JMLR: Workshop and Conference Proceedings 44 (2015) 30-48 NIPS 2015

The 1st International Workshop "Feature Extraction: Modern Questions and Challenges"

Learning Sparse Metrics, One Feature at a Time

Yuval Atzmon YUVAL.ATZMON@BIU.AC.IL
Gonda brain research center, Bar Ilan University, Israel

Uri Shalit UAS1@NYU.EDU
Courant Institute of Mathematical Sciences
New York University, New York, USA

Gal Chechik GAL@GOOGLE.COM
Gonda brain research center, Bar Ilan University, Israel
and Google Research, Mountain View CA, USA

Editor: Dmitry Storcheus

Abstract

Learning distance metrics from data is a fundamental problem in machine learning and useful way to extract data-driven features by using the matrix root of a distance matrix. Finding a proper metric amounts to optimization over the cone of positive definite (PD) matrices. This optimization is difficult since restricting optimization to remain within the PD cone or repeatedly projecting to the cone is prohibitively costly.

Here we describe COMET, a block-coordinate descent procedure, which efficiently keeps the search within the PD cone, avoiding both costly projections and unnecessary computation of full gradients. COMET also continuously maintains the Cholesky root of the matrix, providing feature extraction and embedding of samples in a metric space. We further develop a structurally sparse variant of COMET, where only a small number of features interacts with other features. Sparse-COMET significantly accelerates both training and inference while improving interpretability.

As a block-coordinate descent procedure, COMET has fast convergence bounds showing linear convergence with high probability. When tested on benchmark datasets in a task of retrieving similar images and similar text documents, COMET has significantly better precision than competing projection-free methods. Furthermore, sparse-COMET achieves almost identical precision as dense-COMET in document classification, while running $\times 4.5$ faster, maintaining a 0.5% sparsity level, and outperforming competing methods both in precision and in run time.

Keywords: Feature extraction, Metric learning, Proximal sparse methods, Positive definite matrix

1. Introduction

Metric learning, learning a measure of pairwise distance among data samples, is an approach for extracting features in a data-driven way and a fundamental task in machine learning. Learned metrics can be used to project data into a new feature space, providing a representation for supervised learning techniques based on distances such as nearest-neighbors or kernel methods (Kulis, 2012). Learned metrics can also be used for ranking samples similar to a query sample, like finding similar images, or recommend online content to a user visiting a webpage.

Metric learning is tightly related to convex feature extraction, for the following reason: Learning a metric is often cast as solving a convex optimization problem over the cone of positive definite (PD) matrices by optimizing a similarity measure $sim_W(x, y) = x^\top W y$ such that $W \in \mathbb{R}^{d \times d}$ is a PD matrix (Kulis, 2012; Bellet et al., 2014). When W is PD, it can be factored as $W = L^\mathsf{T} L$, and L

can be used to map any data sample x to a new feature space Lx. Unfortunately, enforcing the PD constraint has a high computational cost. Projections to the PD cone require an eigendecomposition which is cubic in the number of features, or restricting optimization to the PD cone which is hard to perform efficiently. As a result, metric learning has been limited to mid-sized problems, and finding efficient optimization algorithms for metric learning is an ongoing challenge.

A major opportunity to speed metric learning and inference lies in the sparsity structure of the learned metric. At the extreme, enforcing the matrix to be diagonal could speed learning and inference dramatically, but it would ignore all interactions among feature pairs and does not extract any new features from data. Other sparse approaches focus on limiting the number of off-diagonal entries (Liu et al., 2014), and achieve state-of-the-art results in some sparsity regimes. Another natural sparsity structure that was not explored before, is the case were some features are limited to the diagonal, while another small set of features, determined from data, have off-diagonal terms. We show here that this structure can be learned very efficiently, and that it dramatically speeds learning and inference while the added sparsity constrains hardly hurt the quality of the learned metric.

This paper describe two new efficient algorithms for learning PD metrics called *COordinate-descent METric learning* (COMET), introducing a cost-effective method for performing block-coordinate descent over PD matrices. The first algorithm, **dense-COMET**, learns a dense PD matrix efficiently, while limiting optimization within the PD cone. The second algorithm, **sparse-COMET**, further introduces a new sparsity structure and uses it to speed training and inference. Both COMET algorithms efficiently optimize standard metric learning loss functions, while maintaining a PD matrix model continuously during learning, with no need for eigendecomposition. The algorithms can be applied efficiently to any smooth and convex objective function on PD matrices which decomposes over the matrix rows or columns.

COMET operates by updating the learned matrix one column and row at a time, thus updating the terms relating to one feature at each iteration. We use a $\log\det$ term as a barrier function for the PD cone, and employ the Schur complement to efficiently calculate an exact bound over the step size that guarantees that the model remains within the PD cone. Evaluations of COMET on benchmark datasets show that it outperforms other continuous-PD metric learning methods, and that sparse-COMET identifies highly informative features.

2. Related work

Learning distance metrics and metric similarity measures from data has been intensively studied, see Bellet et al. (2014); Kulis (2012) for recent surveys. This paper focuses on learning Mahalanobis distance matrices, where a major challenge is to efficiently enforce the PD matrix constraint during optimization. The simplest approach is to project the learned matrix onto the cone of PSD matrices, but this projection amounts to solving an eigendecomposition problem and is therefore costly (generally cubic in the feature dimensionality). Qian et al. (2014a,b) showed that the number of projections can be cleverly reduced, however each projection is still slow. A second common approach to enforcing PD is to learn a factored model $L^\mathsf{T} L$, but in that case the learning problem is no longer convex.

A second line of work avoids costly projections by keeping optimization within the PSD cone. Davis et al. (2007) and Jain et al. (2009) took this approach and introduced a $\log\det$ divergence term which acts as a log-barrier regularizer term. Another type of projection-free methods views a PSD matrix as a combination of other simpler PSD matrices. HDSL (Liu et al., 2014) learns a PSD matrix as a weighted combination of rank-1 sparse PSD update matrices, which are all zeros except

for a 2×2 entry corresponding to a pair of feature. BoostMetric (Shen et al., 2009) learns the metric matrix using rank-1 (PSD) updates which are generated by a boosting-based process. See also Bi et al. (2011); Liu and Vemuri (2012). Sparse Metric Learning (Ying et al., 2009) learns a low-rank matrix regularized with the squared *trace-norm*, where solutions are not necessarily sparse.

