{E}_{\bar{x} \sim F, \boldsymbol{y}(\boldsymbol{e})} \left[\left(\bar{x}^T M^* \bar{x} - \bar{x}^T \hat{M} \bar{x} \right)^2 \right] &\leq \mathbb{E}_{\bar{x} \sim F, \boldsymbol{y}(\boldsymbol{e})} \left[||\bar{x}||_2^4 \cdot ||M^* - \hat{M}||_2^2 \right] \\ &= \mathbb{E}_{\bar{x} \sim F} \left[||\bar{x}||_2^4 \right] \cdot \mathbb{E}_{\boldsymbol{y}(\boldsymbol{e})} \left[||M^* - \hat{M}||_2^2 \right]. \end{aligned} \tag{5.11}$$

Our claim follows from the upper bounds for $E_{\boldsymbol{y}(\boldsymbol{e})} \left[||M^* - \hat{M}||_2^2 \right]$ in Lemma 1 and 3. □

Theorem 2 provides an upper bound of the weighted average of the mean-square error of our estimation and the expected payments made to the workers. Notice that this upper bound is independent of the underlying true metric and only depends on the task assignment and the effort levels of the workers. Hence, we can choose the task assignment and effort levels that minimize this upper bound, and design our payment commitments accordingly. For Theorem 2 to hold, $\mathrm{rank}(\mathcal{V})$ needs to be $\frac{d(d+1)}{2}$. The following Theorem shows that a natural requirement on the task assignment is sufficient to guarantee this condition.

Theorem 3. *If there exists* $d+1$ *data points in* T *that do not lie in any subspace with dimension lower than* d*, and all* $d(d+1)/2$ *pairwise distance between these* $d+1$ *points are assigned to some worker in* $\mathcal{W}$*, then* $\mathrm{rank}(\mathcal{V}) = \frac{d(d+1)}{2}$.

we argue that $\mathrm{rank}(\mathcal{V}) = \frac{d(d+1)}{2}$ as long as the task assignment satisfies the condition in Theorem 3. First, we provide an alternative way to view this condition.

Lemma 4. *$rank(\mathcal{V}) = \frac{d(d+1)}{2}$ if and only if there are $\frac{d(d+1)}{2}$ linear independent vectors in $\{V_{\mathcal{X}_i}\}_{i\in\mathcal{W}}$.*

Proof. Let $\ell = d(d+1)/2$. We first prove that if there are ℓ linear independent vectors in $\{V_{\mathcal{X}_i}\}_{i\in\mathcal{W}}$, then $\mathcal{V}$ has rank ℓ. Let $S = \{V_{\mathcal{X}_{j_1}}, \ldots, V_{\mathcal{X}_{j_\ell}}\}$ contain these ℓ linear independent vectors. We call a vector V_M symmetric if M is a symmetric matrix and V_M is the vectorization of M. As the space of symmetric vectors has dimension ℓ, any symmetric V_M can be written as a linear combination of vectors in S: $V_M = \sum_{k=1}^{\ell} \alpha_k \cdot V_{\mathcal{X}_{j_k}}$. If $0 = V_M^T \cdot \mathcal{V} \cdot V_M = \sum_{j\in\mathcal{W}} V_M^T \cdot V_{\mathcal{X}_j} \cdot V_{\mathcal{X}_j} \cdot V_M$, then $V_M^T V_{\mathcal{X}_{j_k}} = 0$ for all $k \in [\ell]$, which implies that $||V_M||_2^2 = 0$. Therefore, all symmetric vectors are orthogonal to the null space of $\mathcal{V}$, the null space has $rank$ at most $d^2 - \ell = d(d-1)/2$. On the other hand, it is easy to argue that the null space has rank at least $d(d-1)/2$ as the vectorization of any $d \times d$ skew matrix is in the null space of $\mathcal{V}$. To sum up, $rank(\mathcal{V}) = \ell$.

