Inference Driven Metric Learning (IDML) for Graph Construction

Paramveer S. Dhillon[1], Partha Pratim Talukdar[1], and Koby Crammer[2]

[1] CIS Department, University of Pennsylvania, Philadelphia, PA 19104, U.S.A.
[2] Electrical Engineering Department, The Technion, Haifa 32000, Israel.

Abstract. Graph-based semi-supervised learning (SSL) methods usually consist of two stages: in the first stage, a graph is constructed from the set of input instances; and in the second stage, the available label information along with the constructed graph is used to assign labels to the unlabeled instances.
Most of the previously proposed graph construction methods are unsupervised in nature, as they ignore the label information already present in the SSL setting in which they operate. In this paper, we explore how available labeled instances can be used to construct a better graph which is tailored to the current classification task. To achieve this goal, we evaluate effectiveness of various *supervised* metric learning algorithms during graph construction. Additionally, we propose a new metric learning framework: Inference Driven Metric Learning (IDML), which extends existing supervised metric learning algorithms to exploit widely available unlabeled data during the metric learning step itself. We provide extensive empirical evidence demonstrating that inference over graph constructed using IDML learned metric can lead to significant reduction in classification error, compared to inference over graphs constructed using existing techniques. Finally, we demonstrate how active learning can be successfully incorporated within the the IDML framework to reduce the amount of supervision necessary during graph construction.

1 Introduction

Supervised machine learning algorithms have been quite successful in a variety of domains, ranging from Natural Language Processing to Bioinformatics. Unfortunately, such algorithms require large amounts of labeled data which is expensive and time consuming to prepare. To addresses this issue, recent research has focused on Semi-Supervised Learning (SSL) algorithms, which learn from limited amounts of labeled data combined with widely available unlabeled data. In particular, graph based SSL algorithms have recently been successfully used in different tasks, e.g. the work of [1].

Given a set of instances, some of which are labeled while the rest are unlabeled, most graph based SSL algorithms first construct a graph where each node corresponds to an instance. Similar nodes are connected by an edge, with edge weight encoding the degree of similarity. Once the graph is constructed, the

nodes corresponding to labeled instances are injected with the corresponding label. Using this initial label information along with the graph structure, graph based SSL algorithms assign labels to all unlabeled nodes in the graph. Most of the graph based SSL algorithms are iterative and also parallelizable, making them suitable for SSL setting where vast amounts of unlabeled data is usually available. Examples of a few graph-based SSL algorithms include Gaussian Random Fields (GRF) [2], Local and Global Consistency (LGC) [3], Manifold Regularization (MR) [4], Quadratic Criteria (QR) [5], Graph Transduction via Alternating Minimization (GTAM) [6], Adsorption and its variants [7], Measure Propagation [8].

Most of the graph based SSL algorithms mentioned above concentrate primarily on the label inference part, i.e. assigning labels to nodes *once the graph has already been constructed*, with very little emphasis on construction of the graph itself. Only recently, the issue of graph construction has begun to receive attention [9–11]. A b-matching (BM) based method for regular graph construction is presented by [10], where each node in the constructed graph is required to have the same degree (b). [11] proposed that each node is required to have a minimum weighted degree of 1, which is relaxed in some cases. Both of these methods emphasize on constructing graphs which satisfy certain structural properties (e.g. degree constraints on each node). Since our focus is on SSL, a certain number of labeled instances are available at our disposal. However, the graph construction methods mentioned above are all unsupervised in nature, i.e. they do not utilize available label information during the graph construction process. In fact, this is left as an open question for future work in [11]. In this paper, we propose a fill to gap and explore how the available label information can be used for graph construction in graph based SSL. In particular, we focus on *learning* a distance metric using available label information, which can then be used to set the edge weights on the constructed graph.

Once the nodes in a graph are fixed, the rest of graph construction process reduces to setting the edge weights on a complete graph, with an edge weight of 0 encoding absence of the corresponding edge. Over the years, several *supervised* metric learning algorithms have been developed, with Information-theoretic Metric Learning (ITML) [12] and large margin distance metric learning (LMNN) [13] being two recent and state-of-the-art metric learning methods. The distance metric learned by such methods can be used to compute similarity[3] between a pair of instances, and subsequently set as the similarity edge weight on the edge connecting the instances in the graph.