Closely related to our approach, is the work of Wen et al. (2009) who used row-column updates in the context of semi definite programming (SDP). Our approach further derives an explicit bound on the step size for the gradient step and applies it to a proximal step procedure. We also provide convergence analysis of the overlapping steps, and address the practical issue of preventing the matrix solutions from becoming ill-posed when too close to the PD-cone boundary.

3. The learning setup

We address the problem of learning a metric over a set of entities such as images or text documents, based on their relative pairwise similarities. Formally, let $\mathcal{P}$ be a set of entities $\{\mathbf{p}_1, ..., \mathbf{p}_N\}$ each represented as a vector in $\mathbb{R}^d$. We measure the similarity of two samples $\mathbf{q}, \mathbf{p} \in \mathcal{P}$ using a bilinear form parametrized by a model $W \in \mathbb{R}^{d \times d}$, $S_W(\mathbf{q}, \mathbf{p}) = \mathbf{q}^\mathsf{T} W \mathbf{p}$. When the matrix W is PSD, it can be factored as $W = L^\mathsf{T} L$ and used to define a similarity measure over pairs of data points. Specifically, the similarity $x^\top W y$ between two data points x and y through the matrix W, is equivalent to an Euclidean inner product $(Lx)^\top (Ly)$ in the transformed space $x \mapsto Lx$.

We assume that a weak form of supervision is given in the form of a ranking over triplets (Weinberger et al., 2006; Chechik et al., 2010; Qian et al., 2014a). Such ranking supervision is often easy to obtain and has widely achieved good performance. We assume we have access to triplets of entities from $\mathcal{P}$, where each triplet t consists of a "query" instance $\mathbf{q}_t \in \mathcal{P}$, and two instance ${\mathbf{p}_t}^+, {\mathbf{p}_t}^- \in \mathcal{P}$ such that $\mathbf{q}_t$ is more similar to ${\mathbf{p}_t}^+$ than to ${\mathbf{p}_t}^-$.

We aim to find a similarity measure S_W that agrees with the ranking of these triplets, namely, $S_W(\mathbf{q}, \mathbf{p}^+) > S_W(\mathbf{q}, \mathbf{p}^-)$. To achieve this, we use one of the following triplet loss functions

$$
\begin{aligned}
l^h_W(\mathbf{q}_t, {\mathbf{p}_t}^+, {\mathbf{p}_t}^-) &= [1 - {\mathbf{q}_t}^\mathsf{T} W {\mathbf{p}_t}^+ + {\mathbf{q}_t}^\mathsf{T} W {\mathbf{p}_t}^-]_+ \\
l^{hs}_W(\mathbf{q}_t, {\mathbf{p}_t}^+, {\mathbf{p}_t}^-) &= [1 - {\mathbf{q}_t}^\mathsf{T} W {\mathbf{p}_t}^+ + {\mathbf{q}_t}^\mathsf{T} W {\mathbf{p}_t}^-]^2_+ \\
l^{log}_W(\mathbf{q}_t, {\mathbf{p}_t}^+, {\mathbf{p}_t}^-) &= log(1 + exp(-{\mathbf{q}_t}^\mathsf{T} W {\mathbf{p}_t}^+ + {\mathbf{q}_t}^\mathsf{T} W {\mathbf{p}_t}^-)),
\end{aligned}
\tag{1}
$$

where $[z]_+ \doteq \max(0, z)$. Given a batch of $\mathcal{T}$ triplets, adding a Frobenius regularization term, and a log-barrier term, we aim to solve the following regularized optimization problem

$$
L(W) = \min_W \sum_{t \in \mathcal{T}} l_W(\mathbf{q}_t, {\mathbf{p}_t}^+, {\mathbf{p}_t}^-) - \alpha \log\det(W) + \frac{\beta}{2} \|W\|^2_F, \tag{2}
$$

where l_W is any of the triplet loss functions and $\|W\|^2_F$ is the squared Frobenius norm of the matrix W. This optimization problem is convex in W. The log-barrier term $\log\det(W)$ ensures that the optimum is within the PD cone. The method we introduce in Section 4.1 guarantees that all iterates remain within the PD cone even without the log-barrier term, but we found that term to contribute to empirical performance and to numerical stability, especially when the optimum is near the PSD cone edge.

The loss in Eq. (2) can be minimized using gradient descent (GD), yielding

$$
\frac{\partial L(W)}{\partial W} = \sum_{t \in \mathcal{T}} \{[\tfrac{1}{2}[\mathbf{q}_t \Delta {\mathbf{p}_t}^\mathsf{T} + \Delta \mathbf{p}_t {\mathbf{q}_t}^\mathsf{T}] l'(\mathbf{q}_t, {\mathbf{p}_t}^+, {\mathbf{p}_t}^-)\} - \alpha W^{-1} + \beta W, \tag{3}
$$

where $\Delta\mathbf{p}_t = \mathbf{p}_t^- - \mathbf{p}_t^+$, and $l'(x) \doteq \frac{dl(x)}{dx}$ is the outer derivative of the loss function (Appendix A).

4. Learning a PD metric with block-coordinate descent: Dense COMET

The learning setup above is commonly studied, but optimizing it using gradient descent is computationally hard, because the $\log\det$ term yields a W^{-1} term in the gradient of Eq. (3). This term makes naive implementations with matrix inversion slow, scaling cubically with the matrix dimension. Another difficulty is that while in theory the $\log\det$ term ensures that the optimum is within the PD cone, in practice the intermediate iterates for first-order methods are not necessarily confined to the cone. The reason is that the gradients of the $\log\det$ term are not Lipschitz and any fixed step size might take the update outside the cone. Forcing the objective to be Lipschitz by adding a small constant term to the diagonal of W, would still require the step size to be small to ensure remaining within the cone, leading to slower convergence.