Next, we show that if $rank(\mathcal{V}) = \ell$, then there are ℓ linear independent vectors in $\{V_{\mathcal{X}_j}\}_{j\in\mathcal{W}}$. As $\mathcal{V}$ is a PSD matrix, it has ℓ linear independent eigenvectors $\{u_1, \ldots, u_\ell\}$ that have positive eigenvalues $\{\lambda_1, \ldots, \lambda_\ell\}$. We have $\lambda_k \cdot u_k = \mathcal{V} \cdot u_k = \sum_{j\in\mathcal{W}} \left(V_{\mathcal{X}_j}^T u_k\right) \cdot V_{\mathcal{X}_j}$. Hence, for all $k \in [\ell]$, u_k lies in the span of $\{V_{\mathcal{X}_j}\}_{j\in\mathcal{W}}$. Because $\{u_1, \ldots, u_\ell\}$ are linear independent, there must also exist ℓ linear independent vectors in $\{V_{\mathcal{X}_j}\}_{j\in\mathcal{W}}$. □

In the next Lemma, we show that under a natural condition $\{V_{\mathcal{X}_j}\}_{j\in\mathcal{W}}$ has rank $d(d+1)/2$. In particular, we show that if there exists $d+1$ data points in $\mathbb{R}^d$ that do not lie in any subspace with dimension lower than d, and all $(d+1)d/2$ pairwise distance between these $d+1$ points are assigned to some worker in $\mathcal{W}$, then $\{V_{\mathcal{X}_j}\}_{j\in\mathcal{W}}$ has rank $d(d+1)/2$.

Lemma 5. *Let $z_1, \ldots, z_{d+1} \in \mathbb{R}^d$ be $d+1$ different points that do not lie in any lower dimensional subspace. Let $\mathcal{Z}_{ij} = (z_i - z_j)\cdot(z_i - z_j)^T \in \mathbb{R}^{d\times d}$ and $V_{\mathcal{Z}_{ij}} \in \mathbb{R}^{d^2}$ be the vectorization of $\mathcal{Z}_{ij}$ for all $i < j$. Then these $d(d+1)/2$ different $V_{\mathcal{Z}_{ij}}$ are linear independent.*

Proof. WLOG, we assume z_{d+1} to be 0. As $z_1, \ldots, z_{d+1}$ do not lie in any lower dimensional subspace, $(z_1 - z_{d+1}), \ldots, (z_d - z_{d+1})$ are linear independent. In other

words, $z_1, \ldots, z_d$ are linear independent. Suppose there exists a linear combination of $V_{\mathcal{Z}_{ij}}$, $\sum_{i<j} \alpha_{ij} \cdot V_{\mathcal{Z}_{ij}} = 0$, then $\sum_{i<j} \alpha_{ij} \cdot \mathcal{Z}_{ij} = 0$. For notational simplicity, we let $\alpha_{ji} = \alpha_{ij}$ for all $i, j \in [d]$. Therefore for any $x \in \mathbb{R}^d$, $x^T \cdot \sum_{i<j} \alpha_{ij} \cdot \mathcal{Z}_{ij} = 0$. We first expand the LHS of this equation

$$x^T \cdot \sum_{i<j} \alpha_{ij} \cdot \mathcal{Z}_{ij} = \sum_{i \in [d]} x^T \cdot \left(\sum_{j \in [d+1]-\{i\}} \alpha_{ij} \cdot z_i - \sum_{j \in [d]-\{i\}} \alpha_{ij} \cdot z_j \right) \cdot z_i^T. \tag{5.12}$$

The RHS is a linear combination of $z_1, \ldots, z_d$, so it is 0 if and only if $x^T \cdot \left(\sum_{j \in [d+1]-\{i\}} \alpha_{ij} \cdot z_i - \sum_{j \in [d]-\{i\}} \alpha_{ij} \cdot z_j \right) = 0$ for all $i \in [d]$. As x can be a arbitrary vector, $\sum_{j \in [d+1]-\{i\}} \alpha_{ij} \cdot z_i - \sum_{j \in [d]-\{i\}} \alpha_{ij} \cdot z_j$ must be 0 for all $i \in [d]$. Notice that this is again a linear combination of $z_1, \ldots, z_d$, so it is 0 iff all the coefficients are all 0. Therefore, $\alpha_{ij} = 0$ for all $i \in [d]$ and $j \in [d]$. As the coefficient for z_i should be 0 as well, so $\alpha_{i(d+1)} = 0$ as well. Thus, $\sum_{i<j} \alpha_{ij} \cdot V_{\mathcal{Z}_{ij}} = 0$ iff $\alpha_{ij} = 0$ for all $i < j$, so $V_{\mathcal{Z}_{ij}}$ are linear independent. □

5.3 Metric Learning Algorithm

The optimization problem (5.3) could be tackled in various ways. Algorithm 3 is a heuristic algorithm that can be used to tackle it by projection-based gradient descent, similarly to [42]. In every iteration, they first run gradient descent for one step, followed by projecting the current solution to the cone of positive semi-definite matrices. We first run gradient descent until we reach some solution M whose norm of the gradient is smaller than a parameter γ, then project the solution to the positive semi-definite cone. If the projected solution is within β (another parameter) of M in $\mathcal{L}_2$ distance, we output the projected solution. Otherwise, we go back to the gradient descent phase and repeat. The output of this algorithm can be used with any method that requires a metric.