In this paper, we make the following contributions:

1. In contrast to previously proposed unsupervised graph construction methods which ignore already available label information, we empirically validate effectiveness of *supervised* distance metric learning algorithms during graph construction for graph based SSL.

[3] By using an appropriate distance to similarity mapping, e.g. Gaussian kernel which we review in Section 5.

2. In order to exploit widely available unlabeled data during distance metric learning, we propose a new metric learning framework: Inference Driven Metric Learning (IDML). We provide extensive empirical evidence demonstrating that inference over graph constructed using IDML learned metric can lead to significant reduction in classification error, compared to inference over graphs constructed using existing techniques.
3. Finally, we demonstrate how active learning can be incorporated within the IDML framework to reduce the amount of supervision necessary during graph construction.

The paper is organized as follows: in Section 3, we review two recently proposed metric learning algorithms, ITML and LMNN; in Section 4, we present the Inference-Driven Metric Learning (IDML) framework; in Section 5, we show how learned distance metrics can be used to construct graphs, and subsequently perform label inference on such graphs; in Section 6, we report experimental results on various real-world datasets; and finally in Section 7, we present concluding remarks with directions for future work.

2 Notations

Let X be the $d \times n$ matrix of n instances in a d-dimensional space. Out of the n instances, n_l instances are labeled, while the remaining n_u instances are unlabeled, with $n = n_l + n_u$. $G = (V, E, W)$ is the graph we are interested in constructing; where V is the set of vertices with $|V| = n$, E is the set of edges, and W is the symmetric $n \times n$ matrix of edge weights that we would like to learn. W_{ij} is the weight of edge (i, j), and also the similarity between instances x_i and x_j. S is a $n \times n$ diagonal matrix with $S_{ii} = 1$ iff instance x_i is labeled. m is the total number of labels. Y is the $n \times m$ matrix storing training label information, if any. $\hat{Y}$ is the $n \times m$ matrix of estimated label information, i.e. output of any inference algorithm (e.g. see Section 5). A is a positive definite matrix of size $d \times d$, which parametrizes the (squared) Mahalanobis distance (Equation 1).

3 Metric Learning Review

We now review some of the recently proposed *supervised* methods for learning Mahalanobis distance between instance pairs. We shall concentrate on learning matrix A which parametrizes the distance, $d_A(x_i, x_j)$, between instances x_i and x_j.

$$d_A(x_i, x_j) = (x_i - x_j)^\top A (x_i - x_j) \tag{1}$$

Since A is positive definite, we can decompose it as $A = P^\top P$, where P is another matrix of size $d \times d$. We can then rewrite Equation 1 as,

$$\begin{aligned} d_A(x_i, x_j) &= (x_i - x_j)^\top P^\top P (x_i - x_j) \\ &= (Px_i - Px_j)^\top (Px_i - Px_j) \\ &= d_I(Px_i, Px_j) \end{aligned} \tag{2}$$

Hence, computing Mahalanobis distance w.r.t. A is equivalent to first projecting the instances into a new space using an appropriate transformation matrix P and then computing Euclidean distance in the linearly transformed space.

3.1 Information-Theoretic Metric Learning (ITML)

Information-Theoretic Metric Learning (ITML) [12] assumes the availability of prior knowledge about inter-instance distances. In this scheme, two instances are considered similar if the Mahalanobis distance between them is upper bounded, i.e. $d_A(x_i, x_j) \leq u$, where u is a non-trivial upper bound. Similarly, two instances are considered dissimilar if the distance between them is larger than certain threshold l, i.e. $d_A(x_i, x_j) \geq l$. Similar instances are represented by set S, while dissimilar instances are represented by set D.

