Escaping the Curse of Dimensionality in Similarity Learning: Efficient Frank-Wolfe Algorithm and Generalization Bounds

Kuan Liu, Aurélien Bellet

Escaping the Curse of Dimensionality in Similarity Learning: Efficient Frank-Wolfe Algorithm and Generalization Bounds

Kuan Liu[a,1], Aurélien Bellet[b,*]

[a] *Google Inc., USA*
[b] *INRIA, France*

Abstract

Similarity and metric learning provides a principled approach to construct a task-specific similarity from weakly supervised data. However, these methods are subject to the curse of dimensionality: as the number of features grows large, poor generalization is to be expected and training becomes intractable due to high computational and memory costs. In this paper, we propose a similarity learning method that can efficiently deal with high-dimensional sparse data. This is achieved through a parameterization of similarity functions by convex combinations of sparse rank-one matrices, together with the use of a greedy approximate Frank-Wolfe algorithm which provides an efficient way to control the number of active features. We show that the convergence rate of the algorithm, as well as its time and memory complexity, are independent of the data dimension. We further provide a theoretical justification of our modeling choices through an analysis of the generalization error, which depends logarithmically on the sparsity of the solution rather than on the number of features. Our experiments on datasets with up to one million features demonstrate the ability of our approach to generalize well despite the high dimensionality as well as its superiority compared to several competing methods.

Keywords: Metric learning, Frank-Wolfe algorithm, Generalization bounds

1. Introduction

High-dimensional and sparse data are commonly encountered in many applications of machine learning, such as computer vision, bioinformatics, text mining and behavioral targeting. To classify, cluster or rank data points, it is

*Corresponding author
Email addresses: liukuan@google.com (Kuan Liu), aurelien.bellet@inria.fr (Aurélien Bellet)

[1] Most of this work was done when the author was affiliated with the Information Sciences Institute, University of Southern California, USA.

important to be able to compute semantically meaningful similarities between them. However, defining an appropriate similarity measure for a given task is often difficult as only a small and unknown subset of all features are actually relevant. For instance, in drug discovery studies, chemical compounds are typically represented by a large number of sparse features describing their 2D and 3D properties, and only a few of them play in role in determining whether the compound will bind to a particular target receptor (Leach and Gillet, 2007). In text classification and clustering, a document is often represented as a sparse bag of words, and only a small subset of the dictionary is generally useful to discriminate between documents about different topics. Another example is targeted advertising, where ads are selected based on fine-grained user history (Chen et al., 2009).

Similarity and metric learning (Bellet et al., 2015) offers principled approaches to construct a task-specific similarity measure by learning it from weakly supervised data, and has been used in many application domains. The main theme in these methods is to learn the parameters of a similarity (or distance) function such that it agrees with task-specific similarity judgments (e.g., of the form "data point $\boldsymbol{x}$ should be more similar to $\boldsymbol{y}$ than to $\boldsymbol{z}$"). To account for correlations between features, similarity and metric learning typically estimates a number of parameters which is quadratic in the data dimension d. When data are high-dimensional, these methods are thus particularly affected by the so-called "curse of dimensionality", which manifests itself at both the algorithmic and generalization levels. On the one hand, training the similarity quickly becomes infeasible due to a quadratic or cubic complexity in d. In fact, the $O(d^2)$ parameters may not even fit in memory. On the other hand, putting aside the training phase, learning so many parameters would lead to severe overfitting and poor generalization performance (especially for sparse data where some features are rarely observed). Simple workarounds have been used to address this limitation, such as projecting the data into a low-dimensional space before learning the similarity (see e.g. Davis et al., 2007; Weinberger and Saul, 2009; Guillaumin et al., 2009). However, such heuristics do not provide satisfactory solutions: they often hurt the performance and make the resulting similarity function difficult to interpret.

In this paper, we propose a novel method to learn a bilinear similarity function $S_{\boldsymbol{M}}(\boldsymbol{x}, \boldsymbol{x}') = \boldsymbol{x}^{\mathrm{T}} \boldsymbol{M} \boldsymbol{x}'$ directly in the original high-dimensional space while escaping the curse of dimensionality. This is achieved by combining three ingredients: the sparsity of the data, the parameterization of $\boldsymbol{M}$ as a convex combination of rank-one matrices with a special sparsity structure, and an approximate Frank-Wolfe procedure (Frank and Wolfe, 1956; Jaggi, 2013) to learn the similarity parameters. The resulting algorithm greedily incorporates one pair of features at a time into the learned similarity, providing an efficient way to filter out irrelevant features as well as to guard against overfitting through early stopping. Remarkably, the convergence rate of the algorithm as well as its time and memory complexity are all independent of the dimension d. The resulting similarity functions are extremely sparse, which makes them fast to compute and easier to interpret.

We provide strong theoretical and empirical evidence of the usefulness of our approach. On the theory part, we perform a generalization analysis of the solution returned by our algorithm after a given number of iterations. We derive excess risk bounds with respect to the minimizer of the expected risk which confirm that our modeling choices as well as our Frank-Wolfe algorithm and early stopping policy provide effective ways to avoid overfitting in high dimensions. A distinctive feature of the generalization bound we obtain is the adaptivity of its model class complexity term to the actual sparsity of the approximate solution found by our algorithm, again removing the dependence on the dimension d. We also evaluate the proposed approach on several synthetic and real datasets with up to one million features, some of which have a large proportion of irrelevant features. To the best of our knowledge, it is the first time that a full similarity or distance metric is learned directly on such high-dimensional datasets without first reducing dimensionality. Our experiments show that our approach is able to generalize well despite the high dimensionality, and even to recover the ground truth similarity function when the training similarity judgments are sufficiently informative. Furthermore, our approach clearly outperforms both a diagonal similarity learned in the original space and a full similarity learned in a reduced space (after PCA or random projections). Finally, we show that our similarity functions can be extremely sparse (in the order of 0.0001% of nonzero entries), thereby drastically reducing the dimension while also providing an opportunity to analyze the importance of the original features and their pairwise interactions for the problem at hand.

The present work extends a previously published conference paper (Liu et al., 2015a) by providing additional technical and experimental results. Firstly, we present a novel generalization analysis which further backs up our approach from a statistical learning point of view. Secondly, we conduct experiments on high-dimensional synthetic data showing that our approach generalizes well as the dimensionality increases and can even accurately recover the ground truth notion of similarity. Finally, we extend the discussion of the related work and provide additional details on algorithms and proofs.

The paper is organized as follows. Section 2 introduces some background and related work on similarity learning and Frank-Wolfe algorithms. Section 3 describes our problem formulation, the proposed algorithm and its analysis. Generalization bounds are established in Section 4. Finally, Section 5 describes our experimental results, and we conclude in Section 6.

2. Background and Related Work

In this section, we review some background and related work in metric and similarity learning (Section 2.1) and the Frank-Wolfe algorithm (Section 2.2).

2.1. Metric and Similarity Learning

Metric and similarity learning has attracted a lot of interest over the past ten years. The great majority of work has focused on learning either a Mahalanobis

distance $d_{\boldsymbol{M}}(\boldsymbol{x}, \boldsymbol{x}') = (\boldsymbol{x} - \boldsymbol{x}')^{\mathrm{T}}\boldsymbol{M}(\boldsymbol{x} - \boldsymbol{x}')$ where $\boldsymbol{M}$ is a symmetric positive semi-definite (PSD) matrix, or a bilinear similarity $S_{\boldsymbol{M}}(\boldsymbol{x}, \boldsymbol{x}') = \boldsymbol{x}^{\mathrm{T}}\boldsymbol{M}\boldsymbol{x}'$ where $\boldsymbol{M}$ is often taken to be an arbitrary $d \times d$ matrix. A comprehensive survey of existing approaches can be found in (Bellet et al., 2013). We focus below on the two topics most relevant to our work: (i) efficient algorithms for the high-dimensional setting, and (ii) the derivation of generalization guarantees for metric and similarity learning.

Metric learning in high dimensions. Both Mahalanobis distance metric learning and bilinear similarity learning require estimating $O(d^2)$ parameters, which is undesirable in the high-dimensional setting for the reasons mentioned earlier. In practice, it is thus customary to resort to dimensionality reduction (such as PCA, SVD or random projections) to preprocess the data when it has more than a few hundred dimensions (see e.g., Davis et al., 2007; Weinberger and Saul, 2009; Guillaumin et al., 2009; Ying and Li, 2012; Wang et al., 2012; Lim et al., 2013; Qian et al., 2014; Liu et al., 2015b; Yao et al., 2018). Although this strategy can be justified formally in some cases (Liu et al., 2015b; Qian et al., 2015), the projection may intertwine useful features and irrelevant/noisy ones and thus hurt the performance of the resulting similarity function. It also makes it hard to interpret and use for data exploration, preventing the discovery of knowledge that can be valuable to domain experts.

There have been very few satisfactory solutions to this essential limitation. The most drastic strategy is to learn a diagonal matrix $\boldsymbol{M}$ (Schultz and Joachims, 2003; Gao et al., 2014), which is very restrictive as it amounts to a simple weighting of the features. Instead, some approaches assume an explicit low-rank decomposition $\boldsymbol{M} = \boldsymbol{L}^{\mathrm{T}}\boldsymbol{L}$ and learn $\boldsymbol{L} \in \mathbb{R}^{r \times d}$ in order to reduce the number of parameters (Goldberger et al., 2004; Weinberger and Saul, 2009; Kedem et al., 2012). This results in nonconvex formulations with many local optima (Kulis, 2012), and requires to tune r carefully. Moreover, the training complexity still depends on d and can thus remain quite large. Another direction is to learn $\boldsymbol{M}$ as a combination of rank-one matrices. In particular, Shi et al. (2014) generate a set of rank-one matrices from the training data and then learn a metric as a sparse combination. However, as the dimension increases, a larger dictionary is needed and can be expensive to generate. Some other work has studied sparse and/or low-rank regularization to reduce overfitting in high dimensions (Rosales and Fung, 2006; Qi et al., 2009; Ying et al., 2009) but this does not in itself reduce the training complexity of the algorithm. Zhang and Zhang (2017) proposed a stochastic gradient descent solver together with low-rank regularization in an attempt to keep the intermediate solutions low-rank. The complexity per iteration of their approach is linear in d but cubic in the rank of the current solution, which quickly becomes intractable unless the regularization is very strong.

Finally, some greedy algorithms for metric learning have been proposed in the literature to guarantee a tighter bound on the rank of intermediate solutions. Atzmon et al. (2015) use a block coordinate descent algorithm to update the metric one feature at a time. Shen et al. (2012) selects rank-one updates in a

boosting manner, while DML-eig (Ying and Li, 2012) and its extension DML-ρ (Cao et al., 2012b) rely on a greedy Frank-Wolfe algorithm to optimize over the set of PSD matrices with unit trace. However, these greedy methods still suffer from a computational cost of $O(d^2)$ per iteration and are thus unsuitable for the high-dimensional setting we consider in this work. In contrast, we will propose an algorithm which is *linear in the number of nonzero features* and can thus be efficiently applied to high-dimensional sparse data.

Generalization bounds for metric learning. The derivation of generalization guarantees for metric and similarity learning has been investigated in the supervised setting, where the metric or similarity is learned from a labeled dataset of n points by (regularized) empirical risk minimization. For a given family of loss functions, the results generally bound the maximal deviation between the expected risk (where the expectation is taken over the unknown data distribution) and the empirical risk of the learned metric.[2] These bounds are generally of order $O(1/\sqrt{n})$.

Several technical tools have been used to address the challenge of learning from dependent pairs/triplets, leading to different trade-offs in terms of tightness, generality, and dependence on the feature dimension d. The results of Jin et al. (2009) apply only under Frobenius norm regularization of $\boldsymbol{M}$ and have a $\sqrt{d}$ factor in the rate. Using an adaptation of algorithmic robustness, Bellet and Habrard (2015) obtain bounds which hold also for sparsity-inducing regularizers but with a covering number term that can be exponential in the dimension. Bian and Tao (2011) rely on assumptions on the data distribution and do not show an explicit dependence on the dimension. Cao et al. (2012a) derive bounds based on Rademacher complexity and maximal deviation results for U-statistics (Clémençon et al., 2008). Depending on the regularization used, the dependence on the dimension d ranges from logarithmic to linear. Verma and Branson (2015) show that the $\sqrt{d}$ factor of Jin et al. (2009) is in fact unavoidable in the worst case without some form of regularization (or restriction of the hypothesis class). They derive bounds which do not depend on the dimension d but on the Frobenius norm of the optimal parameter $\boldsymbol{M}$. Note however that their analysis assumes that the metrics are learned from a set of i.i.d. pairs or triplets, which is rarely seen in practice.

In all the above work, generalization in metric learning is studied independently of the algorithm used to solve the empirical risk minimization problem, and none of the bounds are adaptive to the actual sparsity of the solution. In contrast, we will show that one can use early stopping in our algorithm to control the complexity of the hypothesis class so as to make the bounds independent of the dimension d, effectively balancing the empirical (optimization) error and the generalization error.

[2] This is in contrast to a different line of work, inspired by the problem of ordinal embedding, which aims to learn a metric which correctly orders a *fixed set of known points* (see for instance Jain et al., 2017)

Algorithm 1 Standard Frank-Wolfe algorithm

Input: Initial point $\boldsymbol{M}^{(0)} \in \mathcal{D}$
for $k = 0, 1, 2, \ldots$ **do**
  $\boldsymbol{S}^{(k)} \leftarrow \arg\min_{\boldsymbol{S} \in \mathcal{D}} \langle \boldsymbol{S}, \nabla f(\boldsymbol{M}^{(k)}) \rangle$
  $\gamma^{(k)} \leftarrow \frac{2}{k+2}$ (or determined by line search)
  $\boldsymbol{M}^{(k+1)} \leftarrow (1 - \gamma^{(k)}) \boldsymbol{M}^{(k)} + \gamma^{(k)} \boldsymbol{S}^{(k)}$
end for

2.2. Frank-Wolfe Algorithms

The Frank-Wolfe (FW) algorithm was originally introduced by Frank and Wolfe (1956) and further generalized by Clarkson (2010) and Jaggi (2013). FW aims at solving constrained optimization problems of the following general form:

$$\min_{\boldsymbol{M} \in \mathcal{D}} f(\boldsymbol{M}), \tag{1}$$

where f is a convex and continuously differentiable function, and the feasible domain $\mathcal{D}$ is a convex and compact subset of some Hilbert space equipped with inner product $\langle \cdot, \cdot \rangle$.

Starting from a feasible initial point $\boldsymbol{M}^{(0)} \in \mathcal{D}$, the standard FW algorithm iterates over the following steps. First, it finds the feasible point $\boldsymbol{S}^{(k)} \in \mathcal{D}$ which minimizes the linearization of f at the current point $\boldsymbol{M}^{(k)}$:

$$\boldsymbol{S}^{(k)} \in \arg\min_{\boldsymbol{S} \in \mathcal{D}} \ \langle \boldsymbol{S}, \nabla f(\boldsymbol{M}^{(k)}) \rangle. \tag{2}$$

The next iterate $\boldsymbol{M}^{(k+1)}$ is then constructed as a convex combination of $\boldsymbol{M}^{(k)}$ and $\boldsymbol{S}^{(k)}$, where the relative weight of each component is given by a step size $\gamma^{(k)}$. The step size can be decreasing with the iteration number k or set by line search. The overall algorithm is summarized in Algorithm 1. FW is guaranteed to converge to an optimal solution of (1) at rate $O(1/k)$, see for instance (Jaggi, 2013) for a generic and concise proof.