Algorithm 3 Modified Metric Learning

Input: $\{(x_i, y_i)\}_{i=1}^n$: vector of training data

$h(M) := \sum_{i=1}^n (y_i - x_i^T M x_i)^2$

M_0: initial $n \times n$ matrix

α: step size of gradient descent

γ: hyper parameter

β: hyper parameter

Output: a PSD matrix.

1. $M \leftarrow M_0$ (M_0 is a symmetric matrix)
2. **repeat**:
   3. **repeat**:
      Gradient Step $M \leftarrow M + \alpha \sum_{i=1}^n (y_i - x_i^T M x_i)(x_i x_i^T)$
   4. **until** $\|\nabla h(M)\|_2 < \gamma$
   5. **Projection Step** $\hat{M} \leftarrow$ projection of M to S_+^n (positive semi-definite cone)
6. **until** $\left\|\hat{M} - M\right\|_2 < \beta$
7. **output** $\hat{M}$

5.4 Empirical illustration

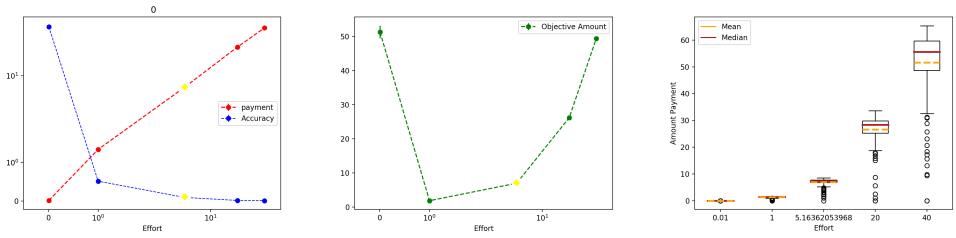
To illustrate the behavior of our payment strategy, we used synthetic problems constructed from UCI datasets (breast cancer, bank, and balance) [25]. Here is a brief description of these datasets (in all of these datasets we have reformatted the last attribute as the target value):

- **Breast cancer:** This dataset contains 10 attributes where the last one is the state of the corresponding breast cancer which can be malignant or benign. For more information, please refer to [26].
- **Balance:** This dataset contains 8 attributes where the last one is the state of the balance (right, left, balanced). For more information, please refer to [25].
- **Bank:** This dataset contains 5 attributes where the last one is the class of corresponding bank. For more information, please refer to [25]

To simulate the behavior of the workers, we added Gaussian noise to the label of each data tuple. In the first task on the balance dataset, we used the standard deviation function proportional to the inverse of the effort for the Gaussian noise.

$$\sigma(e) = \frac{1}{e}$$

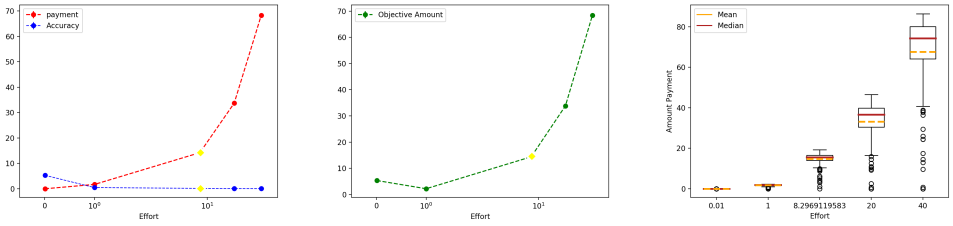
We used 200 training data points and 190 test data points to measure the accuracy of the computed metric matrix using Algorithm 3. We used Theorem 1 to compute the payments. All experiments are repeated 5 times and the average result with error bar is depicted in the figures. In this experiment, the parameter α in (5.4) was set to 1 and in the algorithm 3 we chose $\beta = 10^{-4}$ and the step size 0.00005. The effort level that minimizes (5.10) was 5.16362 (it is highlighted with yellow diamonds in the figures). Figure 4.1 shows the results. As it was expected, when we increase the effort, the accuracy and payment increases. Also, the average payment is close to the effort exerted by workers (figure 4.1c). Although equation (5.10) might not be able to give us the real optimal effort, by choosing good combinations of parameters, we are able to find a relatively good suboptimal