In addition to prior knowledge about inter-instance distances, sometimes prior information about the matrix A, denoted by A_0, itself may also be available. For example, Euclidean distance (i.e. $A_0 = I$) may work well in some domains. In such cases, we would like the learned matrix A to be as close as possible to the prior matrix A_0. ITML combines these two types of prior information, i.e. knowledge about inter-instance distances, and prior matrix A_0, in order to learn the matrix A by solving the optimization problem shown in (3).

$$\begin{aligned} \min_{A \succeq 0} \ & D_{\text{ld}}(A, A_0) \\ \text{s.t.} \ & \text{tr}\{A(x_i - x_j)(x_i - x_j)^\top\} \leq u, \ \forall (i,j) \in S \\ & \text{tr}\{A(x_i - x_j)(x_i - x_j)^\top\} \geq l, \ \forall (i,j) \in D \end{aligned} \tag{3}$$

where $D_{\text{ld}}(A, A_0) = \text{tr}(AA_0^{-1}) - \log\det(AA_0^{-1}) - n$, is the LogDet divergence.

It may not always be possible to *exactly* solve the optimization problem shown in (3). To handle such situations, slack variables may be introduced to the ITML objective. Following the notation in [12], let $c(i,j)$ be the index of the $(i,j)^{th}$ constraint, and let ξ be a vector of length $|S| + |D|$ of slack variables. ξ is initialized to ξ_0, whose components equal u for similarity constraints, and l for dissimilarity constraints. The modified ITML objective involving slack variables is shown in (4).

$$\begin{aligned} \min_{A \succeq 0, \xi} \ & D_{\text{ld}}(A, A_0) + \gamma \cdot D_{\text{ld}}(\xi, \xi_0) \\ \text{s.t.} \ & \text{tr}\{A(x_i - x_j)(x_i - x_j)^\top\} \leq \xi_{c(i,j)}, \ \forall (i,j) \in S \\ & \text{tr}\{A(x_i - x_j)(x_i - x_j)^\top\} \geq \xi_{c(i,j)}, \ \forall (i,j) \in D \end{aligned} \tag{4}$$

where γ is a hyperparameter which determines the importance of violated constraints. To solve the optimization problem in (4), an algorithm involving repeated Bregman projections is presented in [12], which we use for the experiments reported in this paper.

3.2 Large Margin Nearest Neighbor (LMNN)

A large margin based method for learning a Mahalanobis distance metric from labeled instances is presented in [13]. The objective here is to learn a linear

transformation of input instances so that the k nearest neighbors (in the linearly transformed space) of each instance share the same label. This metric learning algorithm is tuned towards achieving good performance on k-NN classification. From now on, we shall refer to this large margin metric learning algorithm in [13] as LMNN.

A set of labeled instances and a hyperparameter k is input to LMNN. For each labeled instance, LMNN first determines k *target neighbors*. Target neighbors of an instance x_i are the instances that LMNN desires to be closest to x_i in the linearly transformed target space. The target neighbors are fixed at the start of the algorithm and are not changed subsequently. In many domains, the target neighbors may be selected based on prior knowledge. Another alternative is to select the target neighbors based on Euclidean neighbors in the original space, which is the strategy we use for the experiments in this paper (as in [13]).

Once the target neighbors for each instance is determined, LMNN learns a linear transformation so that for each point, the closest point from a different class (called *impostor*) is further away by a margin from the farthest target neighbor of the point, with all distances measured in the linearly transformed space. In other words, an instance's target neighbors define a neighborhood where instances from other classes are not allowed. In order to achieve this goal, LMNN solves a convex optimization problem which is reproduced from [13] in (5).

$$\min_{A,\xi} \; (1-\mu) \sum_{i,j \in N(i)} d_A(x_i, x_j) + \mu \sum_{i,j \in N(i),l} (1 - y_{il})\, \xi_{ijl} \tag{5}$$

$$\text{s.t.} \;\; d_A(x_i, x_l) - d_A(x_i, x_j) \geq 1 - \xi_{ijl} \tag{6}$$

$$A \succeq 0, \xi_{ijl} \geq 0$$

where d_A is the Mahalanobis distance w.r.t A as shown in Equation 1, $N(i)$ consists of x_i's target neighbors, $\{y_{il}\}$ are indicator variables with $y_{il} = 1$ iff $y_i = y_l$, $\{\xi_{ijl}\}$ are slack variables which allow violation of the large margin constraint (6), but with a cost whose contribution to the objective is controlled by the hyperparameter μ. The large margin constraint (6) enforces the requirement that an impostor, x_l (impostors are indexed by l), should be at a *safe* distance away from x_i. The first term in (5) represents the fact that LMNN tries to *pull* together all instances within a target neighborhood. As noted by [13], as most input instance pairs, x_i and x_l, are likely to be well separated from each other, most of the slack variables $\{\xi_{ijl}\}$ are never going to get assigned positive values. This sparseness property can be exploited to develop faster, special purpose solver to optimize the LMNN objective shown above [13].