We propose an algorithm that alleviates these problems by using efficient block-coordinate descent that keeps optimization within the PD cone while amortizing the cost of matrix inversion. In general, block-coordinate descent enjoys provable fast convergence rates, and is especially useful when updating a block is efficient, as we show below. We derive a method that enables using a step size which is as large as possible while still remaining within the PD cone for all iterates, based on the Schur complement condition for PD matrices (Boyd and Vandenberghe, 2004, p. 650).

We perform the block updates as follows: Draw a feature $k \in \{1 \dots d\}$ at random, and define a matrix G that is all zeros except the values of $-\frac{\partial \tilde{L}(W)}{W}$ on the k-th row and k-th column. Then, update $W^{new} = W + \eta G$, using a step size η that is computed analytically to keep the iterates within the PD cone (see subsection 4.1). For a fast access to the $\log\det$ derivative elements, we also hold $(W^{new})^{-1}$ in memory. In each step, we update $(W^{new})^{-1}$ based on W^{-1} and using the Woodbury inverse matrix identity (Woodbury, 1950). This costs $O(d^2)$, see Appendix B for details.

Algorithm 1 Dense COMET

1: **input:** training data, α, β
2: **initialize:** Generate a triplet set $\mathcal{T}$. Set $W \leftarrow I_d$. Set $W^{-1} \leftarrow I_d$.
3: **repeat**
4: Draw a coordinate $k \in \{1 \dots d\}$ uniformly at random.
5: Compute the coordinate step gradient G (Eq. (3)).
6: Select the step size η using the upper limit from Eq. (11) (See Appendix B).
7: Update the metric $W^{new} = W + \eta G$, and W^{new-1} (see Appendix B).
8: **until** stopping condition

4.1 Selecting a step size that guarantees remaining within the PD cone

Taking a coordinate step may take W^{new} out of the PD cone. In Appendix B, we show that given W and G, the step size can be analytically bounded to guarantee that W^{new} remains PD. We also show the bound can be computed efficiently using the Schur complement condition for positive definiteness. In turn, the condition reduces to a scalar inequality for the case of row-column update:

$$W^{new} \succ 0 \quad \Leftrightarrow \quad c^* - \mathbf{b}^{*\mathsf{T}} A^{-1} \mathbf{b}^* > 0, \tag{4}$$

where w.l.g. we assume we updated the first row-column, and $c^* = W^{new}_{1,1} \in \mathbb{R}$ (a scalar), $\mathbf{b}^* = W^{new}_{2:d,1} \in \mathbb{R}^{d-1}$ (a column vector) and $A^* = {W^{new}}_{2:d,2:d} = W_{2:d,2:d} \in \mathbb{R}^{(d-1)}$. The condition

Algorithm 2 Sparse COMET

input: training data, α, β, λ, $\bar{\theta}$
initialize: Generate a triplet set $\mathcal{T}$. Set $W \leftarrow I_d$, $W^{-1} \leftarrow I_d$, $V_0 \leftarrow I_d$, $\{V_k\}_{k=1}^{d} \leftarrow \mathbf{0}$.
Optimize the diagonal of W using Eq. (3). Update W, W^{-1}, $V_0 \leftarrow W$
repeat
  Select a coordinate $k \in \{1 \dots d\}$ uniformly at random.
  Compute the coordinate step gradient G according to Eq. (3).
  Solve the proximal problem Eq. (8) with maximal step size $\bar{\theta}$ and λ sparsity coefficient.
  if ${V_k}^{new} == V_k$ **then**
    Continue to next iteration.
  else
    Select new step size θ, obeying the upper limit from Appendix E.
    Solve the proximal problem Eq. (8) with step size θ and sparsity coefficient λ.
    Update the decomposition $\{V_k\}_{k=1}^{d}$ with ${V_k}^{new}$ (see Eq. (8)).
    Update the metric $W^{new} = {V_k}^{new} + W - V_k$ and the inverse W^{new-1} (see Appendix B).
  end if
until stopping condition

in Eq. (4) induces a condition over the step size η, which gives us an upper limit to the allowable step size of a block coordinate (row-column) step over the gradient update Eq. (3). Respecting the upper limit guarantees that the updated matrix W^{new} will be PD. The computational complexity for calculating the upper limit is $O(d^2)$, as detailed in Appendix B. In practice we use a Cholesky solver (Chen et al., 2008) to speed up training and maintain the matrix square root. See details in Appendices B and C. Maintaining the Cholesky root matrix provides an embedding of the features to the metric space continuously during training. This approach is summarized in Algorithm 1.

Convergence rate. COMET is based on minimizing a strongly-convex function using block-coordinate descent. Since blocks in our method are partially overlapping, we use a convergence result by Richtárik and Takáč (2013) to analyze the convergence of the algorithm, and show that Algorithm 1 converges with high probability to the optimum value in a linear rate. The analysis holds for the squared hinge-loss and the log-loss, but not for the hinge-loss which is not smooth. See Theorem 1 in Appendix D for the detailed claim and proof.

5. Learning a block-sparse PD metric: Sparse COMET

In most high-dimensional learning problems, many feature pairs do not interact intensely with other features, hence their corresponding off-diagonal terms are (close to) zero. The PD metrics learned above fail to take into account such structure, leading to "wasteful" inference that costs $O(d^2)$ extra computation. To benefit from such structure, we propose to enforce a new type of structured sparsity, which can be optimized efficiently. Specifically, we suggest to allow only a small set of features to interact with *any* of the other features, and eliminate the interaction term for any two features which are not in the "interacting set". Importantly, we also maintain weights for the individual features, corresponding to the diagonal of the learned similarity matrix. This interaction structure, illustrated in Figure 3a, leads to sparse matrices and allows faster inference both during training and test time.