Unlike projected gradient, FW is a *projection-free* algorithm: each iterate $\boldsymbol{M}^{(k)}$ is feasible by construction since it is a convex combination of elements of $\mathcal{D}$. Instead of computing projections onto the feasible domain $\mathcal{D}$, FW solves the linear optimization subproblem (2). The linearity of the objective (2) implies that a solution $\boldsymbol{S}^{(k)}$ always lies at an extremal point of $\mathcal{D}$. This leads to the interpretation of FW as a greedy algorithm which adds an extremal point to the current solution at each iteration (Clarkson, 2010). In other words, $\boldsymbol{M}^{(k)}$ can be written as a sparse convex combination of extremal points:

$$\boldsymbol{M}^{(k)} = \sum_{\boldsymbol{S}^{(k)} \in \mathcal{S}^{(k)}} \alpha^{(k)}_{\boldsymbol{S}^{(k)}} \boldsymbol{S}^{(k)}, \qquad \text{where} \sum_{\boldsymbol{S}^{(k)} \in \mathcal{S}^{(k)}} \alpha^{(k)}_{\boldsymbol{S}^{(k)}} = 1 \text{ and } \alpha^{(k)}_{\boldsymbol{S}^{(k)}} \geq 0, \tag{3}$$

where $\mathcal{S}^{(k)}$ denotes the set of "active" extremal points that have been added up to iteration k. When the extremal points of $\mathcal{D}$ have specific structure (such as sparsity, or low-rankness), this structure can be leveraged to compute a solution

Algorithm 2 Frank-Wolfe algorithm with away steps

Input: Initial point $\boldsymbol{M}^{(0)} \in \mathcal{D}$
for $k = 0, 1, 2, \ldots$ **do**
  $\boldsymbol{S}_F^{(k)} \leftarrow \arg\min_{\boldsymbol{S} \in \mathcal{D}} \langle \boldsymbol{S}, \nabla f(\boldsymbol{M}^{(k)}) \rangle$, $\boldsymbol{D}_F^{(k)} = \boldsymbol{S}_F^{(k)} - \boldsymbol{M}^{(k)}$ *// forward direction*
  $\boldsymbol{S}_A^{(k)} \leftarrow \arg\max_{\boldsymbol{S} \in \mathcal{S}^{(k)}} \langle \boldsymbol{S}, \nabla f(\boldsymbol{M}^{(k)}) \rangle$, $\boldsymbol{D}_A^{(k)} = \boldsymbol{M}^{(k)} - \boldsymbol{S}_A^{(k)}$ *// away direction*
  if $\langle \boldsymbol{D}_F^{(k)}, \nabla f(\boldsymbol{M}^{(k)}) \rangle \leq \langle \boldsymbol{D}_A^{(k)}, \nabla f(\boldsymbol{M}^{(k)}) \rangle$ **then**
    $\boldsymbol{D}^{(k)} \leftarrow \boldsymbol{D}_F^{(k)}$ and $\gamma_{\max} \leftarrow 1$ *// choose forward step*
  else
    $\boldsymbol{D}^{(k)} \leftarrow \boldsymbol{D}_A^{(k)}$ and $\gamma_{\max} \leftarrow \alpha^{(k)}_{\boldsymbol{S}_A^{(k)}} / (1 - \alpha^{(k)}_{\boldsymbol{S}_A^{(k)}})$ *// choose away step*
  end if
  $\gamma^{(k)} \leftarrow \frac{2}{k+2}$ (or determined by line search)
  $\boldsymbol{M}^{(k+1)} \leftarrow \boldsymbol{M}^{(k)} + \gamma^{(k)} \boldsymbol{D}^{(k)}$
end for

of (2) much more efficiently than the projection operator, see Jaggi (2011, 2013) for compelling examples.

A drawback of the standard FW algorithm is that "removing" an extremal point $\boldsymbol{S}^{(k)}$ from the current iterate (or significantly reducing its weight $\alpha^{(k)}_{\boldsymbol{S}^{(k)}}$) can only be done indirectly by adding (increasing the weight of) other extremal points. The variant of FW with *away steps* (Guélat and Marcotte, 1986) addresses this issue by allowing the algorithm to choose between adding a new extremal point (forward step) or reducing the weight of an existing one (away step), as shown in Algorithm 2. This can lead to sparser solutions (Guélat and Marcotte, 1986; Clarkson, 2010; Jaggi, 2011) and faster convergence in some cases (Guélat and Marcotte, 1986; Lacoste-Julien and Jaggi, 2015).

In the present work, we will introduce a FW algorithm with away steps to efficiently perform similarity learning for high-dimensional sparse data. One of our key ingredients will be the design of a feasible domain with appropriate sparsity structure.

3. Proposed Approach

This section introduces HDSL (High-Dimensional Similarity Learning), the approach proposed in this paper. We first describe our problem formulation (Section 3.1), then derive and analyze an efficient FW algorithm to solve it in Section 3.2.

3.1. Problem Formulation

In this work, our goal is to learn a similarity function for high-dimensional sparse data. We assume the data points lie in some space $\mathcal{X} \subseteq \mathbb{R}^d$, where d is large ($d > 10^4$) but points are s-sparse on average ($s \ll d$). In other words,

their number of nonzero entries is typically much smaller than d. We focus on learning a similarity function $S_{\boldsymbol{M}} : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ of the form

$$S_{\boldsymbol{M}}(\boldsymbol{x}, \boldsymbol{x}') = \boldsymbol{x}^{\mathrm{T}} \boldsymbol{M} \boldsymbol{x}' = \langle \boldsymbol{x}\boldsymbol{x}'^{\top}, \boldsymbol{M} \rangle,$$

where $\boldsymbol{M} \in \mathbb{R}^{d \times d}$ and $\langle \cdot, \cdot \rangle$ denotes the Frobenius inner product. Notice that for any $\boldsymbol{M}$, $S_{\boldsymbol{M}}$ can be computed in $O(s^2)$ time on average if data points are stored in a sparse format.

Feasible domain. We will derive an algorithm to learn a very sparse $\boldsymbol{M}$ with time and memory requirements that depend on s but not on d. To this end, given a scale $\lambda > 0$ which will play the role of a regularization parameter, we parameterize $\boldsymbol{M}$ as a convex combination of rank-one, 4-sparse $d \times d$ bases:

$$\boldsymbol{M} \in \mathcal{D}_\lambda = \mathrm{conv}(\mathcal{B}_\lambda), \qquad \text{with } \mathcal{B}_\lambda = \bigcup_{ij} \left\{ \boldsymbol{P}_\lambda^{(ij)}, \boldsymbol{N}_\lambda^{(ij)} \right\},$$

where for any pair of features $i, j \in \{1, \ldots, d\}$, $i \neq j$,

$$\boldsymbol{P}_\lambda^{(ij)} = \lambda(\boldsymbol{e}_i + \boldsymbol{e}_j)(\boldsymbol{e}_i + \boldsymbol{e}_j)^T = \begin{pmatrix} \cdot & \cdot & \cdot & \cdot & \cdot \\ \cdot & \lambda & \cdot & \lambda & \cdot \\ \cdot & \cdot & \cdot & \cdot & \cdot \\ \cdot & \lambda & \cdot & \lambda & \cdot \\ \cdot & \cdot & \cdot & \cdot & \cdot \end{pmatrix},$$

$$\boldsymbol{N}_\lambda^{(ij)} = \lambda(\boldsymbol{e}_i - \boldsymbol{e}_j)(\boldsymbol{e}_i - \boldsymbol{e}_j)^T = \begin{pmatrix} \cdot & \cdot & \cdot & \cdot & \cdot \\ \cdot & \lambda & \cdot & -\lambda & \cdot \\ \cdot & \cdot & \cdot & \cdot & \cdot \\ \cdot & -\lambda & \cdot & \lambda & \cdot \\ \cdot & \cdot & \cdot & \cdot & \cdot \end{pmatrix}.$$

The use of such sparse matrices was first suggested by Jaggi (2011). Besides the fact that they are instrumental to the efficiency of our algorithm (see Section 3.2), we give some additional motivation for their use in the context of similarity learning.

First, any $\boldsymbol{M} \in \mathcal{D}_\lambda$ is a convex combination of symmetric PSD matrices and is thus also symmetric PSD. Unlike many metric learning algorithms, we thus avoid the $O(d^3)$ cost of projecting onto the PSD cone. Constraining $\boldsymbol{M}$ to be symmetric PSD provides useful regularization to prevent overfitting (Chechik et al., 2009) and ensures that $S_{\boldsymbol{M}}$ can be interpreted as a dot product after a linear transformation of the inputs:

$$S_{\boldsymbol{M}}(\boldsymbol{x}, \boldsymbol{x}') = \boldsymbol{x}^{\mathrm{T}} \boldsymbol{M} \boldsymbol{x}' = (\boldsymbol{L}\boldsymbol{x})^{\mathrm{T}} (\boldsymbol{L}\boldsymbol{x}'),$$

where $\boldsymbol{M} = \boldsymbol{L}\boldsymbol{L}^{\mathrm{T}}$ with $\boldsymbol{L} \in \mathbb{R}^{d \times k}$. Because the bases in $\mathcal{B}_\lambda$ are rank-one, the dimensionality k of the transformed space is at most the number of bases composing $\boldsymbol{M}$.

Second, each basis operates on two features only. In particular, $S_{\boldsymbol{P}_\lambda^{(ij)}}(\boldsymbol{x}, \boldsymbol{x}') = \lambda(x_i x'_i + x_j x'_j + x_i x'_j + x_j x'_i)$ assigns a higher similarity score when feature i appears jointly in $\boldsymbol{x}$ and $\boldsymbol{x}'$ (likewise for j), as well as when feature i in $\boldsymbol{x}$ and feature j in $\boldsymbol{y}$ co-occur (and vice versa). Conversely, $S_{\boldsymbol{N}_\lambda^{(ij)}}$ penalizes the cross-occurrences of features i and j. In the context of text data represented as

bags-of-words (or other count data), the semantic behind the bases in $\mathcal{B}_\lambda$ is quite natural: they can be intuitively thought of as encoding the fact that a term i or j present in both documents makes them more similar, and that two terms i and j are associated with the same/different class or topic.

Optimizing over the convex hull $\mathcal{D}_\lambda$ of $\mathcal{B}_\lambda$ will allow us to easily control the number of active features, thereby learning a very compact representation with efficient similarity computations.

Optimization problem. We now describe the optimization problem to learn the similarity parameters. Following previous work (see for instance Schultz and Joachims, 2003; Weinberger and Saul, 2009; Chechik et al., 2009), our training data consists of weak supervision in the form of triplet constraints:

$$\mathcal{T} = \{\boldsymbol{x}_t \text{ should be more similar to } \boldsymbol{y}_t \text{ than to } \boldsymbol{z}_t\}_{t=1}^T .$$

Such constraints can be built from a labeled training sample (see Section 4), provided directly by domain experts or crowdsourcing campaign, or obtained through implicit feedback such as clicks on search engine results. For notational convenience, we denote $\boldsymbol{A}^t = \boldsymbol{x}_t(\boldsymbol{y}_t - \boldsymbol{z}_t)^{\mathrm{T}} \in \mathbb{R}^{d\times d}$ for each constraint $t = 1, \ldots, T$ so that we can concisely write $S_{\boldsymbol{M}}(\boldsymbol{x}_t, \boldsymbol{y}_t) - S_{\boldsymbol{M}}(\boldsymbol{x}_t, \boldsymbol{z}_t) = \langle \boldsymbol{A}^t, \boldsymbol{M}\rangle$. We measure the degree of violation of each constraint t with the smoothed hinge loss $\ell : \mathbb{R} \to \mathbb{R}^+$ defined as

$$\ell\left(\langle \boldsymbol{A}^t, \boldsymbol{M}\rangle\right) = \begin{cases} 0 & \text{if } \langle \boldsymbol{A}^t, \boldsymbol{M}\rangle \geq 1 \\ \frac{1}{2} - \langle \boldsymbol{A}^t, \boldsymbol{M}\rangle & \text{if } \langle \boldsymbol{A}^t, \boldsymbol{M}\rangle \leq 0 \\ \frac{1}{2}\left(1 - \langle \boldsymbol{A}^t, \boldsymbol{M}\rangle\right)^2 & \text{otherwise} \end{cases} .$$

This convex loss is a continuously differentiable version of the standard hinge loss which tries to enforce a margin constraint of the form $S_{\boldsymbol{M}}(\boldsymbol{x}_t, \boldsymbol{y}_t) \geq S_{\boldsymbol{M}}(\boldsymbol{x}_t, \boldsymbol{z}_t) + 1$. When this constraint is satisfied, the value of the loss is zero. On the other hand, when the margin is negative, i.e. $S_{\boldsymbol{M}}(\boldsymbol{x}_t, \boldsymbol{y}_t) \leq S_{\boldsymbol{M}}(\boldsymbol{x}_t, \boldsymbol{z}_t)$, the penalty is linear in the margin violation. A quadratic interpolation is used to bridge between these two cases to ensure that the loss is differentiable everywhere.