After optimizing the LMNN objective, we obtain a positive semi-definite matrix A which can be used to compute the Mahalanobis distance between any two instances using Equation 1.

Algorithm 1 Inference Driven Metric Learning (IDML)
Input: instances X, training labels Y, training instance indicator S, label entropy threshold β, neighborhood size k
Output: Mahalanobis distance parameter A

1: $\hat{Y} \leftarrow Y$, $\hat{S} \leftarrow S$
2: **repeat**
3: $A \leftarrow \textsc{MetricLearner}(X, \hat{S}, \hat{Y})$
4: $W \leftarrow \textsc{ConstructKnnGraph}(X, A, k)$
5: $\hat{Y}' \leftarrow \textsc{GraphLabelInference}(W, \hat{S}, \hat{Y})$
6: $U \leftarrow \textsc{SelectLowEntInstances}(\hat{Y}', \hat{S}, \beta)$
7: $\hat{Y} \leftarrow \hat{Y} + U\hat{Y}'$
8: $\hat{S} \leftarrow \hat{S} + U$
9: **until** convergence (i.e. $U_{ii} = 0,\ \forall i$)
10: return A

4 Inference Driven Metric Learning (IDML)

The metric learning algorithms we have reviewed so far, ITML in Section 3.1 and LMNN in Section 3.2, are supervised in nature, and hence they do not exploit widely available unlabeled data. In this section, we present Inference Driven Metric Learning (IDML) (Algorithm 1), a metric learning framework which combines existing *supervised* metric learning algorithm (such as ITML or LMNN) along with *trasnductive* graph-based label inference to learn a new distance metric from labeled as well as unlabeled data combined. In self-training styled iterations, IDML alternates between metric learning and label inference; with output of label inference used during next round of metric learning, and so on.

IDML starts out with the assumption that existing supervised metric learning algorithms, such as ITML and LMNN, can learn a better metric if the number of available labeled instances is increased. Since we are focusing in the SSL setting with n_l labeled and n_u unlabeled instances, the idea is to automatically label the unlabeled instances using a graph based SSL algorithm, and then include instances with low assigned label entropy (i.e. high confidence label assignments) in the next round of metric learning. The number of instances added in each iteration depends on the threshold β[4]. This process is continued until no new instances can be added to the set of labeled instances, which can happen when either all the instances are already exhausted, or when none of the remaining unlabeled instances can be assigned labels with high confidence.

The IDML framework is presented in Algorithm 1. In Line 1, any supervised metric learner, such as ITML or LMNN, may be used as the MetricLearner. Using the distance metric learned in Line 1, a new k-NN graph is constructed in

[4] During the experiments in Section 6, we set $\beta = 0.05$ and observed that the low entropy instances, which are selected for inclusion in next iteration of metric learning, are usually classified with fairly high accuracy.

Line 1, whose edge weight matrix is stored in W. In Line 1, GraphLabelInference optimizes over the newly constructed graph the GRF objective [2] shown in (7).

$$\min_{\hat{Y}'} \quad \text{tr}\{\hat{Y}'^{\top} L \hat{Y}'\}, \quad \text{s.t.} \quad \hat{S}\hat{Y} = \hat{S}\hat{Y}' \tag{7}$$

where $L = D - W$ is the (unnormalized) Laplacian, and D is a diagonal matrix with $D_{ii} = \sum_j W_{ij}$. The constraint, $\hat{S}\hat{Y} = \hat{S}\hat{Y}'$, in (7) makes sure that labels on training instances are not changed during inference. In Line 1, a currently unlabeled instance x_i (i.e. $\hat{S}_{ii} = 0$) is considered a new labeled training instance, i.e. $U_{ii} = 1$, for next round of metric learning if the instance has been assigned labels with high confidence in the current iteration, i.e. if its label distribution has low entropy (i.e. $\text{Entropy}(\hat{Y}'_{i:}) \leq \beta$). Finally in Line 1, training instance label information is updated. This iterative process is continued till no new labeled instance can be added, i.e. when $U_{ii} = 0 \ \forall i$. IDML returns the learned matrix A which can be used to compute Mahalanobis distance using Equation 1.