(a) (b) (c)

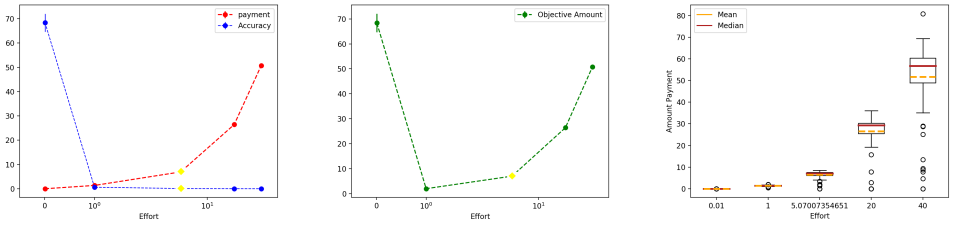
Figure 5.1: Summary of results of proposed payment scheme on balance dataset. Figure a shows payment and accuracy. Figure b shows the the value of objective (5.4). Figure c shows the distribution of individual payments. The yellow diamond denotes the effort computed by (5.10). Standard deviation function here is $\frac{1}{e}$ and the number of workers is 200.

effort level. Based on experiments, we found that the minimum of the objective function (5.4) cannot be less than 1.1, which is still close to the objective value of the effort level 5.16362. We also performed similar experiments on other datasets. In the next couple of pages (Figures 4.1 - 4.12) we have included all the results for both 100 and 200 number of workers plus two different standard deviation function $\sigma(e) = \frac{1}{e}$, $\sigma(e) = \frac{1}{\sqrt{e}}$ for all the three datasets. It can be seen that as we change the number of workers, this has a minor effect on the upper bound effort we find from (5.10). On the other hand the deviation function has more notable impacts. Moreover, the mechanism has relatively similar behavior on different datasets with similar deviation function and number of workers which shows the consistency of the mechanism we have developed here.



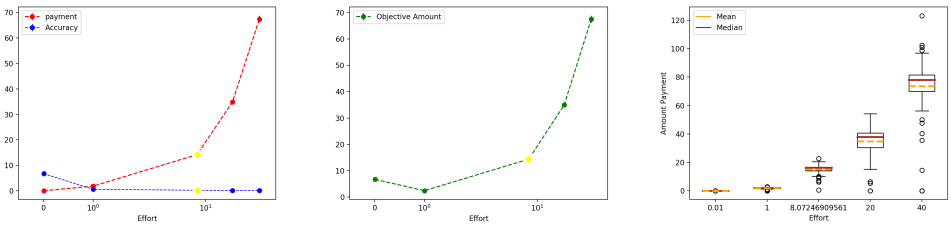
(a) (b) (c)

Figure 5.2: Summary of results of proposed payment scheme on balance dataset. Figure a shows payment and accuracy. Figure b shows the the value of objective (5.4). Figure c shows the distribution of individual payments.The yellow diamond denotes the effort computed by (5.10). Standard deviation function here is $\frac{1}{\sqrt{e}}$. Number of workers is 200.



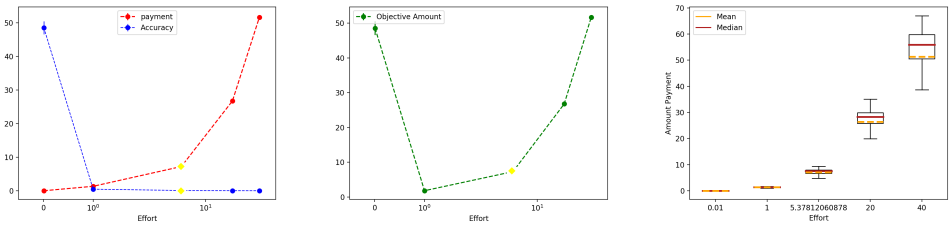
(a) (b) (c)

Figure 5.3: Summary of results of proposed payment scheme on balance dataset. Figure a shows payment and accuracy. Figure b shows the the value of objective (5.4). Figure c shows the distribution of individual payments. The yellow diamond denotes the effort computed by (5.10). Standard deviation function here is $\frac{1}{e}$ and the number of workers is 100.