Remark 1 (Choice of loss)**.** *One may use any other convex and continuously differentiable loss function in our framework, such as the squared hinge loss, the logistic loss or the exponential loss.*

Given $\lambda > 0$, our similarity learning formulation aims at finding the matrix $\boldsymbol{M} \in \mathcal{D}_\lambda$ that minimizes the average margin penalty (as measured by ℓ) over the triplet constraints in $\mathcal{T}$:

$$\min_{\boldsymbol{M}\in\mathbb{R}^{d\times d}} \quad f(\boldsymbol{M}) = \frac{1}{T}\sum_{t=1}^{T} \ell\left(\langle \boldsymbol{A}^t, \boldsymbol{M}\rangle\right) \quad \text{s.t.} \quad \boldsymbol{M} \in \mathcal{D}_\lambda. \tag{4}$$

Due to the convexity of the smoothed hinge loss, (4) involves minimizing a convex function over the convex domain $\mathcal{D}_\lambda$. Note that the gradient of the

Algorithm 3 Frank Wolfe algorithm for problem (4)

initialize $\boldsymbol{M}^{(0)}$ to an arbitrary $\boldsymbol{B} \in \mathcal{B}_\lambda$
for $k = 0, 1, 2, \ldots$ **do**
  $\boldsymbol{B}_F^{(k)} \leftarrow \arg\min_{\boldsymbol{B} \in \mathcal{B}_\lambda} \langle \boldsymbol{B}, \nabla f(\boldsymbol{M}^{(k)}) \rangle$, $\boldsymbol{D}_F^{(k)} \leftarrow \boldsymbol{B}_F^{(k)} - \boldsymbol{M}^{(k)}$ *// forward dir.*
  $\boldsymbol{B}_A^{(k)} \leftarrow \arg\max_{\boldsymbol{B} \in \mathcal{S}^{(k)}} \langle \boldsymbol{B}, \nabla f(\boldsymbol{M}^{(k)}) \rangle$, $\boldsymbol{D}_A^{(k)} \leftarrow \boldsymbol{M}^{(k)} - \boldsymbol{B}_A^{(k)}$ *// away dir.*
  if $\langle \boldsymbol{D}_F^{(k)}, \nabla f(\boldsymbol{M}^{(k)}) \rangle \leq \langle \boldsymbol{D}_A^{(k)}, \nabla f(\boldsymbol{M}^{(k)}) \rangle$ **then**
    $\boldsymbol{D}^{(k)} \leftarrow \boldsymbol{D}_F^{(k)}$ and $\gamma_{\max} \leftarrow 1$ *// choose forward step*
  else
    $\boldsymbol{D}^{(k)} \leftarrow \boldsymbol{D}_A^{(k)}$ and $\gamma_{\max} \leftarrow \alpha_{\boldsymbol{B}_A^{(k)}}^{(k)} / (1 - \alpha_{\boldsymbol{B}_A^{(k)}}^{(k)})$ *// choose away step*
  end if
  $\gamma^{(k)} \leftarrow \arg\min_{\gamma \in [0, \gamma_{\max}]} f(\boldsymbol{M}^{(k)} + \gamma \boldsymbol{D}^{(k)})$ *// perform line search*
  $\boldsymbol{M}^{(k+1)} \leftarrow \boldsymbol{M}^{(k)} + \gamma^{(k)} \boldsymbol{D}^{(k)}$ *// update iterate towards direction*
end for

objective is given by

$$\nabla f(\boldsymbol{M}) = \frac{1}{T} \sum_{t=1}^{T} \boldsymbol{G}^t(\boldsymbol{M}),$$
$$\text{with } \boldsymbol{G}^t(\boldsymbol{M}) = \begin{cases} \boldsymbol{0} & \text{if } \langle \boldsymbol{A}^t, \boldsymbol{M} \rangle \geq 1 \\ -\boldsymbol{A}^t & \text{if } \langle \boldsymbol{A}^t, \boldsymbol{M} \rangle \leq 0 \\ (\langle \boldsymbol{A}^t, \boldsymbol{M} \rangle - 1)\, \boldsymbol{A}^t & \text{otherwise} \end{cases}. \tag{5}$$

In the next section, we propose a greedy algorithm to efficiently find sparse approximate solutions to this problem.

3.2. Algorithm

3.2.1. Exact Frank-Wolfe Algorithm

We propose to use a Frank-Wolfe algorithm with away steps (see Section 2.2) to learn the similarity. We will exploit the fact that in our formulation (4), the extremal points (vertices) of the feasible domain $\mathcal{D}_\lambda$ are the elements of $\mathcal{B}_\lambda$ and have special structure. Our algorithm is shown in Algorithm 3. During the course of the algorithm, we explicitly maintain a representation of each iterate $\boldsymbol{M}^{(k)}$ as a convex combination of basis elements as previously discussed in Section 2.2:

$$\boldsymbol{M}^{(k)} = \sum_{\boldsymbol{B} \in \mathcal{B}_\lambda} \alpha_{\boldsymbol{B}}^{(k)} \boldsymbol{B}, \qquad \text{where } \sum_{\boldsymbol{B} \in \mathcal{B}_\lambda} \alpha_{\boldsymbol{B}}^{(k)} = 1 \text{ and } \alpha_{\boldsymbol{B}}^{(k)} \geq 0.$$

We denote the set of active basis elements in $\boldsymbol{M}^{(k)}$ as $\mathcal{S}^{(k)} = \{\boldsymbol{B} \in \mathcal{B}_\lambda : \alpha_{\boldsymbol{B}}^{(k)} > 0\}$. The algorithm goes as follows. We initialize $\boldsymbol{M}^{(0)}$ to a random basis element. Then, at each iteration, we greedily choose between moving towards a (possibly) new basis (forward step) or reducing the weight of an active one (away step). The extent of the step is determined by line search. As a result,

Algorithm 3 adds only one basis (at most 2 new features) at each iteration, which provides a convenient way to control the number of active features and maintains a compact representation of $\boldsymbol{M}^{(k)}$ for a memory cost of $O(k)$. Furthermore, away steps provide a way to reduce the importance of a potentially "bad" basis element added at an earlier iteration (or even remove it completely when $\gamma^{(k)} = \gamma_{\max}$). Recall that throughout the execution of the FW algorithm, all iterates $\boldsymbol{M}^{(k)}$ remain convex combinations of basis elements and are thus feasible. The following proposition shows that the iterates of Algorithm 3 converge to an optimal solution of (4) with a rate of $O(1/k)$.

Proposition 1. *Let $\lambda > 0$, $\boldsymbol{M}^*$ be an optimal solution to (4) and $L = \frac{1}{T}\sum_{t=1}^{T} \|\boldsymbol{A}^t\|_F^2$. At any iteration $k \geq 1$ of Algorithm 3, the iterate $\boldsymbol{M}^{(k)} \in \mathcal{D}_\lambda$ satisfies $f(\boldsymbol{M}^{(k)}) - f(\boldsymbol{M}^*) \leq 16L\lambda^2/(k+2)$. Furthermore, it has at most rank $k+1$ with $4(k+1)$ nonzero entries, and uses at most $2(k+1)$ distinct features.*

Proof. We first show that ∇f is L-Lipschitz continuous on $\mathcal{D}_\lambda$ with respect to the Frobenius norm, i.e. for any $\boldsymbol{M}_1, \boldsymbol{M}_2 \in \mathcal{D}_\lambda$,

$$\|\nabla f(\boldsymbol{M}_1) - \nabla f(\boldsymbol{M}_2)\|_F \leq L\|\boldsymbol{M}_1 - \boldsymbol{M}_2\|_F \tag{6}$$

for some $L \geq 0$. Note that

$$\begin{aligned}\|\nabla f(\boldsymbol{M}_1) - \nabla f(\boldsymbol{M}_2)\|_F &= \left\| \frac{1}{T}\sum_{t=1}^{T} \boldsymbol{G}^t(\boldsymbol{M}_1) - \frac{1}{T}\sum_{t=1}^{T} \boldsymbol{G}^t(\boldsymbol{M}_2) \right\|_F \\ &\leq \frac{1}{T}\sum_{t=1}^{T} \left\| \boldsymbol{G}^t(\boldsymbol{M}_1) - \boldsymbol{G}^t(\boldsymbol{M}_2) \right\|_F .\end{aligned}$$

Let $t \in \{1, \ldots, T\}$. We will now bound $\Delta_t = \|\boldsymbol{G}^t(\boldsymbol{M}_1) - \boldsymbol{G}^t(\boldsymbol{M}_2)\|_F$ for any $\boldsymbol{M}_1, \boldsymbol{M}_2 \in \mathcal{D}_\lambda$. The form of the gradient (5) requires to consider several cases:

(i) If $\langle \boldsymbol{A}^t, \boldsymbol{M}_1 \rangle \geq 1$ and $\langle \boldsymbol{A}^t, \boldsymbol{M}_2 \rangle \geq 1$, we have $\Delta_t = 0$.

(ii) If $\langle \boldsymbol{A}^t, \boldsymbol{M}_1 \rangle \leq 0$ and $\langle \boldsymbol{A}^t, \boldsymbol{M}_2 \rangle \leq 0$, we have $\Delta_t = 0$.

(iii) If $0 < \langle \boldsymbol{A}^t, \boldsymbol{M}_1 \rangle < 1$ and $0 < \langle \boldsymbol{A}^t, \boldsymbol{M}_2 \rangle < 1$, we have:

$$\begin{aligned}\Delta_t &= \|\langle \boldsymbol{A}^t, \boldsymbol{M}_1 - \boldsymbol{M}_2 \rangle \boldsymbol{A}^t\|_F = \|\boldsymbol{A}^t\|_F |\langle \boldsymbol{A}^t, \boldsymbol{M}_1 - \boldsymbol{M}_2 \rangle| \\ &\leq \|\boldsymbol{A}^t\|_F^2 \|\boldsymbol{M}_1 - \boldsymbol{M}_2\|_F.\end{aligned}$$

(iv) If $\langle \boldsymbol{A}^t, \boldsymbol{M}_1 \rangle \geq 1$ and $\langle \boldsymbol{A}^t, \boldsymbol{M}_2 \rangle \leq 0$, we have

$$\Delta_t = \|\boldsymbol{A}^t\|_F \leq \|\boldsymbol{A}^t\|_F |\langle \boldsymbol{A}^t, \boldsymbol{M}_1 - \boldsymbol{M}_2 \rangle| \leq \|\boldsymbol{A}^t\|_F^2 \|\boldsymbol{M}_1 - \boldsymbol{M}_2\|_F.$$

(v) If $\langle \boldsymbol{A}^t, \boldsymbol{M}_1 \rangle \geq 1$ and $0 < \langle \boldsymbol{A}^t, \boldsymbol{M}_2 \rangle < 1$, we have:

$$\begin{aligned}\Delta_t &= \|(\langle \boldsymbol{A}^t, \boldsymbol{M}_2 \rangle - 1)\boldsymbol{A}^t\|_F = \|\boldsymbol{A}^t\|_F (1 - \langle \boldsymbol{A}^t, \boldsymbol{M}_2 \rangle) \\ &\leq \|\boldsymbol{A}^t\|_F (1 - \langle \boldsymbol{A}^t, \boldsymbol{M}_2 \rangle) + \|\boldsymbol{A}^t\|_F (\langle \boldsymbol{A}^t, \boldsymbol{M}_1 \rangle - 1) \\ &= \|\boldsymbol{A}^t\|_F \langle \boldsymbol{A}^t, \boldsymbol{M}_1 - \boldsymbol{M}_2 \rangle \leq \|\boldsymbol{A}^t\|_F^2 \|\boldsymbol{M}_1 - \boldsymbol{M}_2\|_F.\end{aligned}$$

(vi) If $\langle \boldsymbol{A}^t, \boldsymbol{M}_1 \rangle \leq 0$ and $0 < \langle \boldsymbol{A}^t, \boldsymbol{M}_2 \rangle < 1$, we have:

$$\begin{aligned}\Delta_t &= \| - \boldsymbol{A}^t - (\langle \boldsymbol{A}^t, \boldsymbol{M}_2 \rangle - 1)\boldsymbol{A}^t\|_F = \|\boldsymbol{A}^t \langle \boldsymbol{A}^t, \boldsymbol{M}_2 \rangle\|_F = \|\boldsymbol{A}^t\|_F \langle \boldsymbol{A}^t, \boldsymbol{M}_2 \rangle \\ &\leq \|\boldsymbol{A}^t\|_F \langle \boldsymbol{A}^t, \boldsymbol{M}_2 \rangle - \|\boldsymbol{A}^t\|_F \langle \boldsymbol{A}^t, \boldsymbol{M}_1 \rangle \\ &= \|\boldsymbol{A}^t\|_F \langle \boldsymbol{A}^t, \boldsymbol{M}_2 - \boldsymbol{M}_1 \rangle \leq \|\boldsymbol{A}^t\|_F^2 \|\boldsymbol{M}_1 - \boldsymbol{M}_2\|_F.\end{aligned}$$

The remaining cases are also bounded by $\|\boldsymbol{A}^t\|_F^2 \|\boldsymbol{M}_1 - \boldsymbol{M}_2\|_F$ by symmetry to cases (iv)-(v)-(vi). Hence ∇f is L-Lipschitz continuous with $L = \|\boldsymbol{A}^t\|_F^2$.

It is easy to see that $\operatorname{diam}_{\|\cdot\|_F}(\mathcal{D}_\lambda) = \sqrt{8}\lambda$. The convergence rate then follows from the general analysis of the FW algorithm (Jaggi, 2013).

The second part of the proposition follows directly from the structure of the bases and the greedy nature of the algorithm. □

Note that the optimality gap in Proposition 1 is independent of d. Indeed, $\boldsymbol{A}^t$ has $O(s^2)$ nonzero entries on average, hence the term $\|\boldsymbol{A}^t\|_F^2$ in the Lipschitz constant L can be bounded by $s^2\|\boldsymbol{A}^t\|_\infty$, where $\|\boldsymbol{A}\|_\infty = \max_{i,j=1}^d |A_{i,j}|$. This means that Algorithm 3 is able to find a good approximate solution based on a small number of features in only a few iterations, which is very appealing in the high-dimensional setting we consider.

3.2.2. Complexity Analysis

We now analyze the time and memory complexity of Algorithm 3. The form of the gradient (5) along with the structure of the algorithm's updates are crucial to its efficiency. Since $\boldsymbol{M}^{(k+1)}$ is a convex combination of $\boldsymbol{M}^{(k)}$ and a 4-sparse matrix $\boldsymbol{B}^{(k)}$, we can efficiently compute most of the quantities of interest through careful book-keeping.

In particular, storing $\boldsymbol{M}^{(k)}$ at iteration k requires $O(k)$ memory. We can also recursively compute $\langle \boldsymbol{A}^t, \boldsymbol{M}^{(k+1)} \rangle$ for all constraints in only $O(T)$ time and $O(T)$ memory based on $\langle \boldsymbol{A}^t, \boldsymbol{M}^{(k)} \rangle$ and $\langle \boldsymbol{A}^t, \boldsymbol{B}^{(k)} \rangle$. This allows us, for instance, to efficiently compute the objective value as well as to identify the set of satisfied constraints (those with $\langle \boldsymbol{A}^t, \boldsymbol{M}^{(k)} \rangle \geq 1$) which are ignored in the computation of the gradient. Finding the away direction at iteration k can be done in $O(Tk)$ time. For the line search, we use a bisection algorithm to find a root of the gradient of the 1-dimensional function of γ, which only depends on $\langle \boldsymbol{A}^t, \boldsymbol{M}^{(k)} \rangle$ and $\langle \boldsymbol{A}^t, \boldsymbol{B}^{(k)} \rangle$, both of which are readily available. Its time complexity is $O(T \log \frac{1}{\epsilon})$ where ϵ is the precision of the line-search, with a memory cost of $O(1)$.

The bottleneck is to find the forward direction. Indeed, sequentially considering each basis element is intractable as it takes $O(Td^2)$ time. A more efficient strategy is to sequentially consider each constraint, which requires $O(Ts^2)$ time and $O(Ts^2)$ memory. The overall iteration complexity of Algorithm 3 is given in Table 1.

Variant	Time complexity	Memory complexity
Exact (Algorithm 3)	$\tilde{O}(Ts^2 + Tk)$	$\tilde{O}(Ts^2 + k)$
Mini-batch	$\tilde{O}(Ms^2 + Tk)$	$\tilde{O}(T + Ms^2 + k)$
Mini-batch + heuristic	$\tilde{O}(Ms + Tk)$	$\tilde{O}(T + Ms + k)$

Table 1: Complexity of iteration k (ignoring logarithmic factors) for different variants of the algorithm.

3.2.3. Approximate Forward Step

Finding the forward direction can be expensive when T and s are both large. We propose two strategies to alleviate this cost by finding an approximately optimal basis (see Table 1 for iteration complexity).