Lemma: IDML (Algorithm 1) will terminate after at most n iterations.

Proof: The GraphLabelInference method used in IDML ensures that labels of already labeled instances are not changed (see (7)). Hence, once an instance is assigned a label in one of the iterations (Line 1), its label is not changed in all subsequent iterations. Since there are only finitely many points, hence only a finite number of new points can be selected in Line 1 of IDML. Hence, IDML can iterate for at most n iterations, where n is the total number of instances.

4.1 Relationship to Other Methods

IDML is similar in spirit to the EM-based HMRF-KMeans algorithm in [14], which focuses on integrating constraints and metric learning for semi-supervised clustering. However, there are crucial differences. Firstly, the inference algorithm in [14] is parametric (k-Means), while the inference in IDML is non-parametric (graph based). Secondly, and more importantly, the label constraints in [14] are hard constraints which are fixed a-priori and are not changed during EM iterations. In case of IDML , the graph structure induced in each iteration (Line 1 in Algorithm 1) imposes a Manifold Regularization (MR) [4] style smoothing penalty, as shown in (7). Hence, compared to the hard and fixed constraints in HMRF-KMeans, IDML constraints are soft and new constraints are added in each iteration of the algorithm.

The main difference that separates IDML from previous work on supervised metric learning [12, 13, 15] is the use of unlabeled data during metric learning. Moreover, in all of these previously proposed algorithms, the constraints used during metric learning are fixed a-priori, as they are usually derived from labeled instances which don't change during the course of algorithm; while the constraints in IDML are adaptive and new constraints are added in each iteration, when automatically labeled instances are included in each iteration (Line

1). In Section 6, we shall see experimental evidence that these additional constraints can be quite effective in improving IDML's performance compared to ITML or LMNN.

The distance metric learned using IDML can be used to compute inter-instance distance in b-matching (BM) [10], and in this way the two approaches can compliment one another.

5 Graph Construction & Inference

Gaussian kernel [2, 16], a widely used measure of similarity between data instances, can be used to compute edge weights as shown in Equation 8.

$$W_{ij} = \exp\left(\frac{-d_A(x_i, x_j)}{2\sigma^2}\right) \tag{8}$$

where $d_A(x_i, x_j)$ is the distance measure between instances x_i and x_j, and σ is the kernel bandwidth parameter. By setting $A = I$, we get the standard Euclidean distance in input space, which has traditionally been used in previous work [2]. Instead, by using A as learned by metric learning algorithms in Section 3.1, 3.2 or 4, we can define a similarity metric which is tailored to the current classification task.

Setting edge weights directly using Equation 8 will result in a complete graph, where any two pair of nodes will be connected with a positively weighted edge. This may be undesirable as a dense graph may slow down any subsequent inference. We may sparsify the graph by retaining only edges to k nearest neighbors of each node, and dropping all other edges (i.e. setting corresponding edge weights to 0), a commonly used graph sparsification strategy. As an alternative, other recently proposed method (e.g. b-matching [10]) may also be used to generate a sparse graph.

Inference: With the graph $G = (V, E, W)$ constructed, we may now perform inference over this graph to assign labels to all n_u unlabeled nodes. Any of the several graph based SSL algorithms mentioned in Section 1 may be used for this task. For the experiments in this paper, we use the GRF algorithm [2] which minimizes the optimization problem shown in (7).

6 Experiments

In this section, we evaluate the importance of metric learning for constructing graphs, where the constructed graphs, along with the labeled instances, are used to classify initially unlabeled instances. We evaluate performance using classification error (i.e. 1 - accuracy) . We use Gaussian kernel to set edge weights, followed by k-NN graph sparsification, as described in Section 5. The hyperparameters $k \in \{2, 5, 10, 50, 100, 200, 500, 1000\}$ and the Gaussian kernel bandwidth multiplier[5], $\rho \in \{1, 2, 5, 10, 50, 100\}$, are tuned on a separate development set. In

[5] $\sigma = \rho\ \sigma_0$, where ρ is the tuned multiplier, and σ_0 is set to average distance.