We now describe a method for learning PD metrics with this interaction structure, using a block-coordinate descent method that maintains the PD property throughout training. In the metric learning setting, each feature corresponds to a row-column block which overlaps with all the other row-column blocks. For example, the first row-column of the matrix W, corresponding to the first feature, intersects the i-th row-column at entries W_{1i} and W_{i1}. We therefore use an *overlapping decomposition* of W into blocks (Jacob et al., 2009; Obozinski et al., 2011). Specifically, we decompose the matrix W into $d+1$ group-components matrices $\{V_k\}_{k=0}^d$. The matrix V_0 is a diagonal matrix, and each matrix V_k ($k > 0$) is a symmetric matrix of non-zero values only on the k^{th} row and column, with an all-zeros diagonal. Finally, we define W as the sum $W = \sum_{k=0}^d V_k$. Given this definition of W, the loss from Eq. (2) can be expressed as a function of the groups $\{V_k\}_{k=0}^d$:

$$L(\{V_k\}_{k=0}^d) = \min_{\{V_k\}_{k=0}^d} \sum_{t\in\mathcal{T}} l_W(\mathbf{q}_t, \mathbf{p}_t{}^+, \mathbf{p}_t{}^-) - \alpha \log\det(\sum_{k=0}^d V_k). \tag{5}$$

The gradient of this loss w.r.t. an element of V_k is therefore:

$$\frac{\partial L(V_k)}{\partial v_{ki}} = \sum_{t\in\mathcal{T}} \{[\tfrac{1}{2}[\mathbf{q}_t \Delta\mathbf{p}_t{}^\mathsf{T} + \Delta\mathbf{p}_t \mathbf{q}_t{}^\mathsf{T}]_{k,i} l'_W(\mathbf{q}_t, \mathbf{p}_t{}^+, \mathbf{p}_t{}^-)\} - \alpha[W^{-1}]_{k,i} \quad , \tag{6}$$

where for $k \geq 1$, the gradient $\frac{\partial L(V_k)}{\partial v_{li}}$ vanishes if $l \neq k \wedge i \neq k$, or $l = i$. Computing the gradient term of Eq. (6) requires computing W^{-1}, which we maintain and update as in dense-COMET. Adding a group-sparse norm penalty to the loss to encourage solutions with fewer features, we obtain:

$$L(\{V_k\}_{k=0}^d) = \min_{\{V_k\}_{k=0}^d} \sum_{t\in\mathcal{T}} l_W(\mathbf{q}_t, \mathbf{p}_t{}^+, \mathbf{p}_t{}^-) - \alpha \log\det(\sum_{k=0}^d V_k) + \lambda \sum_{k=1}^d \|V_k\|_F \quad . \tag{7}$$

One can also add an L_2 or L_2^2 regularization factor to the diagonal group V_0. The regularized loss in Eq. (7) is a convex function of $\{V_k\}_{k=0}^d$, since the negative $\log\det$ term is a composition of a convex function with a linear function, and therefore convex. The group sparsity term $\lambda \sum_{i=1}^d \|V_k\|_F$ encourages some of the groups V_k, $1 \leq k \leq d$ to be 0. Since the groups V_k control the off-diagonal terms of W, any pair $i \neq j$ for which $V_i = 0, V_j = 0$, has the corresponding off-diagonal elements W_{ij} equal to 0. Given the decomposition $\{V_k\}_{k=0}^d$, we can Eq. (7) using standard block-coordinate methods for non-smooth objectives, using the method of Richtárik and Takáč (2014).

Let $\mathcal{V}_k$ denote the set of $d \times d$ matrices which are all zero except the non-diagonal entries on the k-th row and column. At each block update, we solve the following proximal problem, which admits a closed form solution (Bach et al., 2012) (See appendix E):

$$V_k^{new} = \arg\min_{V\in\mathcal{V}_k} \left\langle \frac{\partial L(V_k)}{\partial V_k}, V \right\rangle + \frac{1}{2\theta}\|V - V_k\|_F^2 + \lambda\|V\|_F, \tag{8}$$

and set $W^{new} = W + V_k^{new} - V_k$, where θ corresponds to the step size of the proximal update.

The proximal update of Eq. (8) maintains many of the groups V_k as identically zero. This leads to a sparse update schedule, since a group that is 0 often remains 0, saving the computation of the PD bound and of updating W. Therefore sparse COMET reduces the mean cost per step to $O(\rho d^2)$, where ρ is the group sparsity of $\{V_k\}_{k=1}^d$. The method is summarized in Algorithm 2.

METHOD:	D/SPCOMET	SGD+PROJECT.	HDSL	LEGO
COMPUTATIONAL COMPLEXITY.	$O(\gamma^2 d^2 T + \rho d^3)$	$O(\gamma^2 d^2 T + \frac{T}{P} \cdot d^3)$	$O(T \cdot d^4)$	$O(d^2 \cdot T)$

Table 1: Asymptotic computational complexity per one pass over all triplets and coordinates, comparing COMET, SGD-based methods (Chechik et al., 2010; Qian et al., 2014a), HDSL (Liu et al., 2014) and LEGO (Jain et al., 2009). In our experiments, HDSL converged before going over all coordinates, but then typically achieves significantly inferior test performance. COMET has lower complexity than SGD with multiple projections HDSL. COMET also has lower complexity than LEGO if $T(1-\gamma^2) > d$, that is, when the data is even moderately sparse and the number of triplets is larger than the number of features. **Notation:** T: number of constraints (triplets). d: the dimension. $0 < \gamma \leq 1$: data sparsity, often is $O(1/\sqrt{d})$. P: size of triplet batch between PSD projections for SGD-based methods; (Qian et al., 2014a) used $T/P = 0.1T$.

Convergence. Similar to the case of dense-COMET, the objective $L(\{V_k\}_{k=0}^d)$ in Eq. (5) is strongly convex but not smooth. Let $\tilde{L}(\{V_k\}_{k=0}^d) = L(\{\kappa I_d + V_0\} \cup \{V_k\}_{k=1}^d)$ be a modified version of the loss in Eq. (7), where I_d is the $d \times d$ identity matrix, and $\kappa > 0$ is a fixed parameter. As shown in (Richtárik and Takáč, 2014, Theorem 7), the proximal block coordinate descent we use converges w.h.p. linearly to the optimal value, given that the maximal step size θ_{max} is smaller than the block-Lipschitz constants of $\tilde{L}(\{V_k\}_{k=0}^d)$. As with dense-COMET, the block-Lipschitz constants for the proximal step size provide a theoretical convergence guarantee which is very conservative, yet ensures every iterate W^{new} remains within the PD cone.