Mini-batch approximation. Instead of finding the forward and away directions based on the full gradient at each iteration, we can estimate it on a mini-batch of $M \ll T$ constraints drawn uniformly at random (without replacement). The complexity of finding the forward direction is thus reduced to $O(Ms^2)$ time and $O(Ms^2)$ memory. Consider the deviation between the "value" of any basis element $\boldsymbol{B} \in \mathcal{B}_\lambda$ on the full set of constraints and its estimation on the mini-batch, namely

$$\Big| \frac{1}{M} \sum_{t \in \mathcal{M}} \langle \boldsymbol{B}, \boldsymbol{G}_t \rangle - \frac{1}{T} \sum_{t=1}^{T} \langle \boldsymbol{B}, \boldsymbol{G}_t \rangle \Big|, \tag{7}$$

where $\mathcal{M}$ is the set of M constraint indices drawn uniformly and without replacement from the set $\{1, \dots, T\}$. Under mild assumptions, concentration bounds such as Hoeffding's inequality for sampling without replacement (Serfling, 1974; Bardenet and Maillard, 2015) can be used to show that the probability of (7) being larger than some constant decreases exponentially fast with M. The FW algorithm is known to be robust to inexact gradients, and convergence guarantees similar to Proposition 1 can be obtained directly from (Jaggi, 2013; Freund and Grigas, 2013).

Fast heuristic. To avoid the quadratic dependence on s, we propose to use the following heuristic to find a good forward basis. We first pick a feature $i \in [d]$ uniformly at random, and solve the linear problem over the restricted set $\bigcup_j \{\boldsymbol{P}_\lambda^{(ij)}, \boldsymbol{N}_\lambda^{(ij)}\}$. We then solve it again over the set $\bigcup_k \{\boldsymbol{P}_\lambda^{(kj)}, \boldsymbol{N}_\lambda^{(kj)}\}$ and use the resulting basis for the forward direction. This can be done in only $O(Ms)$ time and $O(Ms)$ memory and gives good performance in practice, as we shall see in Section 5.

4. Generalization Analysis

In this section, we derive generalization bounds for the proposed method. Our main goal is to give a theoretical justification of our approach, in particular

by (i) showing that our choice of feasible domain $\mathcal{D}_\lambda$ helps to reduce overfitting in high dimensions, and (ii) showing that the proposed greedy Frank-Wolfe algorithm provides a simple way to balance between optimization and generalization errors through early stopping.

4.1. Setup and Notations

As in previous work on generalization bounds for metric learning, we consider the supervised learning setting where the training sample is a set of labeled points $S = \{\boldsymbol{z}_i = (\boldsymbol{x}_i, y_i)\}_{i=1}^n$ drawn i.i.d. from a probability distribution μ over the space $\mathcal{Z} = \mathcal{X} \times \mathcal{Y}$, where $\mathcal{X} \subseteq \mathbb{R}^d$ and $\mathcal{Y} = \{1, \dots, C\}$ is the label set. We assume that $B_{\mathcal{X}} = \sup_{\boldsymbol{x},\boldsymbol{x}',\boldsymbol{x}'' \in \mathcal{X}} \|\boldsymbol{x}(\boldsymbol{x}' - \boldsymbol{x}'')^T\|$ is bounded for some convenient matrix norm $\|\cdot\|$.

For simplicity, we assume that the univariate loss function $\ell : \mathbb{R} \to \mathbb{R}^+$ is 1-Lipschitz, which is the case for the smoothed hinge loss used in our algorithm. Given a triplet $(\boldsymbol{z}, \boldsymbol{z}', \boldsymbol{z}'') \in \mathcal{Z}^3$, we say that it is *admissible* if $y = y' \neq y''$. Since we only want to consider admissible triplets, we will use the triplet-wise loss function $L_{\boldsymbol{M}}(\boldsymbol{z}, \boldsymbol{z}', \boldsymbol{z}'') = \mathbb{I}[y = y' \neq y''] \cdot \ell(\langle \boldsymbol{x}(\boldsymbol{x}' - \boldsymbol{x}'')^{\mathrm{T}}, \boldsymbol{M} \rangle)$ indexed by $\boldsymbol{M} \in \mathcal{D}_\lambda$, which is equal to zero for non-admissible triplets.

Given a matrix $\boldsymbol{M} \in \mathcal{D}_\lambda$, we define its empirical risk associated on the training set S as follows:

$$\mathcal{L}_S(\boldsymbol{M}) = \frac{1}{n(n-1)(n-2)} \sum_{i \neq j \neq k} L_{\boldsymbol{M}}(\boldsymbol{z}_i, \boldsymbol{z}_j, \boldsymbol{z}_k). \tag{8}$$

Similarly, its expected risk is defined as

$$\mathcal{L}(\boldsymbol{M}) = \mathop{\mathbb{E}}_{\boldsymbol{z},\boldsymbol{z}',\boldsymbol{z}'' \sim \mu} \left[L_{\boldsymbol{M}}(\boldsymbol{z}, \boldsymbol{z}', \boldsymbol{z}'') \right]. \tag{9}$$

In contrast to the standard supervised classification setting, note that the empirical risk (8) takes the form of an *average of dependent terms* known as a U-statistic (Lee, 1990).

From our feasible domain $\mathcal{D}_\lambda = \mathrm{conv}(\mathcal{B}_\lambda)$, we can define a sequence of nested sets as follows:

$$\mathcal{D}_\lambda^{(k)} = \left\{ \sum_{i=1}^k \alpha_i \boldsymbol{B}_i : \boldsymbol{B}_i \in \mathcal{B}_\lambda, \alpha_i \geq 0, \sum_{i=1}^k \alpha_i = 1 \right\}, \quad k = 1, \dots, 2d(d-1). \tag{10}$$

In other words, $\mathcal{D}_\lambda^{(k)}$ consists of all $d \times d$ matrices which can be decomposed as a convex combination of at most k elements of the basis set $\mathcal{B}_\lambda$. Clearly, we have $\mathcal{D}_\lambda^{(1)} \subset \mathcal{D}_\lambda^{(2)} \subset \cdots \subset \mathcal{D}_\lambda^{(2d(d-1))} = \mathcal{D}_\lambda$. Note also that since ℓ is 1-Lipschitz, by Holder's inequality we have $\forall k$:

$$\begin{aligned} \sup_{\boldsymbol{z},\boldsymbol{z}',\boldsymbol{z}'' \in \mathcal{Z}, \boldsymbol{M} \in \mathcal{D}_\lambda^{(k)}} |L_{\boldsymbol{M}}(\boldsymbol{z}, \boldsymbol{z}', \boldsymbol{z}'')| &\leq \sup_{\boldsymbol{x},\boldsymbol{x}',\boldsymbol{x}'' \in \mathcal{X}, \boldsymbol{M} \in \mathcal{D}_\lambda^{(k)}} |\ell(\langle \boldsymbol{x}(\boldsymbol{x}' - \boldsymbol{x}'')^T, \boldsymbol{M} \rangle)| \\ &\leq B_{\mathcal{X}} \sup_{\boldsymbol{M} \in \mathcal{D}_\lambda^{(k)}} \|\boldsymbol{M}\|_*, \end{aligned} \tag{11}$$

where $\|\cdot\|_*$ is the dual norm of $\|\cdot\|$.

In the following, we derive theoretical results that take advantage of the structural properties of our algorithm, namely that the matrix $\boldsymbol{M}^{(k)}$ returned after $k \geq 1$ iterations of Algorithm 3 belongs to $\mathcal{D}_\lambda^{(k)}$. We first bound the Rademacher complexity of $\mathcal{D}_\lambda^{(k)}$ and derive bounds on the maximal deviation between $\mathcal{L}(\boldsymbol{M})$ and $\mathcal{L}_S(\boldsymbol{M})$ for any $\boldsymbol{M} \in \mathcal{D}_\lambda^{(k)}$. We then use these results to derive bounds on the excess risk $\mathcal{L}(\boldsymbol{M}^{(k)}) - \mathcal{L}(\boldsymbol{M}^*)$, where $\boldsymbol{M}^* \in \arg\min_{\boldsymbol{M} \in \mathcal{D}_\lambda} \mathcal{L}(\boldsymbol{M})$ is the expected risk minimizer. All proofs can be found in the appendix.

4.2. Main Results

We first characterize the Rademacher complexity of the loss functions indexed by elements of $\mathcal{D}_\lambda^{(k)}$. Given $k \in \{1, \ldots, 2d(d-1)\}$, consider the family $\mathcal{F}^{(k)} = \{L_{\boldsymbol{M}} : \boldsymbol{M} \in \mathcal{D}_\lambda^{(k)}\}$ of functions mapping from $\mathcal{Z}^3$ to $\mathbb{R}^+$. We will consider the following definition of the Rademacher complexity of $\mathcal{F}^{(k)}$ with respect to distribution μ and sample size $n \geq 3$, adapted from (Clémençon et al., 2008; Cao et al., 2012a):

$$R_n\left(\mathcal{F}^{(k)}\right) = \mathbb{E}_{\boldsymbol{\sigma}, S\sim\mu^n}\left[\sup_{\boldsymbol{M}\in\mathcal{D}_\lambda^{(k)}} \frac{1}{\lfloor n/3 \rfloor} \sum_{i=1}^{\lfloor n/3 \rfloor} \sigma_i L_{\boldsymbol{M}}(\boldsymbol{z}_i, \boldsymbol{z}_{i+\lfloor n/3 \rfloor}, \boldsymbol{z}_{i+2\times\lfloor n/3 \rfloor})\right], \tag{12}$$

where $\boldsymbol{\sigma} = (\sigma_1, \ldots, \sigma_{\lfloor n/3 \rfloor})$ are independent uniform random variables taking values in $\{-1, 1\}$. The following lemma gives a bound on the above Rademacher complexity.

Lemma 1 (Bounded Rademacher complexity)**.** *Let $n \geq 3$, $\lambda > 0$ and $1 \leq k \leq 2d(d-1)$. We have*

$$R_n(\mathcal{F}^{(k)}) \leq 8\lambda B_{\mathcal{X}} \sqrt{\frac{2\log k}{\lfloor n/3 \rfloor}}.$$

Proof. See Appendix B. □

There are two important consequences to Lemma 1. First, restricting the set of feasible matrices $\boldsymbol{M}$ to $\mathcal{D}_\lambda = \mathcal{D}_\lambda^{(2d(d-1))}$ instead of $\mathbb{R}^{d\times d}$ leads to a Rademacher complexity with a very mild $O(\sqrt{\log d})$ dependence in the dimension. This validates our design choice for the feasible domain in the high-dimensional setting we consider. Second, the Rademacher complexity can actually be made independent of d by further restricting the number of bases k.

Using this result, we derive a bound for the deviation between the expected risk $\mathcal{L}(\boldsymbol{M})$ and the empirical risk $\mathcal{L}_S(\boldsymbol{M})$ of any $\boldsymbol{M} \in \mathcal{D}_\lambda^{(k)}$.

Theorem 1 (Maximal deviations)**.** *Let S be a set of of n points drawn i.i.d. from μ, $\lambda > 0$ and $1 \leq k \leq 2d(d-1)$. For any $\delta > 0$, with probability $1 - \delta$ we*

have

$$\sup_{\boldsymbol{M}\in\mathcal{D}_\lambda^{(k)}} [\mathcal{L}(\boldsymbol{M}) - \mathcal{L}_S(\boldsymbol{M})] \leq 16\lambda B_{\mathcal{X}} \sqrt{\frac{2\log k}{\lfloor n/3 \rfloor}} + 3B_{\mathcal{X}} B_{\mathcal{D}_\lambda^{(k)}} \sqrt{\frac{2\ln(2/\delta)}{n}}, \quad (13)$$

where $B_{\mathcal{D}_\lambda^{(k)}} = \sup_{\boldsymbol{M}\in\mathcal{D}_\lambda^{(k)}} \|\boldsymbol{M}\|_*$.

Proof. See Appendix C. □

The generalization bounds given by Theorem 1 exhibit a standard $O(1\sqrt{n})$ rate. They also confirm that restricting the number k of bases is a good strategy to guard against overfitting when the feature dimension d is high. Interestingly, note that due to the convex hull structure of our basis set, $B_{\mathcal{D}_\lambda^{(k)}} = \sup_{\boldsymbol{M}\in\mathcal{D}_\lambda^{(k)}} \|\boldsymbol{M}\|_*$ can be easily bounded by a quantity independent of d for any $k \geq 1$ and any dual norm $\|\cdot\|_*$. We thus have complete freedom to choose the primal norm $\|\cdot\|$ so as to make $B_{\mathcal{X}} = \sup_{\boldsymbol{x},\boldsymbol{x}',\boldsymbol{x}''\in\mathcal{X}} \|\boldsymbol{x}(\boldsymbol{x}'-\boldsymbol{x}'')^T\|$ as small as possible. A good choice of primal norm is the infinity norm $\|\boldsymbol{A}\|_\infty = \max_{i,j=1}^d |A_{i,j}|$, which is independent of d. For instance, if $\mathcal{X} = [0,1]^d$ we have $B_{\mathcal{X}} = 1$. The dual norm of the infinity norm being the L_1 norm, we then have for any $k \geq 1$:

$$B_{\mathcal{D}_\lambda^{(k)}} = \sup_{\boldsymbol{M}\in\mathcal{D}_\lambda^{(k)}} \|\boldsymbol{M}\|_1 = \sup_{\boldsymbol{M}\in\mathcal{D}_\lambda^{(k)}} \sum_{i,j=1}^d |M_{i,j}| \leq 4\lambda. \quad (14)$$

Theorem 1 is directly comparable to the results of Cao et al. (2012a), who derived generalization bounds for similarity learning under various norm regularizers. Their bounds have a similar form, but exhibit a dependence on the feature dimension d which is at least logarithmic (sometimes even linear, depending on the norm used to regularize the empirical risk). In contrast, our bounds depend logarithmically on $k \ll d$. This offers more flexibility in the high-dimensional setting because k can be directly controlled by stopping our algorithm after $k \ll d$ iterations to guarantee that the output is in $\mathcal{D}_\lambda^{(k)}$. This is highlighted by the following corollary, which combines the generalization bounds of Theorem 1 with the $O(1/k)$ convergence rate of our Frank-Wolfe optimization algorithm (Proposition 1).

Corollary 1 (Excess risk bound)**.** *Let S be a set of n points drawn i.i.d. from μ, $\lambda > 0$. Given $k \in \{1,\ldots,2d(d-1)\}$, let $\boldsymbol{M}^{(k)}$ be the solution returned after k iterations of Algorithm 3 applied to the problem $\min_{\boldsymbol{M}\in\mathcal{D}_\lambda} \mathcal{L}_S(\boldsymbol{M})$, and let $\boldsymbol{M}^* \in \arg\min_{\boldsymbol{M}\in\mathcal{D}_\lambda} \mathcal{L}(\boldsymbol{M})$ be the expected risk minimizer over $\mathcal{D}_\lambda$. For any $\delta > 0$, with probability $1-\delta$ we have*

$$\mathcal{L}(\boldsymbol{M}^{(k)}) - \mathcal{L}(\boldsymbol{M}^*) \leq \frac{16L\lambda^2}{k+2} + 16\lambda B_{\mathcal{X}} \sqrt{\frac{2\log k}{\lfloor n/3 \rfloor}} + 5B_{\mathcal{X}} B_{\mathcal{D}_\lambda^{(k)}} \sqrt{\frac{\ln(4/\delta)}{n}}.$$

Proof. See Appendix D. □

Corollary 1 shows that the excess risk with respect to the expected risk minimizer $\boldsymbol{M}^*$ depends on a trade-off between optimization error and complexity of the hypothesis class. Remarkably, this trade-off is ruled by the number k of iterations of the algorithm: as k increases, the optimization error term decreases but the Rademacher complexity terms gets larger. We thus obtain an excess risk bound which adapts to the actual sparsity of the solution output by our algorithm. This is in accordance with our overall goal of reducing overfitting by allowing a strict control on the complexity of the learned similarity, and justifies an early-stopping strategy to achieve a good reduction in empirical risk by selecting the most useful bases while keeping the solution complexity small enough. Again, the excess risk is independent of the feature dimension d, suggesting that in the high-dimensional setting it is possible to find sparse solutions with small excess risk. To the best of our knowledge, this is the first result of this nature for metric or similarity learning.