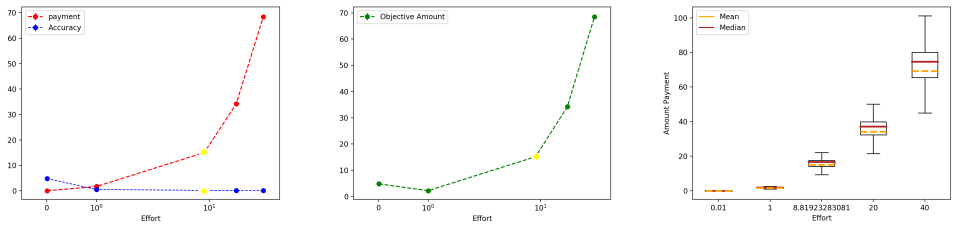
(a) (b) (c)

Figure 5.4: Summary of results of proposed payment scheme on balance dataset. Figure a shows payment and accuracy. Figure b shows the the value of objective (5.4). Figure c shows the distribution of individual payments.The yellow diamond denotes the effort computed by (5.10). Standard deviation function here is $\frac{1}{\sqrt{e}}$. Number of workers is 100.



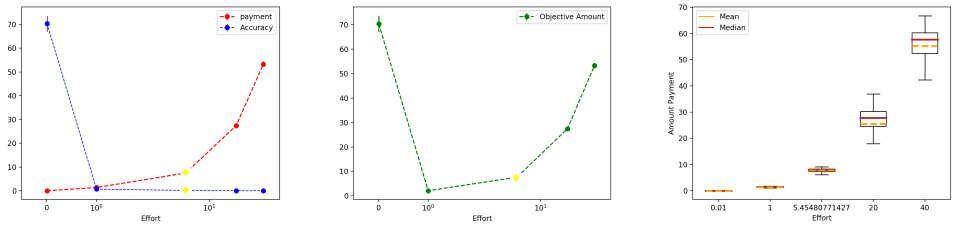
(a) (b) (c)

Figure 5.5: Summary of results of proposed payment scheme on bank dataset. Figure a shows payment and accuracy. Figure b shows the the value of objective (5.4). Figure c shows the distribution of individual payments. The yellow diamond denotes the effort computed by (5.10). Standard deviation function here is $\frac{1}{e}$ and the number of workers is 200.



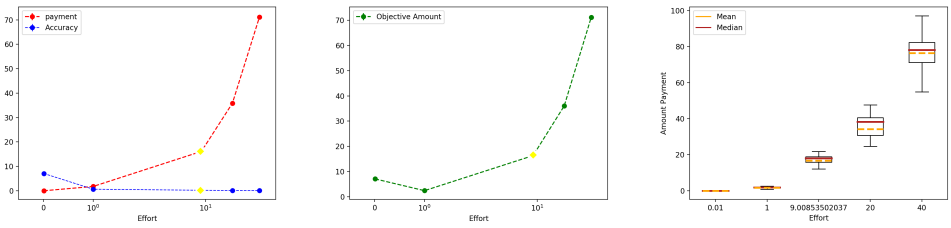
(a) (b) (c)

Figure 5.6: Summary of results of proposed payment scheme on bank dataset. Figure a shows payment and accuracy. Figure b shows the the value of objective (5.4). Figure c shows the distribution of individual payments.The yellow diamond denotes the effort computed by (5.10). Standard deviation function here is $\frac{1}{\sqrt{e}}$. Number of workers is 200.



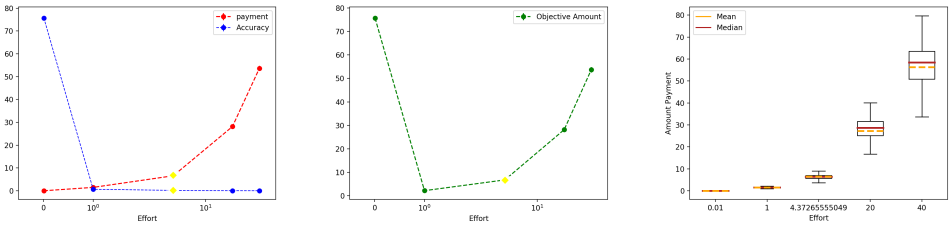
(a) (b) (c)

Figure 5.7: Summary of results of proposed payment scheme on bank dataset. Figure a shows payment and accuracy. Figure b shows the the value of objective (5.4). Figure c shows the distribution of individual payments. The yellow diamond denotes the effort computed by (5.10). Standard deviation function here is $\frac{1}{e}$ and the number of workers is 100.