In practice, faster convergence can be achieved by taking larger step sizes, but large non-adaptive steps may yield a W^{new} that is no longer PD. The approach we take is similar to the dense case, albeit slightly more involved since the proximal step lies in the span of two vectors. Details are provided in Appendix E. Evaluating the step size bound costs $O(d^2)$;

6. Computational complexity

Table 1 summarizes the asymptotic computational complexity of COMET and several competing methods. COMET has lower complexity than SGD-based methods that require repeated projections (Chechik et al., 2010; Qian et al., 2014a), and than HDSL (Liu et al., 2014). In the regime of many samples and a moderate-to-high data sparsity, COMET has lower complexity than LEGO (Jain et al., 2009). See Appendix C for detailed derivation.

7. Experiments

We evaluate COMET on three datasets and compare it with four metric-learning approaches.

7.1 Competing approaches

We compare the two variants of COMET with 4 approaches that learn a Mahalanobis metric matrix while avoiding repeated projections to the PD cone.

(1) COMET. Dense-COMET as described in Algorithm 1 and sparse-COMET in Algorithm 2, both using linear hinge-loss for the triplets loss, see Eq. (2). **(2) Euclidean**. The Euclidean similarity in the original feature space. COMET is initialized using the identity matrix, which is equivalent to this similarity measure. **(3) HDSL** (Liu et al., 2014). A Frank-Wolfe based approach tuned for high-dimensional sparse data. HDSL learns a convex combination of rank-1 PSD matrices that are all zeros except for a 2×2 pair of features elements. It iteratively adds these matrices, one feature-pair at a time, to control the number of active features. **(4) LEGO** (Jain et al., 2009). This approach

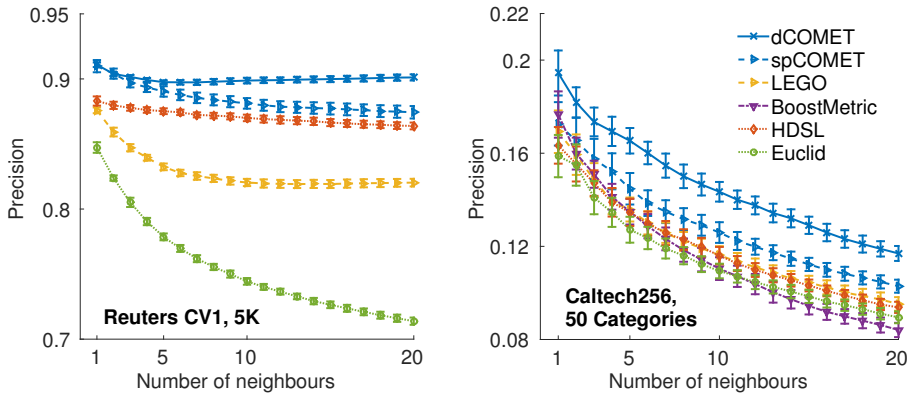

Figure 1: Precision at the top-k nearest neighbors, evaluated on the test set and averaged over five train/test cross validation folds (80%/20%). Error bars denote the standard error of the means across 5 folds.

uses a $\log\det$ divergence term to enforce the PD constraint. While the main variant of LEGO aims to fit pairwise distances, we used a variant of LEGO that, like COMET, learns from relative similarities. Loss is incurred for same-class samples farther than a certain distance, and different-class samples closer than a certain distance. We profiled and optimized the provided LEGO source code to decrease its run time, and made it significantly faster in large scale data scenarios. LEGO uses pairs constraints and not triplets, therefore we used approximately $\times 2$ constraints to train LEGO per each dataset. We chose a number of pairs constraints that demonstrates convergence for LEGO precision on the validation set. **(5) BoostMetric** (Shen et al., 2009). Based on the observation that any positive semidefinite matrix can be decomposed into linear positive combination of rank-1 matrices, BoostMetric uses rank-1 PSD matrices as weak learners.

7.2 Datasets

We evaluate COMET on three benchmark datasets. **(1) REUTERS CV1** is a collection of English text documents. We used the 4-class subset introduced in (Cai and He, 2012) and used in (Liu et al., 2014), and selected subsets of 5000 and 1000 features using the *infogain* criterion (Yang and Pedersen, 1997). Each document was represented as a bag of words, where weights of selected features were *tf-idf* transformed. The sparsity of this dataset, after selecting top 5000 features, is 1.3%. As in Liu et al. (2014), we used 100,000 triplets for training. To train HDSL, we took 8000 iterations as in (Liu et al., 2014). BoostMetric could not converge on this dataset due to memory and runtime issues caused by the large number of features. **(2) CALTECH256** is a dataset of labeled images for visual object recognition. We used the subsets of 50 and 249 classes tested for metric learning in (Chechik et al., 2010). This sets contain 65 images per class (total of 3250 images), represented with 1000 *bag-of-local-descriptors* features provided by these authors. The sparsity of this dataset is 3.3%. We selected the number of triplets based on early-stopping of the OASIS model, resulting in 135,000 triplets. Number of HDSL training iterations was selected using early stopping on a validation set. BoostMetric was slow on this dataset, and used a large amount of memory. For a fair comparison, we took the number of COMET coordinate steps to be the maximal number of BoostMetric rank-1 updates. **(3) PROTEIN** is a dense dataset with 3 classes, 357 features and 24387 samples (Chang and Lin, 2011), which was recently tested for metric learning in (Qian et al., 2014a). We used 20,000 triplets for training.