Remark 2 (Approximation of empirical risk by subsampling)**.** *The empirical risk* (8) *is a sum of* $O(n^3)$ *term, which can be costly to minimize in the large-scale setting. To reduce the computational cost, an alternative to the mini-batch strategy described in Section 3.2.3 is to randomly subsample* M *terms of the sum (e.g., uniformly without replacement) and to solve the resulting approximate empirical risk minimization problem. For general problems involving* U*-statistics, Clémençon et al. (2016) showed that sampling only* $M = O(n)$ *terms is sufficient to maintain the* $O(1/\sqrt{n})$ *rate. These arguments can be adapted to our setting to obtain results similar to Theorem 1 and Corollary 1 for this subsampled empirical risk.*

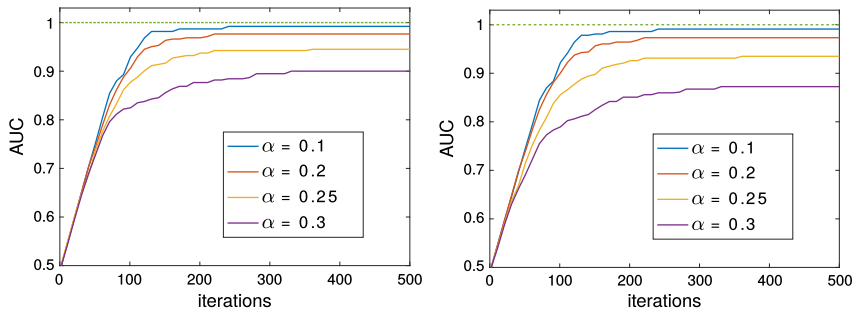
5. Experiments

In this section, we present experiments to evaluate the performance and robustness of HDSL. In Section 5.1, we use synthetic data to study the performance of our approach in terms of similarity recovery and generalization in high dimensions in a controlled environment. Section 5.2 evaluates our algorithm against competing approaches on classification and dimensionality reduction using real-world datasets.

5.1. Experiments on Synthetic Data

We first conduct experiments on synthetic datasets in order to address two questions:

1. Is the algorithm able to recover the ground truth sparse similarity function from (potentially weak) similarity judgments?
2. How well does the algorithm generalize as the dimensionality increases?



(a) Feature recovery AUC (b) Entry recovery AUC

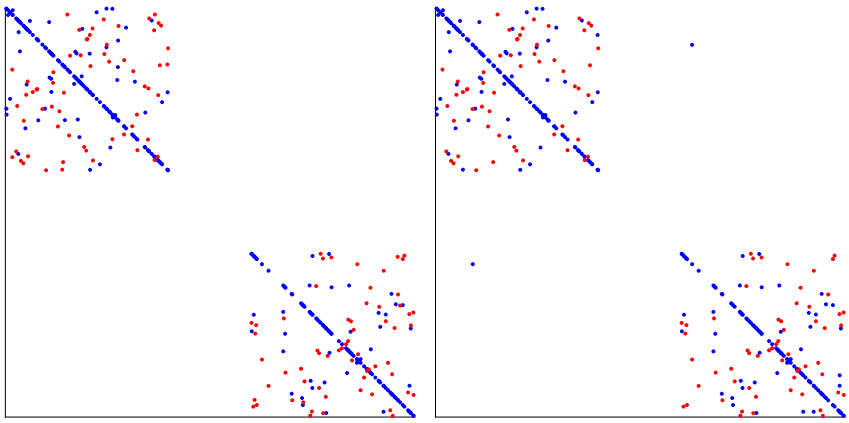
Figure 1: Similarity recovery experiment on synthetic data. Figure 1(a) and Figure 1(b) show the AUC scores (for feature recovery and entry recovery respectively) along the iterations of the algorithm for different values of α.

5.1.1. Similarity Recovery

To investigate the algorithm's ability to recover the underlying similarity, we generate a ground truth similarity metric $M \in \mathbb{R}^{d \times d}$ where $d = 2000$. M is constructed as a convex combination of 100 randomly selected rank-one 4-sparse bases as specified in Section 3.1. The combination coefficients are drawn from a Dirichlet distribution with shape parameter 9 and scale parameter 0.5. Without loss of generality, we choose the metric to be block structured by restricting the basis selection from two blocks. This makes the resulting matrix easier to visualize, as show in Figure 2(a).

We then generate 5000 training samples from the uniform distribution on $[0, 1]$ with 2% sparsity. From this sample, we create $30,000$ training triplets $\{(x_1, x_2, x_3)\}$ where x_1 is randomly picked and x_2 (or x_3) is sampled among x_1's top α% similar (or dissimilar) samples as measured by the ground truth metric M. The parameter α controls the "quality" of the triplet constraints: a larger α leads to less similar (or dissimilar) samples in the triplets, thereby providing a weaker signal about the underlying similarity. We experiment with various α (10%, 20%, 25%, 30%) to investigate the robustness of HDSL to the quality of the supervision. In all our experiments, we use $\lambda = 100$.

Results. We aim to measure how accurately we recover the entries (i.e., pairs of features) that are active in the ground truth similarity as training proceeds. To do so, at each iteration k of HDSL, we rank each pair of features by descending order of the absolute value of the corresponding entry in the current matrix $\boldsymbol{M}^{(k)}$. We then compute the Area under the ROC Curve (AUC) of the ranking induced by the similarity with respect to the list of active entries in the ground truth similarity. The AUC is well-suited to the imbalanced setting (as active entries in the ground truth are a small subset of all entries). It can be interpreted as the probability that a random entry that is active in the ground truth is ranked higher than a random inactive one. Following a similar process, we also compute an AUC score for individual features: this is done by ranking each



(a) True similarity (b) Learned similarity ($\alpha = 20\%$)

Figure 2: Similarity recovery experiment on synthetic data. Figure 2(a) shows the underlying ground truth similarity, where blue dots represent positive entries and red dots represent negative entries (combination coefficients are not displayed). Figure 2(b) shows the similarity learned by HDSL (α =20%), which is visually very close to the ground truth.

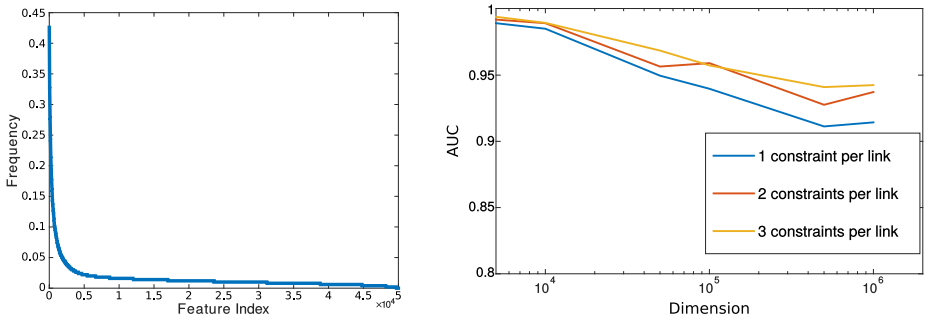
feature by the L_1 norm of its associated row in the matrix.

The AUC scores for feature and entry recovery along the iterations are reported in Figure 1 for different values of α. When the quality of the triplet constraints is high (α =10%,20%), the AUC increases quickly to converge very close to 1.0, indicating an almost perfect recovery of relevant features/entries. This confirms that HDSL is able to accurately identify the small number of correct features and pairs of features. As α increases (i.e., the similarity constraints become noisy and less informative), the AUC increases at a slower pace and the final value decreases. This is expected as the quality of the information carried by the similarity judgments is key to recover the ground truth similarity. Yet, even for α =30%, the final AUC score is still very high (above 0.85 for both feature and entry recovery). This good recovery behavior is confirmed by the visual representations of the ground truth and learned similarity matrices shown in Figure 2. We observe that the learned similarity (when $\alpha = 20\%$) clearly recovers the block structure of the true similarity, and is able to correctly identify most individual entries with very few false positives.

5.1.2. Link Prediction

We now investigate the ability of our algorithm to generalize well as the feature dimensionality increases by conducting a signed link prediction experiment, which is the task of distinguishing positive and negative interactions in a network (see e.g. Agrawal et al., 2013).

We generate 500 samples with different number of features d ranging from $5,000$ to $1,000,000$. As the dimension d increases, we decrease the average sparsity of data (from 0.02 to 0.002) to limit running time. In real high-dimensional datasets, features typically do not appear in a uniform frequency: instead, a



(a) Feature frequency ($d = 50,000$) (b) AUC scores on the test set

Figure 3: Link prediction experiment on synthetic data. Figure 3(a) shows the feature frequency distribution, which follows a power law as in many real high-dimensional datasets. Figure 3(b) shows AUC scores on the test set for different number of features (in log scale) and number of training constraints per link.

small portion of features tends to dominate the others. Following this observation, we generate features whose frequency follow a power law style distribution, as shown in Figure 3(a). The ground truth similarity is then a convex combination of randomly selected bases as in the previous experiment, except that we restrict the selected bases to those involving features that are frequent enough (a frequency of at least 0.1 was chosen for this experiment). This is needed to ensure that the features involved in the ground truth similarity will occur at least a few times in our small dataset, but we emphasize that the algorithm is exposed to the entire feature set and does not know which features are relevant.

Based on the samples and the ground truth similarity, we generate signed link observations of the form $\{x_1^i, x_2^i, y^i\}_i^N$ ($y^i \in \{-1, 1\}$). We associate the label $y^i = 1$ (positive link) to pairs for which the similarity between x_1 and x_2 ranks in the top 5% of x_1's (or x_2's) neighbors according to the ground truth similarity measure. On the other hand, $y^i = -1$ (negative link) indicates that the similarity ranks in the bottom 5% of x_1's (or x_2's) neighbors. We split these link observations into training, validation and test sets of $1,000$ observations each. Triplets constraints are generated from training links — given a pair x_1, x_2, y, we randomly sample x_3 as a similar (if $y = -1$) or dissimilar (if $y = 1$) node. The validation set is used to tune the hyperparameter λ and for early stopping.

Results. We measure the generalization ability of HDSL by the AUC score of link prediction on the test set. Figure 3(b) reports these AUC scores across different dimensions. We also show results for different numbers of constraints per training link. The results are averaged over 5 random runs. As one would expect, the task becomes increasingly difficult as the dimension becomes larger, since the size of the training set is fixed ($1,000$ training links generated from 500 nodes). However, the performance decreases slowly (roughly logarithmically) with the dimension, and we achieve very high AUC scores (larger than 0.9)

Datasets	Dimension	Sparsity	Training size	Validation size	Test size
dexter	20,000	0.48%	300	300	2,000
dorothea	100,000	0.91%	800	350	800
rcv1_2	47,236	0.16%	12,145	4,048	4,049
rcv1_4	29,992	0.26%	3,850	2,888	2,887

Table 2: Datasets used in the experiments

even for one million features. We also see that training from more constraints tends to improve the prediction performance.

5.2. Experiments on Real Datasets

We now present comparative experiments on several high-dimensional real datasets, evaluating our approach against several baselines and competing methods.

5.2.1. Setup

Datasets. We report experimental results on several real-world classification datasets with up to 100,000 features. Dorothea and dexter come from the NIPS 2003 feature selection challenge (Guyon et al., 2004) and are respectively pharmaceutical and text data with predefined splitting into training, validation and test sets. They both contain a large proportion of noisy/irrelevant features. Reuters CV1 is a popular text classification dataset with bag-of-words representation. We use the binary classification version from the LIBSVM dataset collection[3] (with 60%/20%/20% random splits) and the 4-classes version (with 40%/30%/30% random splits) introduced by Cai and He (2012). Detailed information on the datasets and splits is given in Table 2. All datasets are normalized such that each feature takes values in $[0, 1]$.

Competing methods. We compare the proposed approach (HDSL) to several methods:

- DOT: The standard dot product, which is equivalent to setting $\boldsymbol{M} = \boldsymbol{I}$.
- DIAG: Diagonal similarity learning (i.e., a weighting of the features), as done in Gao et al. (2014). We obtain it by minimizing the same loss as in HDSL with ℓ_2 and ℓ_1 regularization, i.e.,

$$\min_{\boldsymbol{w}\in\mathbb{R}^d} \quad f(\boldsymbol{w}) = \frac{1}{T}\sum_{t=1}^{T} \ell\left(\langle \boldsymbol{A}^t, \mathrm{diag}(\boldsymbol{w})\rangle\right) + \lambda\Omega(\boldsymbol{w}),$$

  where $\Omega(\boldsymbol{w}) \in \{\|\boldsymbol{w}\|_2^2, \|\boldsymbol{w}\|_1\}$ and λ is the regularization parameter. Optimization was done using (proximal) gradient descent.

[3] `http://www.csie.ntu.edu.tw/~cjlin/libsvmtools/datasets/`

- RP+OASIS: Similarity learning in random projected space. Given $r \ll d$, let $\boldsymbol{R} \in \mathbb{R}^{d \times r}$ be a matrix where each entry r_{ij} is randomly drawn from $\mathcal{N}(0, 1)$. For each data instance $\boldsymbol{x} \in \mathbb{R}^d$, we generate $\tilde{\boldsymbol{x}} = \frac{1}{\sqrt{r}} \boldsymbol{R}^{\mathrm{T}} \boldsymbol{x} \in \mathbb{R}^r$ and use this reduced data in OASIS (Chechik et al., 2009), a fast online method to learn a bilinear similarity from triplet constraints.

- PCA+OASIS: Similarity learning in PCA space. Same as RP+OASIS, except that PCA is used instead of random projections to project the data into $\mathbb{R}^r$.

- SVM: Support Vector Machines. We use linear SVM, which is known to perform well for sparse high-dimensional data (Caruana et al., 2008), with ℓ_2 and ℓ_1 regularization. We also use nonlinear SVM with the polynomial kernel (2nd and 3rd degree) popular in text classification (Chang et al., 2010). The SVM models are trained using liblinear (Fan et al., 2008) and libsvm (Chang and Lin, 2011) with 1vs1 paradigm for multiclass.

We have also tried to compare our method with COMET (Atzmon et al., 2015), which also learns a bilinear similarity in a greedy fashion with rank-1 updates. However, as mentioned in Section 2.1 their coordinate descent algorithm has a time complexity of $O(d^2)$ per iteration, as well as overall memory complexity of $O(d^2)$. We run the sparse version of code provided by the authors[4] on a machine with a 2.3GHz Intel Core i7 and 16GB memory. On the dexter dataset (which has the smallest dimensionality in our benchmark), a single pass over the features took more than 4 hours, while the authors reported that about 10 passes are generally needed for COMET to converge (Atzmon et al., 2015). On the dorothea dataset, COMET returned a memory error. As a result, we did not include COMET to our empirical comparison. In contrast, on the same hardware, our approach HDSL takes less than 1 minute on dexter and less than 1 hour on dorothea.