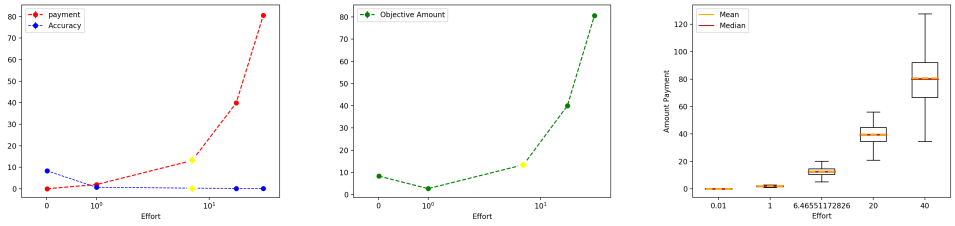
(a) (b) (c)

Figure 5.8: Summary of results of proposed payment scheme on bank dataset. Figure a shows payment and accuracy. Figure b shows the the value of objective (5.4). Figure c shows the distribution of individual payments.The yellow diamond denotes the effort computed by (5.10). Standard deviation function here is $\frac{1}{\sqrt{e}}$. Number of workers is 100.



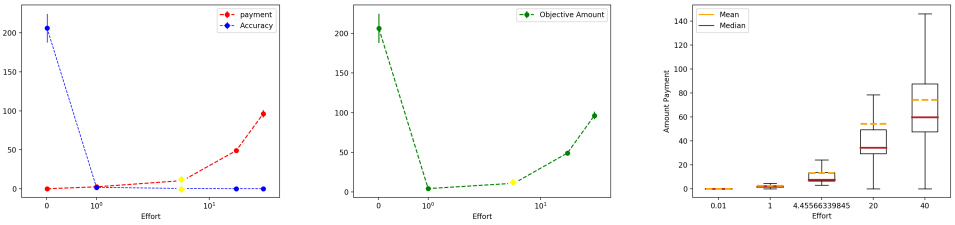
(a) (b) (c)

Figure 5.9: Summary of results of proposed payment scheme on breast dataset. Figure a shows payment and accuracy. Figure b shows the the value of objective (5.4). Figure c shows the distribution of individual payments. The yellow diamond denotes the effort computed by (5.10). Standard deviation function here is $\frac{1}{e}$ and the number of workers is 200.



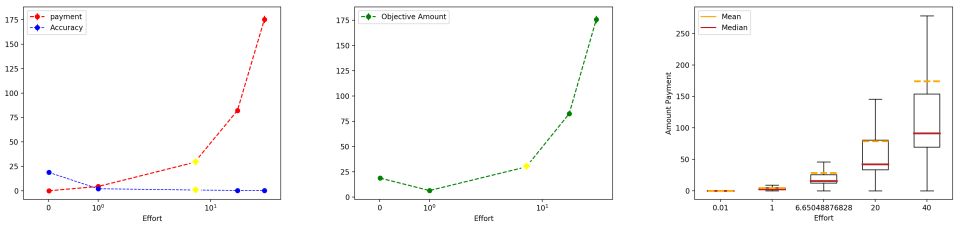
(a) (b) (c)

Figure 5.10: Summary of results of proposed payment scheme on breast dataset. Figure a shows payment and accuracy. Figure b shows the the value of objective (5.4). Figure c shows the distribution of individual payments.The yellow diamond denotes the effort computed by (5.10). Standard deviation function here is $\frac{1}{\sqrt{e}}$. Number of workers is 200.



(a) (b) (c)

Figure 5.11: Summary of results of proposed payment scheme on breast dataset. Figure a shows payment and accuracy. Figure b shows the the value of objective (5.4). Figure c shows the distribution of individual payments. The yellow diamond denotes the effort computed by (5.10). Standard deviation function here is $\frac{1}{e}$ and the number of workers is 100.



(a) (b) (c)

Figure 5.12: Summary of results of proposed payment scheme on breast dataset. Figure a shows payment and accuracy. Figure b shows the the value of objective (5.4). Figure c shows the distribution of individual payments.The yellow diamond denotes the effort computed by (5.10). Standard deviation function here is $\frac{1}{\sqrt{e}}$. Number of workers is 100.