7.3 Experimental setup and evaluation measures

In all datasets, two samples are considered similar if they share the same class label. Each data set is tested on a two-layer 5 fold cross validation experiment. We use the same (frozen) random splits

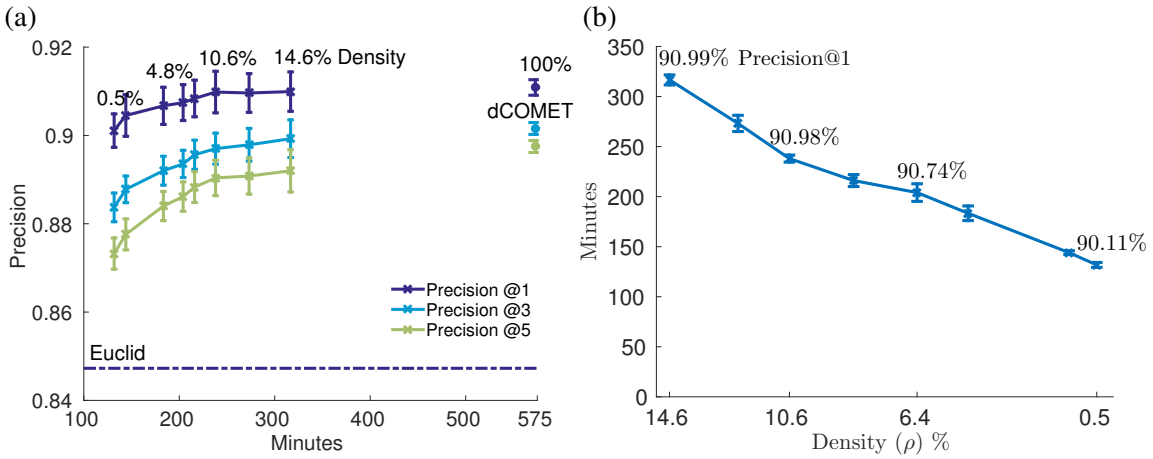

Figure 2: The effect of sparsity on sparse-COMET precision and runtime. RCV1 dataset with 5K features. **(a)** Precision at 1, 3 and 5 nearest neighbor evaluated on the test set *vs.* the mean training run time of COMET. Percentiles denote the density of the learned matrix. Error bars denote the standard error of the mean across 5 random train/test partitions (80%/20%). Dashed line denotes the *precision-at-1* of the Euclidean baseline. **(b)** Mean training time as a function of the learned matrix density. Percentiles denote the *Precision-at-1*.

across all approaches. Training all learners with the exact same set of triplets, except for LEGO that uses pairs constraints. We verified that we choose the triplets/constraints number in a regime such that test performance converges (figures not shown due to space constraints). We generated triplets randomly while keeping a fixed number of triplets per query sample.

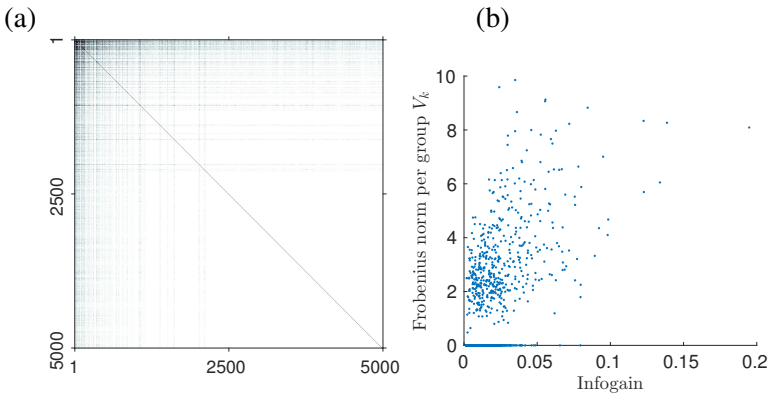

Figure 3: Structured sparsity and extracted features. **(a)** A heat map of the absolute values of the elements of W trained on RCV1, illustrating the structured sparseness of the learned metric. Features are ordered by their information gain. **(b)** Frob. norm of the groups V_k against the information gain of feature k. Sparse COMET assigns zero weights to less-informative features.

We evaluated the performance of all algorithms using standard ranking precision measures based on nearest neighbors. For each query instance in the test set, all other test instances were ranked according to their similarity to the query instance. The number of same-class instances among the top k instances (the k-nearest-neighbors) was computed, and averaged to yields the *precision-at-k*, providing a precision curve as a function of the rank k. *Precision-at-k* for k >1 is useful in retrieval of multiple objects that are similar to a query object, as done in image search engines.

7.4 Results

We evaluate the learned metric in a setup of ranking samples by their distance from test examples, and evaluating the fraction of nearest neighbors that belong to the same class as the test sample. We

DATASET	DCOMET	SPCOMET		HDSL	LEGO
REUTERS CV1 (5K FEATURES)	573 ± 22	238 ± 8	$\rho = 10.6\%$	522 ± 24	423 ± 29
		132 ± 5	$\rho = 0.5\%$		
CALTECH256 50 CAT. (1K F.)	32 ± 2	25 ± 1	$\rho = 20.0\%$	495 ± 73	15 ± 3
CALTECH256 249 CAT. (1K F.)	90 ± 9	44 ± 2	$\rho = 24.5\%$	495 ± 39	20 ± 3
REUTERS CV1 (1K FEATURES)	53 ± 3	25 ± 1	$\rho = 24.7\%$	115 ± 18	11 ± 3
PROTEIN (357 FEATURES)	6.1 ± 0.5	15 ± 0.3	$\rho = 10.3\%$	163 ± 11	0.5 ± 0.1

Table 2: Run-time, minutes. $\pm$ denotes the standard deviation. For spCOMET, we selected ρ values to illustrate the performance gain. These are the ρ values used in Figures 1,3,4. For other ρ values, see Figure 2.

report the *precision-at-k* on test set as a function of k neighbours, averaged across 5 random train/test partitions (80%/20%). Hyperparameters were tuned using a second layer of cross validation. We discuss here results for RCV1 5000 features and Caltech256 50 categories, and discuss the other datasets in Appendix F. We observed that convergence is usually achieved after $6 \cdot d$ to $8 \cdot d$ coordinate steps for dense COMET and $8 \cdot d$ to $11 \cdot d$ with sparse COMET. We note that instead of random sampling with replacement we used a random permutation of the coordinates $\{1, \ldots, d\}$ for each epoch, as previous work indicated sampling without replacement leads to faster convergence. Surprisingly, when tuning hyper-parameter values, we found that the Frobenius regularizer obtained very small weights, and setting its coefficient to zero did not harm performance.