Training Procedure. For all similarity learning algorithms, we generate 15 training constraints for each instance by identifying its 3 target neighbors (nearest neighbors with same label) and 5 impostors (nearest neighbors with different label), following Weinberger and Saul (2009). Due to the very small number of training instances in dexter, we found that better performance is achieved across all methods when using 20 training constraints per instance, drawn at random based on its label. All parameters are tuned using the accuracy on the validation set. For HDSL, we use the fast heuristic described in Section 3.2.3 and tune the scale parameter $\lambda \in \{1, 10, \ldots, 10^9\}$. The regularization parameters of DIAG and SVM are tuned in $\{10^{-9}, \ldots, 10^8\}$ and the "aggressiveness" parameter of OASIS is tuned in $\{10^{-9}, \ldots, 10^2\}$.

Dataset	DOT	RP+OASIS	PCA+OASIS	DIAG-ℓ_2	DIAG-ℓ_1	HDSL
dexter	20.1	24.0 [1000]	9.3 [50]	8.4	8.4 [773]	**6.5** [183]
dorothea	9.3	11.4 [150]	9.9 [800]	6.8	6.6 [860]	**6.5** [731]
rcv1_2	6.9	7.0 [2000]	4.5 [1500]	3.5	3.7 [5289]	**3.4** [2126]
rcv1_4	11.2	10.6 [1000]	6.1 [800]	6.2	7.2 [3878]	**5.7** [1888]

Table 3: k-NN test error (%) of the similarities learned with each method. The number of features used by each similarity (when smaller than d) is given in brackets. Best accuracy on each dataset is shown in bold.

Dataset	SVM-poly-2	SVM-poly-3	SVM-linear-ℓ_2	SVM-linear-ℓ_1	HDSL
dexter	9.4	9.2	8.9	8.9 [281]	**6.5** [183]
dorothea	7	6.6	8.1	6.6 [366]	**6.5** [731]
rcv1_2	3.4	**3.3**	3.5	4.0 [1915]	3.4 [2126]
rcv1_4	5.7	5.7	**5.1**	5.7 [2770]	5.7 [1888]

Table 4: Test error (%) of several SVM variants compared to HDSL. As in Table 3, the number of features is given in brackets and best accuracies are shown in bold.

5.2.2. Results

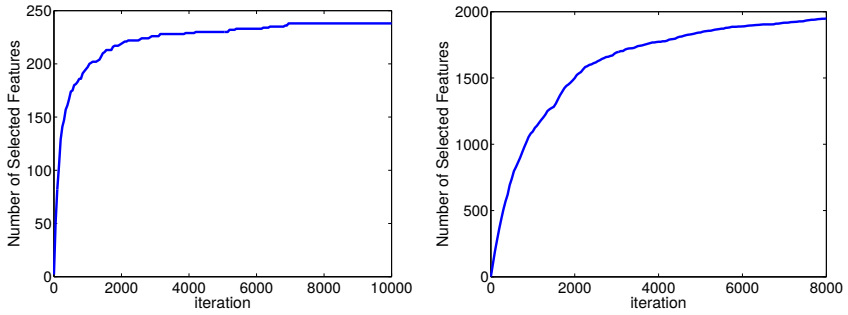
Classification Performance. We first investigate the performance of each similarity learning approach in k-NN classification (k was set to 3 for all experiments). For RP+OASIS and PCA+OASIS, we choose the dimension r of the reduced space based on the accuracy of the learned similarity on the validation set, limiting our search to $r \leq 2000$ because OASIS is extremely slow beyond this point.[5] Similarly, we use the performance on validation data to do early stopping in HDSL, which also has the effect of restricting the number of features used by the learned similarity.

Table 3 shows the k-NN classification performance. We can first observe that RP+OASIS often performs worse than DOT, which is consistent with previous observations showing that a large number of random projections may be needed to obtain good performance (Fradkin and Madigan, 2003). PCA+OASIS gives much better results, but is generally outperformed by a simple diagonal similarity learned directly in the original high-dimensional space. HDSL, however, outperforms all other algorithms on these datasets, including DIAG. This shows the good generalization performance of the proposed approach, even though the number of training samples is sometimes very small compared to the number of features, as in dexter and dorothea. It also shows the relevance of encoding "second order" information (pairwise interactions between the original features) in the similarity instead of considering a simple weighting of features as in DIAG.

Table 4 shows the comparison with SVMs. Interestingly, HDSL outperforms all SVM variants on dexter and dorothea, both of which have a large proportion

[4] `https://github.com/yuvalatzmon/COMET`

[5] Note that the number of PCA dimensions is at most the number of training examples. Therefore, for dexter and dorothea, r is at most 300 and 800 respectively.



(a) dexter dataset (b) rcv1_4 dataset

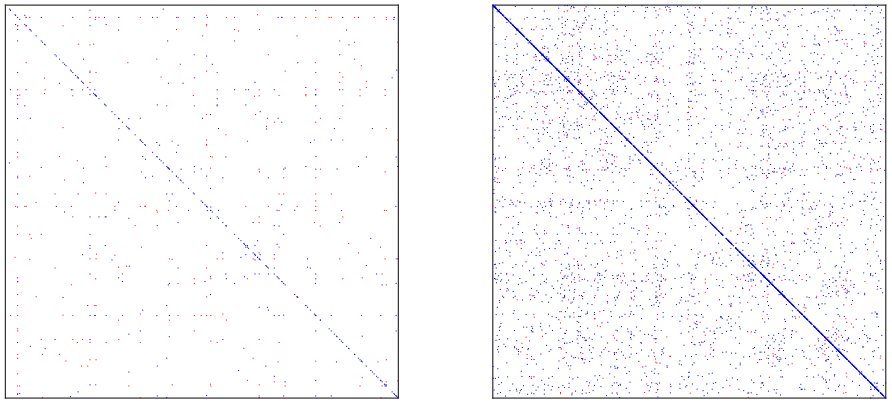
Figure 4: Number of active features learned by HDSL as a function of the iteration number.

of irrelevant features. This shows that its greedy strategy and early stopping mechanism achieves better feature selection and generalization than the ℓ_1 version of linear SVM. On the other two datasets, HDSL is competitive with SVM, although it is outperformed slightly by one variant (SVM-poly-3 on rcv1_2 and SVM-linear-ℓ_2 on rcv1_4), both of which rely on all features.

Feature selection and sparsity. We now focus on the ability of HDSL to perform feature selection and more generally to learn sparse similarity functions. To better understand the behavior of HDSL, we show in Figure 4 the number of selected features as a function of the iteration number for two of the datasets. Remember that at most two new features can be added at each iteration. Figure 4 shows that HDSL incorporates many features early on but tends to eventually converge to a modest fraction of features (the same observation holds for the other two datasets). This may explain why HDSL does not suffer much from overfitting even when training data is scarce as in dexter.

Another attractive characteristic of HDSL is its ability to learn a matrix that is sparse not only on the diagonal but also off-diagonal (the proportion of nonzero entries is in the order of 0.0001% for all datasets). In other words, the learned similarity only relies on a few relevant pairwise interactions between features. Figure 5 shows two examples, where we can see that HDSL is able to exploit the product of two features as either a positive or negative contribution to the similarity score. This opens the door to an analysis of the importance of pairs of features (for instance, word co-occurrence) for the application at hand. Finally, the extreme sparsity of the matrices allows very fast similarity computation. Together with the superior accuracy brought by HDSL, it makes our approach potentially useful in a variety of contexts (k-NN, clustering, ranking, etc).

Finally, it is also worth noticing that HDSL uses significantly less features than DIAG-ℓ_1 (see numbers in brackets in Table 3). We attribute this to the extra modeling capability brought by the non-diagonal similarity observed in



(a) dexter ($20,000 \times 20,000$ matrix, 712 nonzeros)

(b) rcv1_4 ($29,992 \times 29,992$ matrix, 5263 nonzeros)

Figure 5: Sparsity structure of the matrix $\boldsymbol{M}$ learned by HDSL. Positive and negative entries are shown in blue and red, respectively (best seen in color).

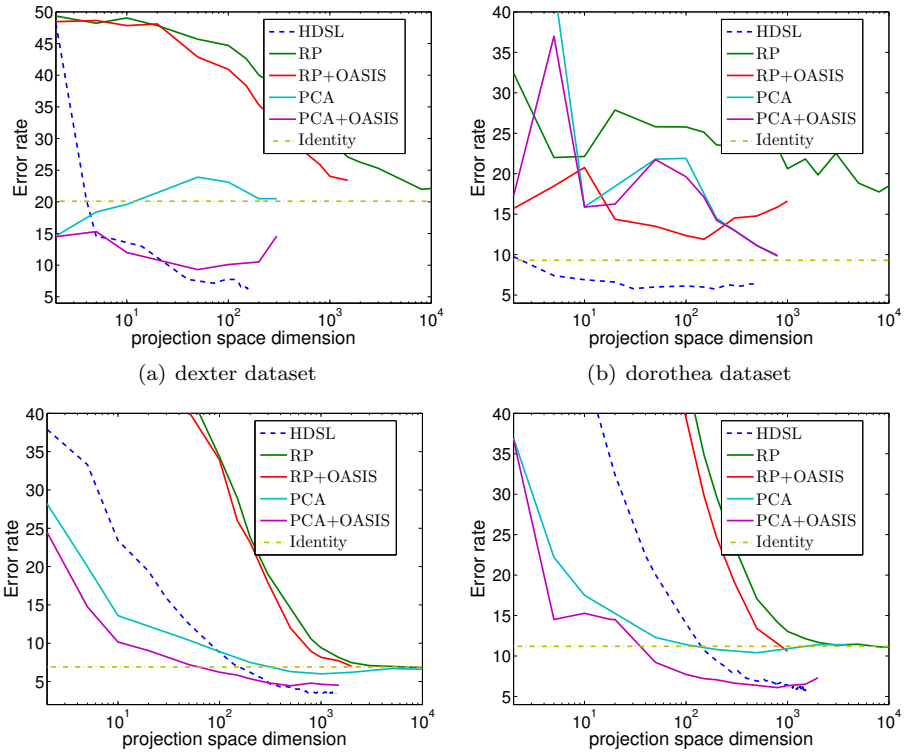
Figure 5.[6]

Dimension reduction. We now investigate the potential of HDSL for dimensionality reduction. Recall that HDSL learns a sequence of PSD matrices $\boldsymbol{M}^{(k)}$. We can use the square root of $\boldsymbol{M}^{(k)}$ to project the data into a new space where the dot product is equivalent to $S_{\boldsymbol{M}^{(k)}}$ in the original space. The dimension of the projection space is equal to the rank of $\boldsymbol{M}^{(k)}$, which is upper bounded by $k+1$ (see Section 3.1). A single run of HDSL can thus be seen as incrementally building projection spaces of increasing dimensionality.

To assess the dimensionality reduction quality of HDSL (measured by k-NN classification error on the test set), we plot its performance at various iterations during the runs that generated the results of Table 3. We compare it to two standard dimensionality reduction techniques: random projection and PCA. We also evaluate RP+OASIS and PCA+OASIS, i.e., learn a similarity with OASIS on top of the RP and PCA features.[7] Note that OASIS was tuned separately for each projection size, making the comparison a bit unfair to HDSL. The results are shown in Figure 6. As observed earlier, random projection-based approaches achieve poor performance. When the features are not too noisy (as in rcv1_2 and rcv1_4), PCA-based methods are better than HDSL at compressing the space into very few dimensions, but HDSL eventually catches up. On the other hand, PCA suffers heavily from the presence of noise (dexter and dorothea), while

[6] Note that HDSL uses roughly the same number of features as SVM-linear-ℓ_1 (Table 4), but it is difficult to draw any solid conclusion because the objective and training data for each method are different, and SVM is a combination of binary models.

[7] Again, we were not able to run OASIS beyond a certain dimension due to computational complexity.



(a) dexter dataset (b) dorothea dataset

(c) rcv1_2 dataset (d) rcv1_4 dataset

Figure 6: k-NN test error as a function of the dimensionality of the space (in log scale). Best seen in color.

HDSL is able to quickly improve upon the standard similarity in the original space. Finally, on all datasets, we observe that HDSL converges to a stationary dimension without overfitting, unlike PCA+OASIS which exhibits signs of overfitting on dexter and rcv1_4 especially.

6. Concluding Remarks

In this work, we proposed an efficient approach to learn similarity functions from high-dimensional sparse data. This is achieved by forming the similarity as a combination of simple sparse basis elements that operate on only two features and the use of an (approximate) Frank-Wolfe algorithm. Our algorithm is completed by a novel generalization analysis which validates the design choices and highlights the robustness of our approach to high dimensions. Experiments on synthetic and real datasets confirmed the good practical behavior of our method for classification and dimensionality reduction. The learned similarity may be applied to other algorithms that rely on a similarity function (clustering,

ranking), or as a way to preprocess the data before applying another learning algorithm. We also note that St.Amand and Huan (2017) have recently extended our HDSL algorithm to learn local metrics for different regions of the space in addition to the global metric.

We leave several fundamental questions for future work. In particular, our framework could be extended to optimize a loss function related to a linear classification objective. We could then attempt to adapt our analysis to obtain generalization bounds directly for the classification error. Such bounds exist in the literature (see Bellet et al., 2012; Guo and Ying, 2014) but exhibit a classic dependence on the data dimension that could be avoided with our approach. Another interesting, though challenging direction is to formally study the conditions under which a sparse ground truth similarity can be accurately recovered from similarity judgments. Inspiration could be drawn from the related problem of sparse recovery in the compressed sensing literature (Foucart and Rauhut, 2013).

Acknowledgments. This work was partially supported by a grant from CPER Nord-Pas de Calais/FEDER DATA Advanced data science and technologies 2015-2020. It was also supported in part by the Intelligence Advanced Research Projects Activity (IARPA) via Department of Defense U.S. Army Research Laboratory (DoD / ARL) contract number W911NF-12-C-0012, a NSF IIS-1065243, an Alfred. P. Sloan Research Fellowship, DARPA award D11AP00278, and an ARO YIP Award (W911NF-12-1-0241). The U.S. Government is authorized to reproduce and distribute reprints for Governmental purposes notwithstanding any copyright annotation thereon. The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies or endorsements, either expressed or implied, of IARPA, DoD/ARL, or the U.S. Government.

References

Agrawal, P., Garg, V.K., Narayanam, R., 2013. Link label prediction in signed social networks., in: IJCAI, pp. 2591–2597.

Atzmon, Y., Shalit, U., Chechik, G., 2015. Learning sparse metrics, one feature at a time, in: NIPS 2015 Workshop on Feature Extraction: Modern Questions and Challenges.

Bardenet, R., Maillard, O.A., 2015. Concentration inequalities for sampling without replacement. Bernoulli 21, 1361–1385.

Bellet, A., Habrard, A., 2015. Robustness and Generalization for Metric Learning. Neurocomputing 151, 259–267.

Bellet, A., Habrard, A., Sebban, M., 2012. Similarity Learning for Provably Accurate Sparse Linear Classification, in: ICML, pp. 1871–1878.