6 Conclusion

In this thesis, we provided a mechanism to incentivize workers to provide high quality data for metric learning. Previous works have designed mechanisms for regression problems [8, 17]. Metric learning is a more challenging task, so we relaxed some of the criteria used in previous works and developed a few new tools (see chapter 5). To summarize, we provided an extension of OSE method [8] which was primarily designed for regression learning. In regression, the mechanism designed in [8] incentivizes the workers to exert the optimal effort levels. We showed that for any effort levels, there is always a mechanism that incentivizes the workers to choose those effort levels. However, unlike in the regression setting, finding the optimal effort level is difficult and may require information that is not available to the designer. Our approach is to first provide a relaxation of the problem and use the optimal solution of the relaxed problem to guide the choice of the effort levels. Although this may not give us the optimal effort level, we believe this heuristic achieves a reasonably good performance. Finally, we should mention that unlike [17] which provides some guarantees to avoid any coalition by workers, the mechanism we have designed here is incentive-compatible but not group-strategy-proof.

6.1 Future Directions

Our formulation assumes that the similarity between instances is given in real-valued, rather than binary form. We suspect that the results we obtained can be generalized to the latter case by using a formulation based on cross-entropy optimization instead of least-squares. This development is left for future work. More experimentation in a realistic setting (with human workers on Mechanical Turk, instead of simulation) would also be useful. It is interesting to consider other heuristics for choosing the effort levels, and we leave it for future work. It is also possible that by changing the whole mechanism, we can get better guarantees on the performance. As discussed earlier, we assume the workers cannot form any coalition. Although this assumption might be reasonable in many scenarios like crowd-sourcing via the internet, an interesting question is how to elevate this mechanism to avoid any coalition. Here, we tried to build a mechanism on top of a relatively sophisticated algorithm by relaxing the previous definitions and theorems already developed. In this regard, it is interesting to see to what extent this relaxation can help us in designing a mechanism for machine learning tasks. Specifically, the top-notch, highly complex, and non-linear tasks like neural network which needs massive amount of training data is an interesting area for research to see if any reasonable mechanism can be developed to gather the training data through crowd-sourcing. There are also some emerging works to do crowd-sourcing through a multiplayer online game like [12, 21] where people are involved in prediction problem while playing a game. It is also interesting to see how mechanism design can be added to these kinds of models.

Bibliography

[1] Bálint Antal and András Hajdu. An ensemble-based system for automatic screening of diabetic retinopathy. *Knowledge-Based Systems*, 60:20–27, 2014.

[2] Krzysztof R Apt and Erich Grädel. *Lectures in game theory for computer scientists*. Cambridge University Press, 2011.

[3] Aharon Bar-Hillel, Tomer Hertz, Noam Shental, and Daphna Weinshall. Learning a mahalanobis metric from equivalence constraints. *Journal of Machine Learning Research*, 6(Jun):937–965, 2005.

[4] Aurélien Bellet, Amaury Habrard, and Marc Sebban. A survey on metric learning for feature vectors and structured data. *arXiv preprint arXiv:1306.6709*, 2013.

[5] Christopher M Bishop. Pattern recognition. *Machine Learning*, 128:1–58, 2006.

[6] Stephen Boyd and Lieven Vandenberghe. Convex optimization. 2004.

[7] Lev M Bregman. The relaxation method of finding the common point of convex sets and its application to the solution of problems in convex pro-

gramming. *USSR computational mathematics and mathematical physics*, 7(3):200–217, 1967.

[8] Yang Cai, Constantinos Daskalakis, and Christos H Papadimitriou. Optimum statistical estimation with strategic data sources. In *COLT*, pages 280–296, "2015".

[9] Sumit Chopra, Raia Hadsell, and Yann LeCun. Learning a similarity metric discriminatively, with application to face verification. In *Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on*, volume 1, pages 539–546. IEEE, 2005.

[10] Edward H. Clarke. Multipart pricing of public goods. *Public Choice*, 11(1):17–33, 1971.

[11] Andrew M Colman. *Game theory and its applications: In the social and biological sciences*. Psychology Press, 2013.

[12] Seth Cooper, Firas Khatib, Adrien Treuille, Janos Barbero, Jeehyung Lee, Michael Beenen, Andrew Leaver-Fay, David Baker, Zoran Popović, et al. Predicting protein structures with a multiplayer online game. *Nature*, 466(7307):756–760, 2010.

[13] Paulo Cortez, António Cerdeira, Fernando Almeida, Telmo Matos, and José Reis. Modeling wine preferences by data mining from physicochemical properties. *Decision Support Systems*, 47(4):547–553, 2009.

[14] Thomas Cover and Peter Hart. Nearest neighbor pattern classification. *IEEE transactions on information theory*, 13(1):21–27, 1967.

[15] Jason V Davis and Inderjit Dhillon. Differential entropic clustering of multivariate gaussians. In *NIPS*, pages 337–344, 2006.