Dataset	Type	Dimension	Balanced
Amazon	Real-World	132553	Yes
Newsgroups	Real-World	32965	Yes
Reuters	Real-World	10964	Yes
EnronA	Real-World	5802	No
Text	Real-World	11960	Yes
USPS	Real-World	241	No
BCI	Real-World	117	Yes
Digit	Artificial	241	No

Table 1. Description of datasets used in experiments in Section 6. All datasets were binary, with 1500 total instances in each, except BCI which had 400 instances.

all experiments, GRF (see Section 5) is used. Similarly, we use standard implementations of ITML and LMNN made available by respective authors. Brief description of the datasets used for experiments in this section is presented in Table 1; the first four datasets are obtained from [17], while the rest are from [18].

6.1 Results on Real-World Datasets

Datasets	Original	RP	PCA	ITML	LMNN	IDML-LM	IDML-IT
Amazon	0.4686	0.4681	0.2329	0.1542	0.2069	0.2065	**0.1537**
Newsgroups	0.3648	0.3778	0.3490	0.1791	0.2860	0.2791	**0.1650**
Reuters	0.2912	0.5016	0.5016	0.1300	0.4991	0.4003	**0.1264**
EnronA	0.3514	0.3514	0.3200	0.1855	0.3124	0.3096	**0.1671**
Text	0.4541	0.4954	0.4835	0.3140	0.3297	0.3247	**0.3121**
USPS	0.1536	0.1667	–	0.1484	0.1388	**0.1388**	0.1467
BCI	0.4693	0.4678	–	0.4264	0.4122	**0.4093**	0.4196
Digit	**0.0246**	0.0357	–	0.0438	0.1186	0.0991	0.0381

Table 2. Comparison of transductive classification performance over graphs constructed using different methods (see Section 6.1), with $n_l = 50$ and $n_u = 1450$. All results are averaged over four trials. IDML-LM and IDML-IT are the proposed methods.

In this section, we compare the following methods of estimating A, which in turn is used in Equation 8 to estimate edge weights:

Original: We set $A = I_{d \times d}$, i.e. the data is not transformed and Euclidean distance in the input space is used to compute distance between instances.

RP: The data is first projected into a lower dimensional space of dimension $d' = \frac{\log n}{\epsilon^2 \log \frac{1}{\epsilon}}$ using the Random Projection (RP) method [19]. We set $A = R^\top R$, where R is the projection matrix used by RP. ϵ was set to 0.25 for the experiments in Section 6.1.

Datasets	Original	RP	PCA	ITML	LMNN	IDML-LM	IDML-IT
Amazon	0.4046	0.3964	0.1554	0.1418	0.2405	0.2004	**0.1265**
Newsgroups	0.3407	0.3871	0.3098	**0.1664**	0.2172	0.2136	**0.1664**
Reuters	0.2928	0.3529	0.2236	0.1088	0.3093	0.2731	**0.0999**
EnronA	0.3246	0.3493	0.2691	0.2307	0.1852	**0.1707**	0.2179
Text	0.4523	0.4920	0.4820	0.3072	0.3125	0.3125	**0.2893**
USPS	**0.0639**	0.0829	–	0.1096	0.1336	0.1225	0.0834
BCI	0.4508	0.4692	–	0.4217	0.3058	**0.2967**	0.4081
Digit	**0.0218**	0.0250	–	0.0281	0.1186	0.0877	0.0281

Table 3. Comparison of transductive classification performance over graphs constructed using different methods (see Section 6.1), with $n_l = 100$ and $n_u = 1400$. All results are averaged over four trials. IDML-LM and IDML-IT are the proposed methods.

PCA: Instances are first projected into a lower dimensional space using Principal Components Analysis (PCA) . For all experiments, dimensionality of the projected space was set at 250[6]. We set $A = P^\top P$, where P is the projection matrix generated by PCA.
LMNN: A is learned by applying LMNN (see Section 3.2) on the PCA projected space (above).
ITML: A is learned by applying ITML (see Section 3.1) on the PCA projected space (above).
IDML-LM: A is learned by applying IDML (Algorithm 1) on the PCA projected space (above); with LMNN used as MetricLearner in IDML.
IDML-IT: A is learned by applying IDML (Algorithm 1) (see Section 4) on the PCA projected space (above); with ITML used as MetricLearner in IDML.