Figure 1 compares the precision obtained by sparse and dense COMET with the four competing approaches described above. Both sparse and dense COMET achieved consistently superior or equal precision throughout the full range of number of neighbours tested. Figure 2 shows the precision of sparse COMET on RCV1 over several sparsity levels, and compares it with dense COMET and the Euclidean baseline. It demonstrates that sparse-COMET achieves 99% of the nearest neighbor precision of dense COMET, while cutting training time by $\times 4.5$ and maintaining a 0.5% sparsity level. Moreover, even these highly sparse solutions outperform competing methods in terms of precision and run time.

We further compared dense-COMET with OASIS, a method achieving state-of-the-art precision on the Caltech-256 dataset. Dense-COMET precision is nearly identical to that of OASIS, even though OASIS solutions are not limited to the PSD cone. This is likely because both methods essentially optimize a similar objective. Table 2 compares the run times of the competing approaches. BoostMetric results were partial hence not shown. Sparse COMET is fastest on the RCV1 5K features dataset. LEGO is fastest on the smaller datasets; HDSL is mostly slower than COMET. Importantly, both sparse and dense COMET converged to a significantly better optimum.

8. Summary

We presented COMET, a new metric learning approach that avoids both costly projections and unnecessary computation of full gradients. It continuously maintains the PD property, and allows feature extraction and embedding of samples in a metric space throughout the training. We also introduced a new form of structured sparsity, where only a small number of features can interact with other features. Sparse-COMET significantly accelerates both training and inference, maintains interpretability and outperforms competing methods. COMET source code is available for download at `chechiklab.biu.ac.il/~yuvval/COMET/`

Acknowledgements: We thank Prof. Tim Davis for insightful discussions and assistance with Cholesky decomposition for row-column updates.

References

Francis Bach, Rodolphe Jenatton, Julien Mairal, and Guillaume Obozinski. Optimization with sparsity-inducing penalties. *Foundations and Trends® in Machine Learning*, 4(1):1–106, 2012.

Aurélien Bellet, Amaury Habrard, and Marc Sebban. A survey on metric learning for feature vectors and structured data. *arXiv preprint arXiv:1306.6709v4*, 2014.

Jinbo Bi, Dijia Wu, Le Lu, Meizhu Liu, Yimo Tao, and Matthias Wolf. Adaboost on low-rank psd matrices for metric learning. In *Computer Vision and Pattern Recognition (CVPR), 2011 IEEE Conference on*, pages 2617–2624. IEEE, 2011.

Stephen Boyd and Lieven Vandenberghe. *Convex optimization*. Cambridge university press, 2004.

Deng Cai and Xiaofei He. Manifold adaptive experimental design for text categorization. *Knowledge and Data Engineering, IEEE Transactions on*, 24(4):707–719, April 2012.

Chih-Chung Chang and Chih-Jen Lin. Libsvm: A library for support vector machines. acm transactions on intelligent systems and technology, 2: 27: 1–27: 27, 2011. *Software available at http://www. csie. ntu. edu. tw/cjlin/libsvm*, 2011.

Gal Chechik, Varun Sharma, Uri Shalit, and Samy Bengio. Large scale online learning of image similarity through ranking. *Journal of Machine Learning Research*, 11:1109–1135, 2010.

Yanqing Chen, Timothy A. Davis, and William W. Hager. Algorithm 887: Cholmod, supernodal sparse cholesky factorization and update/downdate. *ACM Transactions on Mathematical Software*, pages 1–14, 2008.

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

Timothy A. Davis, William, and William W. Hager. Row modifications of a sparse cholesky factorization. *SIAM J. Matrix Anal. Appl*, 26:997–1013, 2005.

Laurent Jacob, Guillaume Obozinski, and Jean-Philippe Vert. Group lasso with overlap and graph lasso. In *Proceedings of the 26th annual international conference on machine learning*, pages 433–440. ACM, 2009.

Prateek Jain, Brian Kulis, Inderjit S Dhillon, and Kristen Grauman. Online metric learning and fast similarity search. In *Advances in neural information processing systems*, pages 761–768, 2009.

Brian Kulis. Metric learning: A survey. *Foundations and Trends in Machine Learning*, 5(4):287–364, 2012.

Kuan Liu, Aurélien Bellet, and Fei Sha. Similarity learning for high-dimensional sparse data. *Proceedings of the Eighteenth International Conference on Artificial Intelligence and Statistics, pp. 653–662, 2015*, 2014.

Meizhu Liu and Baba C Vemuri. A robust and efficient doubly regularized metric learning approach. In *Computer Vision–ECCV 2012*, pages 646–659. Springer, 2012.

Yu Nesterov. Efficiency of coordinate descent methods on huge-scale optimization problems. *SIAM Journal on Optimization*, 22(2):341–362, 2012.

Guillaume Obozinski, Laurent Jacob, and Jean-Philippe Vert. Group lasso with overlaps: the latent group lasso approach. *arXiv preprint arXiv:1110.0413*, 2011.

Qi Qian, Rong Jin, Jinfeng Yi, Lijun Zhang, and Shenghuo Zhu. Efficient distance metric learning by adaptive sampling and mini-batch stochastic gradient descent (sgd). *Machine Learning*, pages 1–20, 2014a.

Qi Qian, Rong Jin, Shenghuo Zhu, and Yuanqing Lin. An integrated framework for high dimensional distance metric learning and its application to fine-grained visual categorization. *arXiv preprint arXiv:1402.0453*, 2014b.

Peter Richtárik and Martin Takáč. On optimal probabilities in stochastic coordinate descent methods. *arXiv preprint arXiv:1310.3438*, 2013.

Peter Richtárik and Martin Takáč. Iteration complexity of randomized block-coordinate descent methods for minimizing a composite function. *Mathematical Programming*, 144(1-2):1–38, 2014.