[16] Jason V Davis, Brian Kulis, Prateek Jain, Suvrit Sra, and Inderjit S Dhillon. Information-theoretic metric learning. In *Proceedings of the 24th international conference on Machine learning*, pages 209–216. ACM, 2007.

[17] Ofer Dekel, Felix Fischer, and Ariel D Procaccia. Incentive compatible regression learning. *Journal of Computer and System Sciences*, 76(8):759–777, 2010.

[18] Arpita Ghosh and Aaron Roth. Selling privacy at auction. *Games and Economic Behavior*, 91:334–346, 2015.

[19] Jacob Goldberger, Geoffrey E Hinton, Sam T. Roweis, and Ruslan 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, 2005.

[20] Theodore Groves. Incentives in teams. *Econometrica*, 41(4):617–631, 1973.

[21] Firas Khatib, Frank DiMaio, Seth Cooper, Maciej Kazmierczyk, Miroslaw Gilski, Szymon Krzywda, Helena Zabranska, Iva Pichova, James Thompson, Zoran Popović, et al. Crystal structure of a monomeric retroviral protease solved by protein folding game players. *Nature structural & molecular biology*, 18(10):1175–1177, 2011.

[22] Brian Kulis et al. Metric learning: A survey. *Foundations and Trends® in Machine Learning*, 5(4):287–364, 2013.

[23] Brian Kulis, Mátyás Sustik, and Inderjit Dhillon. Learning low-rank kernel matrices. In *Proceedings of the 23rd international conference on Machine learning*, pages 505–512. ACM, 2006.

[24] Solomon Kullback and Richard A Leibler. On information and sufficiency. *The annals of mathematical statistics*, 22(1):79–86, 1951.

[25] M. Lichman. UCI machine learning repository, 2013.

[26] Olvi L Mangasarian, W Nick Street, and William H Wolberg. Breast cancer diagnosis and prognosis via linear programming. *Operations Research*, 43(4):570–577, 1995.

[27] Roger A McCain. *Game theory: A nontechnical introduction to the analysis of strategy*. World Scientific Publishing Co Inc, 2014.

[28] Roger B Myerson. Game theory: analysis of conflict. 1991. *Cambridge: Mass, Harvard University*.

[29] Noam Nisan, Tim Roughgarden, Eva Tardos, and Vijay V Vazirani. *Algorithmic game theory*, volume 1. Cambridge University Press Cambridge, 2007.

[30] Peter C Ordeshook. *Game theory and political theory: An introduction*. Cambridge University Press, 1986.

[31] Martin J Osborne and Ariel Rubinstein. *A course in game theory*. MIT press, 1994.

[32] Ruslan Salakhutdinov and Geoffrey E Hinton. Learning a nonlinear embedding by preserving class neighbourhood structure. In *AISTATS*, volume 11, 2007.

[33] Matthew Schultz and Thorsten Joachims. Learning a distance metric from relative comparisons. In *Advances in neural information processing systems*, pages 41–48, 2004.

[34] Shai Shalev-Shwartz and Shai Ben-David. *Understanding machine learning: From theory to algorithms*. Cambridge university press, 2014.

[35] Shai Shalev-Shwartz, Yoram Singer, and Andrew Y Ng. Online and batch learning of pseudo-metrics. In *Proceedings of the twenty-first international conference on Machine learning*, page 94. ACM, 2004.

[36] Adish Singla and Andreas Krause. Truthful incentives for privacy tradeoff: Mechanisms for data gathering in community sensing. 2013.

[37] Koji Tsuda, Gunnar Rätsch, and Manfred K Warmuth. Matrix exponentiated gradient updates for on-line learning and bregman projection. *Journal of Machine Learning Research*, 6(Jun):995–1018, 2005.

[38] William Vickrey. Counterspeculation, auctions, and competitive sealed tenders. *The Journal of Finance*, 16(1):8–37, 1961.

[39] David S Watkins. *Fundamentals of matrix computations*, volume 64. John Wiley & Sons, 2004.

[40] Kilian Q Weinberger, John Blitzer, and Lawrence K Saul. Distance metric learning for large margin nearest neighbor classification. In *Advances in neural information processing systems*, pages 1473–1480, 2006.

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

[42] Eric P Xing, Andrew Y Ng, Michael I Jordan, and Stuart Russell. Distance metric learning with application to clustering with side-information. In *NIPS*, volume 15, page 12, 2002.

[43] Liu Yang and Rong Jin. Distance metric learning: A comprehensive survey. *Michigan State Universiy*, 2(2), 2006.