Experimental results on datasets in Table 1 with 50 and 100 labeled instances (n_l) are shown in Tables 2 and 3, respectively. From these we observe that, constructing a graph using a learned metric can significantly improve performance (in 13 out of 16 cases). We consistently find graphs constructed using IDML-IT to be the most effective. This is particularly true in case of high-dimensional datasets where distances in the original input space are often unreliable because of curse of dimensionality.

We tried our best to include comparisons with graphs constructed using b-matching [10], however that often resulted in disconnected graphs which the GRF code we used (obtained from [2]) was unable to handle.

6.2 Sensitivity to Noisy Features

In order to evaluate sensitivity of IDML-IT, the best performing method from Section 6.1, to input data noise and increasing dimensions, we generated four

[6] PCA was not performed on USPS, BCI and Digit as they already had dimension lower than 250.

Dataset	dim	Original	ITML	IDML-IT
Synth-2	2	0.1188	0.0431	**0.0163**
Synth-5	5	0.3894	0.1275	**0.1156**
Synth-10	10	0.4669	0.2813	**0.2025**
Synth-20	20	0.4900	0.3000	**0.2850**

Table 4. Results on synthetic datasets (see Section 6.2) with $n_l = 100$. All results are averaged over four trials.

synthetic datasets: Synth-$\{2, 5, 10, 20\}$, each consisting of 500 instances in R^d ($d = 2, 5, 10, 20$, respectively), where the first two dimensions were sampled from a 45^o rotated Gaussian distribution with standard deviation 1. The remaining $d-2$ dimensions were sampled independently from Gaussian distributions $N(0, 2)$. Hence, in these datasets, only the first two dimensions are informative, while the rest of the dimensions just add noise.

Experimental results on these synthetic datasets with 100 labeled instances are presented in Table 4. From Table 4, we observe that label inference over graphs constructed using IDML-IT achieve lowest classification error across all datasets. We also observe that as the number of noisy dimensions were increased (from 0 in Synth-2 to 18 in Synth-20), performance in case of Original deteriorated significantly, while IDML-IT is much more resilient to noise. This demonstrates the fact that the learned distance metric is able to filter out the noisy dimensions and concentrate more on the first two informative dimensions, which is essential in these datasets. It is interesting to note that IDML-IT is very effective even in the absence of noise (Synth-2).

6.3 Active Learning

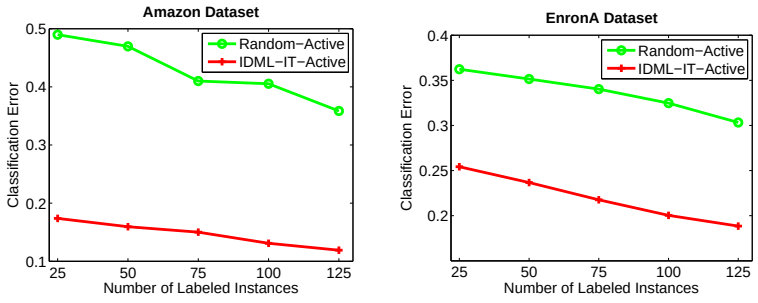

Fig. 1. Results from active learning experiments in Section 6.3. All results are averaged over four trials. Active learning using IDML-IT-Active is the bottom line.

In all the experiments reported so far in Sections 6.1 and 6.2, the n_l training instances, usually labeled by humans, are fixed at the start and are not changed during course of the algorithms. In case of IDML-LM and IDML-IT, this initial set of labeled instances is augmented by adding *automatically* labeled instances with low label entropy (Line 1 in Algorithm 1).

In this section, we take an active learning view and explore whether the instances to be labeled (by a human or an oracle) can be actively selected using a modified version of IDML-IT, which we call IDML-IT-Active. Instead of adding instances with low label entropy as in IDML-IT; IDML-IT-Active selects, at each iteration, top-r instances with highest label entropy which are then labeled by a human or an oracle. These additional r oracle labeled (as opposed to *automatic* labeling in IDML-IT) instances are used in the next iteration of metric learning. In Figure 1, we compare label inference over graphs constructed using IDML-IT-Active, to graphs constructed in the original space (i.e. $A = I$) with equivalent number of randomly selected and labeled instances, called Random-Active in Figure 1. Across both the datasets, we observe that 25-50 actively selected instances using IDML-IT-Active performs better than 125 randomly selected and labeled instances, thereby drastically reducing the amount of supervision necessary to attain a certain level of classification performance.