Bellet, A., Habrard, A., Sebban, M., 2013. A Survey on Metric Learning for Feature Vectors and Structured Data. Technical Report. arXiv:1306.6709.

Bellet, A., Habrard, A., Sebban, M., 2015. Metric Learning. Morgan & Claypool Publishers.

Bian, W., Tao, D., 2011. Learning a Distance Metric by Empirical Loss Minimization, in: IJCAI, pp. 1186–1191.

Cai, D., He, X., 2012. Manifold Adaptive Experimental Design for Text Categorization. IEEE Transactions on Knowledge and Data Engineering 24, 707–719.

Cao, Q., Guo, Z.C., Ying, Y., 2012a. Generalization Bounds for Metric and Similarity Learning. Technical Report. University of Exeter. ArXiv:1207.5437.

Cao, Q., Ying, Y., Li, P., 2012b. Distance Metric Learning Revisited, in: ECML/PKDD, pp. 283–298.

Caruana, R., Karampatziakis, N., Yessenalina, A., 2008. An empirical evaluation of supervised learning in high dimensions, in: ICML, pp. 96–103.

Chang, C.C., Lin, C.J., 2011. LIBSVM : a library for support vector machines. ACM Transactions on Intelligent Systems and Technology 2, 27–27.

Chang, Y.W., Hsieh, C.J., Chang, K.W., Ringgaard, M., Lin, C.J., 2010. Training and Testing Low-degree Polynomial Data Mappings via Linear SVM. Journal of Machine Learning Research 11, 1471–1490.

Chechik, G., Shalit, U., Sharma, V., Bengio, S., 2009. An online algorithm for large scale image similarity learning., in: NIPS, pp. 306–314.

Chen, Y., Pavlov, D., Canny, J.F., 2009. Large-scale behavioral targeting, in: KDD.

Clarkson, K.L., 2010. Coresets, sparse greedy approximation, and the Frank-Wolfe algorithm. ACM Transactions on Algorithms 6, 1–30.

Clémençon, S., Colin, I., Bellet, A., 2016. Scaling-up Empirical Risk Minimization: Optimization of Incomplete U-statistics. Journal of Machine Learning Research 17, 1–36.

Clémençon, S., Lugosi, G., Vayatis, N., 2008. Ranking and Empirical Minimization of U-statistics. Annals of Statistics 36, 844–874.

Davis, J.V., Kulis, B., Jain, P., Sra, S., Dhillon, I.S., 2007. Information-theoretic metric learning, in: ICML, pp. 209–216.

Fan, R.E., Chang, K.W., Hsieh, C.J., Wang, X.R., Lin, C.J., 2008. LIBLINEAR: A Library for Large Linear Classification. Journal of Machine Learning Research 9, 1871–1874.

Foucart, S., Rauhut, H., 2013. A Mathematical Introduction to Compressive Sensing. Birkhaüser.

Fradkin, D., Madigan, D., 2003. Experiments with random projections for machine learning, in: KDD, pp. 517–522.

Frank, M., Wolfe, P., 1956. An algorithm for quadratic programming. Naval Research Logistics Quarterly 3, 95–110.

Freund, R.M., Grigas, P., 2013. New Analysis and Results for the Conditional Gradient Method. Technical Report. arXiv:1307.0873.

Gao, X., Hoi, S.C., Zhang, Y., Wan, J., Li, J., 2014. SOML: Sparse Online Metric Learning with Application to Image Retrieval, in: AAAI, pp. 1206–1212.

Goldberger, J., Roweis, S., Hinton, G., Salakhutdinov, R., 2004. Neighbourhood Components Analysis, in: NIPS, pp. 513–520.

Guélat, J., Marcotte, P., 1986. Some comments on Wolfe's away step. Mathematical Programming 35, 110–119.

Guillaumin, M., Verbeek, J.J., Schmid, C., 2009. Is that you? Metric learning approaches for face identification, in: ICCV, pp. 498–505.

Guo, Z.C., Ying, Y., 2014. Guaranteed Classification via Regularized Similarity Learning. Neural Computation 26, 497–522.

Guyon, I., Gunn, S.R., Ben-Hur, A., Dror, G., 2004. Result Analysis of the NIPS 2003 Feature Selection Challenge, in: NIPS.

Hoeffding, W., 1948. A Class of Statistics with Asymptotically Normal Distribution. The Annals of Mathematical Statistics 19, 293–325.

Jaggi, M., 2011. Sparse Convex Optimization Methods for Machine Learning. Ph.D. thesis. ETH Zurich.

Jaggi, M., 2013. Revisiting Frank-Wolfe: Projection-Free Sparse Convex Optimization, in: ICML.

Jain, L., Mason, B., Nowak, R., 2017. Learning Low-Dimensional Metrics, in: NIPS.

Jin, R., Wang, S., Zhou, Y., 2009. Regularized Distance Metric Learning: Theory and Algorithm, in: NIPS.

Kedem, D., Tyree, S., Weinberger, K., Sha, F., Lanckriet, G., 2012. Non-linear Metric Learning, in: NIPS, pp. 2582–2590.

Kulis, B., 2012. Metric Learning: A Survey. Foundations and Trends in Machine Learning 5, 287–364.

Lacoste-Julien, S., Jaggi, M., 2015. On the Global Linear Convergence of Frank-Wolfe Optimization Variants, in: NIPS.

Leach, A.R., Gillet, V.J., 2007. An Introduction to Chemoinformatics. Springer.

Lee, A.J., 1990. U-Statistics: Theory and Practice. Marcel Dekker, New York.

Lim, D.K., McFee, B., Lanckriet, G., 2013. Robust Structural Metric Learning, in: ICML.

Liu, K., Bellet, A., Sha, F., 2015a. Similarity Learning for High-Dimensional Sparse Data, in: AISTATS, pp. 653–662.

Liu, W., Mu, C., Ji, R., Ma, S., Smith, J.R., Chang, S.F., 2015b. Low-Rank Similarity Metric Learning in High Dimensions, in: AAAI.

McDiarmid, C., 1989. On the method of bounded differences. Surveys in combinatorics 141, 148–188.

Qi, G.J., Tang, J., Zha, Z.J., Chua, T.S., Zhang, H.J., 2009. An Efficient Sparse Metric Learning in High-Dimensional Space via l1-Penalized Log-Determinant Regularization, in: ICML.

Qian, Q., Jin, R., Zhang, L., Zhu, S., 2015. Towards Making High Dimensional Distance Metric Learning Practical. Technical Report. arXiv:1509.04355.

Qian, Q., Jin, R., Zhu, S., Lin, Y., 2014. An Integrated Framework for High Dimensional Distance Metric Learning and Its Application to Fine-Grained Visual Categorization. Technical Report. arXiv:1402.0453.

Rosales, R., Fung, G., 2006. Learning Sparse Metrics via Linear Programming, in: KDD, pp. 367–373.

Schultz, M., Joachims, T., 2003. Learning a Distance Metric from Relative Comparisons, in: NIPS.

Serfling, R.J., 1974. Probability inequalities for the sum in sampling without replacement. The Annals of Statistics 2, 39–48.

Shalev-Shwartz, S., Ben-David, S., 2014. Understanding Machine Learning: From Theory to Algorithms. Cambridge University Press.

Shen, C., Kim, J., Wang, L., van den Hengel, A., 2012. Positive Semidefinite Metric Learning Using Boosting-like Algorithms. Journal of Machine Learning Research 13, 1007–1036.

Shi, Y., Bellet, A., Sha, F., 2014. Sparse Compositional Metric Learning, in: AAAI, pp. 2078–2084.

St.Amand, J., Huan, J., 2017. Sparse Compositional Local Metric Learning, in: KDD.

Verma, N., Branson, K., 2015. Sample complexity of learning mahalanobis distance metrics, in: NIPS.

Wang, J., Woznica, A., Kalousis, A., 2012. Parametric Local Metric Learning for Nearest Neighbor Classification, in: NIPS, pp. 1610–1618.

Weinberger, K.Q., Saul, L.K., 2009. Distance Metric Learning for Large Margin Nearest Neighbor Classification. Journal of Machine Learning Research 10, 207–244.

Yao, D., Zhao, P., Pham, T.A.N., Cong, G., 2018. High-dimensional Similarity Learning via Dual-sparse Random Projection, in: IJCAI.

Ying, Y., Huang, K., Campbell, C., 2009. Sparse Metric Learning via Smooth Optimization, in: NIPS, pp. 2214–2222.

Ying, Y., Li, P., 2012. Distance Metric Learning with Eigenvalue Optimization. Journal of Machine Learning Research 13, 1–26.

Zhang, J., Zhang, L., 2017. Efficient Stochastic Optimization for Low-Rank Distance Metric Learning, in: AAAI.

Appendix A. Technical Lemmas

The following classic result, known as the first Hoeffding's decomposition, allows to represent a U-statistic as a sum of i.i.d. blocks.

Lemma 2 (Hoeffding, 1948)**.** *Let $q : \mathcal{Z} \times \mathcal{Z} \times \mathcal{Z} \to \mathbb{R}$ be a real-valued function. Given the i.i.d. random variables $\boldsymbol{z}_1, \boldsymbol{z}_2, ..., \boldsymbol{z}_n \in \mathcal{Z}$, we have*

$$\begin{aligned} U_n(q) &= \frac{1}{n(n-1)(n-2)} \sum_{i \neq j \neq k} q(\boldsymbol{z}_i, \boldsymbol{z}_j, \boldsymbol{z}_k) \\ &= \frac{1}{n!} \sum_{\pi} \frac{1}{\lfloor n/3 \rfloor} \sum_{i=1}^{\lfloor n/3 \rfloor} q\big(\boldsymbol{z}_{\pi(i)}, \boldsymbol{z}_{\pi(i+\lfloor n/3 \rfloor)}, \boldsymbol{z}_{\pi(i+2\times\lfloor n/3 \rfloor)}\big). \end{aligned}$$

Proof. Observe that $\forall i \neq j \neq k$, $q(\boldsymbol{z}_i, \boldsymbol{z}_j, \boldsymbol{z}_k)$ appears once on the left hand side and $U_n(q)$ has $\frac{1}{n(n-1)(n-2)}$ of its value, while on the right hand side it appears $(n-3)! \times \lfloor n/3 \rfloor$ times, because for each of the $\lfloor n/3 \rfloor$ positions there are $(n-3)!$ possible permutations. Thus the right hand side also has $\frac{1}{n(n-1)(n-2)}$ of its function value. We thus have the equality. □

The next technical lemma is based on the above representation.

Lemma 3. *Let Q be a set of functions from $\mathcal{Z}^3$ to $\mathbb{R}$. If $\boldsymbol{z}_1, \boldsymbol{z}_2, ..., \boldsymbol{z}_n \in \mathcal{Z}$ are i.i.d., then we have*

$$\mathbb{E}[\sup_{q\in Q} \frac{1}{n(n-1)(n-2)} \sum_{i\neq j\neq k} q(\boldsymbol{z}_i, \boldsymbol{z}_j, \boldsymbol{z}_k)] \\ \leq \mathbb{E}[\sup_{q\in Q} \frac{1}{\lfloor n/3 \rfloor} \sum_{i=1}^{\lfloor n/3 \rfloor} q(\boldsymbol{z}_i, \boldsymbol{z}_{i+\lfloor n/3 \rfloor}, \boldsymbol{z}_{i+2\times\lfloor n/3 \rfloor})].$$

Proof. From Lemma 2, we observe that

$$\begin{aligned}
&\mathbb{E}[\sup_{q\in Q} \frac{1}{n(n-1)(n-2)} \sum_{\boldsymbol{z}\neq\boldsymbol{z}'\neq\boldsymbol{z}''} q(\boldsymbol{z}, \boldsymbol{z}', \boldsymbol{z}'')] \\
&= \mathbb{E}[\sup_{q\in Q} \frac{1}{n!} \sum_{\pi} \frac{1}{\lfloor n/3 \rfloor} \sum_{i=1}^{\lfloor n/3 \rfloor} q(\boldsymbol{z}_{\pi(i)}, \boldsymbol{z}_{\pi(i+\lfloor n/3 \rfloor)}, \boldsymbol{z}_{\pi(i+2\times\lfloor n/3 \rfloor)})] \\
&\leq \frac{1}{n!} \mathbb{E}[\sum_{\pi} \sup_{q\in Q} \frac{1}{\lfloor n/3 \rfloor} \sum_{i=1}^{\lfloor n/3 \rfloor} q(\boldsymbol{z}_{\pi(i)}, \boldsymbol{z}_{\pi(i+\lfloor n/3 \rfloor)}, \boldsymbol{z}_{\pi(i+2\times\lfloor n/3 \rfloor)})] \\
&= \frac{1}{n!} \sum_{\pi} \mathbb{E}[\sup_{q\in Q} \frac{1}{\lfloor n/3 \rfloor} \sum_{i=1}^{\lfloor n/3 \rfloor} q(\boldsymbol{z}_{\pi(i)}, \boldsymbol{z}_{\pi(i+\lfloor n/3 \rfloor)}, \boldsymbol{z}_{\pi(i+2\times\lfloor n/3 \rfloor)})] \\
&= \mathbb{E}[\sup_{q\in Q} \frac{1}{\lfloor n/3 \rfloor} \sum_{i=1}^{\lfloor n/3 \rfloor} q(\boldsymbol{z}_i, \boldsymbol{z}_{i+\lfloor n/3 \rfloor}, \boldsymbol{z}_{i+2\times\lfloor n/3 \rfloor})],
\end{aligned}$$

which proves the result. □

Finally, we recall McDiarmid's inequality.

Lemma 4 (McDiarmid, 1989)**.** *Let $\mathcal{Z}$ be some set and let $f : \mathcal{Z}^n \to \mathbb{R}$ be a function of n variables such that for some $c > 0$, for all $i \in \{1, \ldots, n\}$ and for all $z_1, \ldots, z_n, z_i' \in \mathcal{Z}$, we have*

$$|f(z_1, \ldots, z_{i-1}, z_i, z_{i+1}, \ldots, z_n) - f(z_1, \ldots, z_{i-1}, z_i', z_{i+1}, \ldots, z_n)| \leq c.$$

Let $Z_1, \ldots, Z_n$ be n independent random variables taking values in $\mathcal{Z}$. Then, with probability at least $1 - \delta$, we have

$$|f(Z_1, \ldots, Z_n) - \mathbb{E}[f(Z_1, \ldots, Z_n)]| \leq c\sqrt{\frac{n \log(2/\delta)}{2}}.$$

Appendix B. Proof of Lemma 1

Proof. Given a training sample $S = \{\boldsymbol{z}_i = (\boldsymbol{x}_i, y_i) : i \in 1, \ldots, n\} \sim \mu^n$, we denote the set of admissible triplets involved in the Rademacher complexity by

$$A_S = \left\{ i : y_i = y_{i+\lfloor n/3 \rfloor} \neq y_{i+2\times\lfloor n/3 \rfloor}, i = 1, \ldots, \lfloor n/3 \rfloor \right\},$$

and let $m = |A_S| \leq \lfloor n/3 \rfloor$. We have:

$$
\begin{aligned}
R_n(\mathcal{F}^{(k)}) &= \mathbb{E}_{\boldsymbol{\sigma}, S\sim\mu^n} \sup_{\boldsymbol{M}\in\mathcal{D}_\lambda^{(k)}} \frac{1}{\lfloor n/3 \rfloor} \sum_{i\in A_S} \sigma_i \ell(\langle \boldsymbol{x}_i(\boldsymbol{x}_{i+\lfloor n/3\rfloor} - \boldsymbol{x}_{i+2\times\lfloor n/3\rfloor})^{\mathrm{T}}, \boldsymbol{M}\rangle) \\
&\leq \mathbb{E}_{\boldsymbol{\sigma}, S\sim\mu^n} \sup_{\boldsymbol{M}\in\mathcal{D}_\lambda^{(k)}} \frac{1}{\lfloor n/3 \rfloor} \sum_{i\in A_S} \sigma_i \langle \boldsymbol{x}_i(\boldsymbol{x}_{i+\lfloor n/3\rfloor} - \boldsymbol{x}_{i+2\times\lfloor n/3\rfloor})^{\mathrm{T}}, \boldsymbol{M}\rangle \qquad \text{(B.1)} \\
&= \frac{m}{\lfloor n/3 \rfloor} \mathbb{E}_{\boldsymbol{\sigma}, S\sim\mu^n} \frac{1}{m} \sup_{\boldsymbol{M}\in\mathcal{D}_\lambda^{(k)}} \sum_{i\in A_S} \sigma_i \langle \boldsymbol{x}_i(\boldsymbol{x}_{i+\lfloor n/3\rfloor} - \boldsymbol{x}_{i+2\times\lfloor n/3\rfloor})^{\mathrm{T}}, \boldsymbol{M}\rangle \\
&\leq \frac{m}{\lfloor n/3 \rfloor} \max_{\boldsymbol{u}\in U} \|\boldsymbol{u} - \bar{\boldsymbol{u}}\|_2 \frac{\sqrt{2\log k}}{m} \qquad \text{(B.2)} \\
&= \frac{1}{\lfloor n/3 \rfloor} \max_{\boldsymbol{u}\in U} \|\boldsymbol{u} - \bar{\boldsymbol{u}}\|_2 \sqrt{2\log k} \\
&\leq \frac{1}{\lfloor n/3 \rfloor} 8\lambda B_{\mathcal{X}} \sqrt{m} \sqrt{2\log k} \qquad \text{(B.3)} \\
&\leq 8\lambda B_{\mathcal{X}} \sqrt{\frac{2\log k}{\lfloor n/3 \rfloor}},
\end{aligned}
$$

where the set $U = \{\boldsymbol{u}_\tau \in \mathbb{R}^m : \tau = 1, \ldots, k, (\boldsymbol{u}_\tau)_i = \langle \boldsymbol{x}_{\gamma(i)}(\boldsymbol{x}_{\gamma(i)+\lfloor n/3\rfloor} - \boldsymbol{x}_{\gamma(i)+2\times\lfloor n/3\rfloor})^T, \boldsymbol{B}_\tau\rangle, \gamma : \{1, \ldots, m\} \to A_S \text{ is bijective}, \boldsymbol{B}_\tau \in \mathcal{B}_\lambda\}$, and $\bar{\boldsymbol{u}} = \frac{1}{k}\sum_{\tau=1}^k \boldsymbol{u}_\tau$. The inequality (B.1) follows from the contraction property (see Shalev-Shwartz and Ben-David, 2014, Lemma 26.9). The inequality (B.2) follows from the fact $\boldsymbol{M}$ is a convex combination of set of k bases combined with the properties in Shalev-Shwartz and Ben-David (2014, Lemma 26.7, 26.8). Finally, inequality (B.3) follows from the sparsity structure of the bases and the fact that $\boldsymbol{x}_i(\boldsymbol{x}_j - \boldsymbol{x}_k)^T$ has no entries with absolute value greater than $B_{\mathcal{X}}$. □

Appendix C. Proof of Theorem 1

Proof. Let us consider the function

$$\Phi(S) = \sup_{\boldsymbol{M}\in\mathcal{D}_\lambda^{(k)}} [\mathcal{L}(\boldsymbol{M}) - \mathcal{L}_S(\boldsymbol{M})].$$

Let $S = \{\boldsymbol{z}_1, \ldots, \boldsymbol{z}_{q-1}, \boldsymbol{z}_q, \boldsymbol{z}_{q+1}, \ldots, \boldsymbol{z}_n\}$ and $S' = \{\boldsymbol{z}_1, \ldots, \boldsymbol{z}_{q-1}, \boldsymbol{z}'_q, \boldsymbol{z}_{q+1}, \ldots, \boldsymbol{z}_n\}$ be two samples differing by exactly one point. We have:

$$
\begin{aligned}
\Phi(S') - \Phi(S) &\leq \sup_{\boldsymbol{M}\in\mathcal{D}_\lambda^{(k)}} [\mathcal{L}_S(\boldsymbol{M}) - \mathcal{L}_{S'}(\boldsymbol{M})] \\
&\leq \frac{1}{n(n-1)(n-2)} \sup_{\boldsymbol{M}\in\mathcal{D}_\lambda^{(k)}} \sum_{i\neq j\neq k} |L_{\boldsymbol{M}}(\boldsymbol{z}_i, \boldsymbol{z}_j, \boldsymbol{z}_k) - L_{\boldsymbol{M}}(\boldsymbol{z}'_i, \boldsymbol{z}'_j, \boldsymbol{z}'_k)| \\
&\leq \frac{1}{n(n-1)(n-2)} 6(n-1)(n-2) B_{\mathcal{X}} B_{\mathcal{D}_\lambda^{(k)}} = \frac{6}{n} B_{\mathcal{X}} B_{\mathcal{D}_\lambda^{(k)}}.
\end{aligned}
$$

The first inequality comes from the fact that the difference of suprema does not exceed the supremum of the difference. The last inequality makes use of (11). Similarly, we can obtain $\Phi(S) - \Phi(S') \leq 6B_{\mathcal{X}} B_{\mathcal{D}_\lambda^{(k)}}/n$, thus we have $|\Phi(S) - \Phi(S')| \leq 6B_{\mathcal{X}} B_{\mathcal{D}_\lambda^{(k)}}/n$. We can therefore apply McDiarmid's inequality (see Lemma 4 in Appendix A) to $\Phi(S)$: for any $\delta > 0$, with probability at least $1 - \delta$ we have:

$$\sup_{\boldsymbol{M} \in \mathcal{D}_\lambda^{(k)}} [\mathcal{L}(\boldsymbol{M}) - \mathcal{L}_S(\boldsymbol{M})] \leq \mathbb{E}_S \sup_{\boldsymbol{M} \in \mathcal{D}_\lambda^{(k)}} [\mathcal{L}(\boldsymbol{M}) - \mathcal{L}_S(\boldsymbol{M})] + 3B_{\mathcal{X}} B_{\mathcal{D}_\lambda^{(k)}} \sqrt{\frac{2\ln(2/\delta)}{n}}. \tag{C.1}$$

We thus need to bound $\mathbb{E}_S \sup_{\boldsymbol{M} \in \mathcal{D}_\lambda^{(k)}} [\mathcal{L}(\boldsymbol{M}) - \mathcal{L}_S(\boldsymbol{M})]$. Applying Lemma 3 (see Appendix A) with $q_{\boldsymbol{M}}(\boldsymbol{z}, \boldsymbol{z}', \boldsymbol{z}'') = \mathcal{L}(\boldsymbol{M}) - L_{\boldsymbol{M}}(\boldsymbol{z}, \boldsymbol{z}', \boldsymbol{z}'')$ gives

$$\mathbb{E}_S \sup_{\boldsymbol{M} \in \mathcal{D}_\lambda^{(k)}} [\mathcal{L}(\boldsymbol{M}) - \mathcal{L}_S(\boldsymbol{M})] \leq \mathbb{E}_S \sup_{\boldsymbol{M} \in \mathcal{D}_\lambda^{(k)}} [\mathcal{L}(\boldsymbol{M}) - \bar{\mathcal{L}}_S(\boldsymbol{M})],$$

where $\bar{\mathcal{L}}_S(\boldsymbol{M}) = \frac{1}{\lfloor n/3 \rfloor} \sum_{i=1}^{\lfloor n/3 \rfloor} L_{\boldsymbol{M}}(\boldsymbol{z}_i, \boldsymbol{z}_{i+\lfloor n/3 \rfloor}, \boldsymbol{z}_{i+2\times\lfloor n/3 \rfloor})$. Let $\bar{S} = \{\bar{\boldsymbol{z}}_1, ..., \bar{\boldsymbol{z}}_n\}$ be an i.i.d. sample independent of S. Then

$$\begin{aligned}
\mathbb{E}_S \sup_{\boldsymbol{M} \in \mathcal{D}_\lambda^{(k)}} [\mathcal{L}(\boldsymbol{M}) - \bar{\mathcal{L}}_S(\boldsymbol{M})] &= \mathbb{E}_S \sup_{\boldsymbol{M} \in \mathcal{D}_\lambda^{(k)}} [\mathbb{E}_{\bar{S}} \bar{\mathcal{L}}_{\bar{S}}(\boldsymbol{M}) - \bar{\mathcal{L}}_S(\boldsymbol{M})] \\
&\leq \mathbb{E}_{S,\bar{S}} \sup_{\boldsymbol{M} \in \mathcal{D}_\lambda^{(k)}} [\bar{\mathcal{L}}_{\bar{S}}(\boldsymbol{M}) - \bar{\mathcal{L}}_S(\boldsymbol{M})].
\end{aligned}$$

Let $\sigma_1, \ldots, \sigma_{\lfloor \frac{n}{3} \rfloor} \in \{-1, 1\}$ be a collection of i.i.d. Rademacher variables. By standard symmetrization techniques, we have that

$$\begin{aligned}
&\mathbb{E}_{S,\bar{S}} \sup_{\boldsymbol{M} \in \mathcal{D}_\lambda^{(k)}} [\bar{\mathcal{L}}_{\bar{S}}(\boldsymbol{M}) - \bar{\mathcal{L}}_S(\boldsymbol{M})] \\
&= \mathbb{E}_{\boldsymbol{\sigma},S,\bar{S}} \frac{1}{\lfloor n/3 \rfloor} \sup_{\boldsymbol{M} \in \mathcal{D}_\lambda^{(k)}} \sum_{i=1}^{\lfloor n/3 \rfloor} \sigma_i \big[L_{\boldsymbol{M}}(\bar{\boldsymbol{z}}_i, \bar{\boldsymbol{z}}_{i+\lfloor n/3 \rfloor}, \bar{\boldsymbol{z}}_{i+2\times\lfloor n/3 \rfloor}) \\
&\quad - L_{\boldsymbol{M}}(\boldsymbol{z}_i, \boldsymbol{z}_{i+\lfloor n/3 \rfloor}, \boldsymbol{z}_{i+2\times\lfloor n/3 \rfloor}) \big] \\
&\leq \frac{1}{\lfloor n/3 \rfloor} [\mathbb{E}_{\boldsymbol{\sigma},\bar{S}} \sup_{\boldsymbol{M} \in \mathcal{D}_\lambda^{(k)}} \sum_{i=1}^{\lfloor n/3 \rfloor} \sigma_i L_{\boldsymbol{M}}(\bar{\boldsymbol{z}}_i, \bar{\boldsymbol{z}}_{i+\lfloor n/3 \rfloor}, \bar{\boldsymbol{z}}_{i+2\times\lfloor n/3 \rfloor}) \\
&\quad + \mathbb{E}_{\boldsymbol{\sigma},S} \sup_{\boldsymbol{M} \in \mathcal{D}_\lambda^{(k)}} \sum_{i=1}^{\lfloor n/3 \rfloor} \sigma_i L_{\boldsymbol{M}}(\boldsymbol{z}_i, \boldsymbol{z}_{i+\lfloor n/3 \rfloor}, \boldsymbol{z}_{i+2\times\lfloor n/3 \rfloor})] \\
&= 2\mathbb{E}_{\boldsymbol{\sigma},S} \frac{1}{\lfloor n/3 \rfloor} \sup_{\boldsymbol{M} \in \mathcal{D}_\lambda^{(k)}} \sum_{i=1}^{\lfloor n/3 \rfloor} \sigma_i L_{\boldsymbol{M}}(\boldsymbol{z}_i, \boldsymbol{z}_{i+\lfloor n/3 \rfloor}, \boldsymbol{z}_{i+2\times\lfloor n/3 \rfloor}) = 2R_n(\mathcal{F}^{(k)}).
\end{aligned}$$

We have thus shown:

$$\mathbb{E}_S \sup_{\boldsymbol{M} \in \mathcal{D}_\lambda^{(k)}} [\mathcal{L}(\boldsymbol{M}) - \mathcal{L}_S(\boldsymbol{M})] \leq 2R_n(\mathcal{F}^{(k)}). \tag{C.2}$$

Plugging (C.2) into (C.1) and using Lemma 1, we get the desired result. □

Appendix D. Proof of Corollary 1

Proof. The excess risk of $\boldsymbol{M}^{(k)}$ with respect to $\boldsymbol{M}^*$ can be decomposed as follows:

$$\begin{aligned}\mathcal{L}(\boldsymbol{M}^{(k)}) - \mathcal{L}(\boldsymbol{M}^*) &= \mathcal{L}(\boldsymbol{M}^{(k)}) - \mathcal{L}_S(\boldsymbol{M}^{(k)}) + \mathcal{L}_S(\boldsymbol{M}^{(k)}) - \mathcal{L}_S(\boldsymbol{M}_S) \\ &\quad + \mathcal{L}_S(\boldsymbol{M}_S) - \mathcal{L}_S(\boldsymbol{M}^*) + \mathcal{L}_S(\boldsymbol{M}^*) - \mathcal{L}(\boldsymbol{M}^*) \\ &\leq \underbrace{\mathcal{L}(\boldsymbol{M}^{(k)}) - \mathcal{L}_S(\boldsymbol{M}^{(k)})}_{\text{generalization error}} + \underbrace{\mathcal{L}_S(\boldsymbol{M}^{(k)}) - \mathcal{L}_S(\boldsymbol{M}_S)}_{\text{optimization error}} + \mathcal{L}_S(\boldsymbol{M}^*) - \mathcal{L}(\boldsymbol{M}^*), \quad \text{(D.1)}\end{aligned}$$

where $\boldsymbol{M}_S \in \arg\min_{\boldsymbol{M} \in \mathcal{D}_\lambda} \mathcal{L}_S(\boldsymbol{M})$ is an empirical risk minimizer.

The generalization error term in (D.1) can be bounded using Theorem 1 (recalling that $\boldsymbol{M}^{(k)} \in \mathcal{D}_\lambda^{(k)}$ by construction), while the optimization error term is bounded by the convergence rate of our Frank-Wolfe algorithm (Proposition 1). In the last term, $\boldsymbol{M}^*$ does not depend on S, hence we can use Hoeffding's inequality together with (11) and (14) to obtain that for any $\delta > 0$, with probability at least $1 - \delta/2$:

$$\mathcal{L}_S(\boldsymbol{M}^*) - \mathcal{L}(\boldsymbol{M}^*) \leq B_{\mathcal{X}} B_{\mathcal{D}_\lambda^{(k)}} \sqrt{\frac{\log(4/\delta)}{2n}}.$$

We get the corollary by combining the above results using the union bound. □