7 Conclusion

In this paper, we explored how labeled instances, available in the SSL setting, can be used to construct a better graph for improved classification accuracy in graph based SSL. We demonstrated effectiveness of various *supervised* metric learning algorithms to learn distance metrics for graph construction, and subsequent inference over such constructed graphs. To the best of our knowledge, this is the first study of its kind where effectiveness of metric learning for graph construction is studied. Additionally, we proposed a new metric learning framework: Inference Driven Metric Learning (IDML), which extends existing supervised metric learning algorithms to exploit widely available unlabeled data during the metric learning step itself. Through a set of extensive experiments on synthetic as well as various real-world datasets, we demonstrated that inference over graphs constructed using IDML can lead to significant reduction in classification error, compared to inference over graphs constructed either with a supervised metric learner in isolation, or without using any label information at all. Finally, we demonstrated how labeled instances can be actively selected within the the IDML framework to reduce the amount of supervision necessary during graph construction.

Encouraged by these promising initial results, we plan to investigate further into the Inference Driven Metric Learning (IDML) framework, and in particular pose it as an optimization of a regularized metric learning objective.

References

1. Subramanya, A., Bilmes, J.A.: The semi-supervised switchboard transcription project. In: INTERSPEECH, Brighton, UK (September 2009)
2. Zhu, X., Ghahramani, Z., Lafferty, J.: Semi-supervised learning using Gaussian fields and harmonic functions. In: ICML. (2003)
3. Zhou, D., Bousquet, O., Lal, T., Weston, J., Schlkopf, B.: Learning with local and global consistency. In: NIPS, The MIT Press (2004)
4. Belkin, M., Niyogi, P., Sindhwani, V.: Manifold regularization: A geometric framework for learning from labeled and unlabeled examples. The Journal of Machine Learning Research **7** (2006) 2434
5. Bengio, Y., Delalleau, O., Le Roux, N.: Label propagation and quadratic criterion. Semi-supervised learning (2006) 193–216
6. Wang, J., Jebara, T., Chang, S.: Graph transduction via alternating minimization. In: ICML. (2008)
7. Talukdar, P., Crammer, K.: New Regularized Algorithms for Transductive Learning. In: ECML-PKDD, Springer (2009)
8. Subramanya, A., Bilmes, J.A.: Entropic graph regularization in non-parametric semi-supervised classification. In: NIPS, Vancouver, Canada (December 2009)
9. Wang, F., Zhang, C.: Label propagation through linear neighborhoods. In: ICML. (2006)
10. Jebara, T., Wang, J., Chang, S.: Graph construction and b-matching for semi-supervised learning. In: ICML. (2009)
11. Daitch, S., Kelner, J., Spielman, D.: Fitting a graph to vector data. In: ICML. (2009)
12. Davis, J., Kulis, B., Jain, P., Sra, S., Dhillon, I.: Information-theoretic metric learning. In: ICML. (2007)
13. Weinberger, K., Saul, L.: Distance metric learning for large margin nearest neighbor classification. The Journal of Machine Learning Research (2009)
14. Bilenko, M., Basu, S., Mooney, R.: Integrating constraints and metric learning in semi-supervised clustering. In: ICML. (2004)
15. Jin, R., Wang, S., Zhou, Y.: Regularized Distance Metric Learning: Theory and Algorithm. In: NIPS. (2009)
16. Belkin, M., Niyogi, P.: Towards a theoretical foundation for Laplacian-based manifold methods. Journal of Computer and System Sciences **74**(8) (2008) 1289–1308
17. Crammer, K., Dredze, M., Kulesza, A.: Multi-Class Confidence Weighted Algorithms. (2009)
18. Chapelle, O., Scholkopf, B., Zien, A.: Semi-supervised learning. MIT Press (2006)
19. Bingham, E., Mannila, H.: Random projection in dimensionality reduction: applications to image and text data. In: ACM SIGKDD. (2001)