Chunhua Shen, Junae Kim, Lei Wang, and Anton Hengel. Positive semidefinite metric learning with boosting. In *Advances in neural information processing systems*, pages 1651–1659, 2009.

Kilian Q. Weinberger, John Blitzer, and Lawrence K. Saul. Distance metric learning for large margin nearest neighbor classification. 2006.

Zaiwen Wen, Donald Goldfarb, Shiqian Ma, and Katya Scheinberg. Row by row methods for semidefinite programming. Technical report, Dept of IEOR, Columbia University, 2009.

Max A Woodbury. Inverting modified matrices. *Memorandum report*, 42:106, 1950.

Yiming Yang and Jan O Pedersen. A comparative study on feature selection in text categorization. In *ICML*, volume 97, pages 412–420, 1997.

Yiming Ying, Kaizhu Huang, and Colin Campbell. Sparse metric learning via smooth optimization. In *Advances in neural information processing systems*, pages 2214–2222, 2009.

Appendix A. Gradient for a triplet

To compute the matrix gradient step $\frac{\partial l_t(W)}{\partial W}$ of an arbitrary triplet t, we denote the linear part of the hinge loss of a triplet t by $\lambda_W^t \doteq 1 - \mathbf{q}_t{}^\mathsf{T} W \mathbf{p}_t{}^+ + \mathbf{q}_t{}^\mathsf{T} W \mathbf{p}_t{}^-$.

W is PD and therefore symmetric. We enforce its gradient to be symmetric by replacing W with $\frac{1}{2}(W + W^\mathsf{T})$. The derivative of the ranking loss is then given by

$$\frac{\partial l_W^t}{\partial W} = \tfrac{1}{2}[\mathbf{q}_t \Delta \mathbf{p}_t{}^\mathsf{T} + \Delta \mathbf{p}_t \mathbf{q}_t{}^\mathsf{T}] \cdot l'(\lambda_W^t),$$

where $l'(x) \doteq \frac{dl(x)}{dx}$ is the outer derivative of the loss function, $\Delta \mathbf{p}_t \doteq (\mathbf{p}_t^- - \mathbf{p}_t^+)$.

Appendix B. Bounding the step size and updating the inverse matrix

Without loss of generality, assume that the coordinate round updates the first feature $k = 1$. We write the pre- and post-update matrices, as

$$W = \begin{bmatrix} c & \mathbf{b}^\mathsf{T} \\ \mathbf{b} & A \end{bmatrix}, \quad W^{new} = \begin{bmatrix} c^* & \mathbf{b}^{*\mathsf{T}} \\ \mathbf{b}^* & A^* \end{bmatrix}, \tag{9}$$

where $c = W_{1,1} \in \mathbb{R}$ (a scalar), $\mathbf{b} = W_{2:d,1} \in \mathbb{R}^{d-1}$ (a column vector) and $A = W_{2:d,2:d} \in \mathbb{R}^{(d-1)\times(d-1)}$. Similarly for A^*, $\mathbf{b}^*$ and c^*.

According to the Schur-complement condition for positive definiteness (Boyd and Vandenberghe, 2004, p. 650), W^{new} is PD iff both A^* and $c^* - \mathbf{b}^{*\mathsf{T}} A^{*-1} \mathbf{b}^*$ are positive definite. Since $W \succ 0$ and A is a minor of W which is left unchanged by the update, we have $A^* = A \succ 0$. Moreover, $c^* - \mathbf{b}^{*\mathsf{T}} A^{*-1} \mathbf{b}^*$ is a scalar, yielding

$$W^{new} \succ 0 \quad \Leftrightarrow \quad c^* - \mathbf{b}^{*\mathsf{T}} A^{-1} \mathbf{b}^* > 0. \tag{10}$$

Now let $u_1 = c^* - c$ and $\mathbf{u}_{2:d} = \mathbf{b}^* - \mathbf{b}$ be the updated scalar and vector obtained from $\eta G = W^{new} - W$. We expand Eq. (10) and Eq. (3) (with $k = 1$) yielding a necessary and sufficient condition for $W^{new} \succ 0$: $(W_{1,1} + 2\eta u_1) - (W_{2:d,1} + \eta \mathbf{u}_{2:d})^\mathsf{T} A^{-1} (W_{2:d,1} + \eta \mathbf{u}_{2:d}) > 0$. Grouping as a quadratic inequality in η, and using the notation from Eq. (9) we end up with

$$(\mathbf{u}_{2:d}{}^\mathsf{T} A^{-1} \mathbf{u}_{2:d})\, \eta^2 - 2(u_1 - \mathbf{u}_{2:d}{}^\mathsf{T} A^{-1} \mathbf{b})\, \eta - (c - \mathbf{b}^\mathsf{T} A^{-1} \mathbf{b}) < 0. \tag{11}$$

For $\eta = 0$, this inequality always holds since $W \succ 0$ guarantees that $c - \mathbf{b}^\mathsf{T} A^{-1} \mathbf{b} > 0$. As a result, and since A^{-1} is PD, Eq. (11) always has a real root $\eta > 0$. This root provides an upper bound on η that guarantees that W^{new} is PD. Computing the coefficients involves computing bilinear terms and costs $O(d^2)$ if A^{-1} is given.

In practice, evaluating the condition of Eq. (11) efficiently requires that we maintain an updated Cholesky root matrix L such that $W = L^\mathsf{T} L$. It enables us to efficiently derive a Cholseky root for A (Davis et al., 2005) and in turn to efficiently evaluate the terms of Eq. (11) with a cost of $O(d^2)$.

To evaluate the $\log\det$ gradient, the new inverse W^{new-1} can be easily computed using the Woodbury matrix identity (Woodbury, 1950). We rewrite the update and Eq. (3), using $W^{new} = W + \eta G = W + \widetilde{G}$ and write

$$\widetilde{G} = UCV = \begin{bmatrix} \mathbf{u} & \mathbf{e_k} \end{bmatrix} \begin{bmatrix} \eta & 0 \\ 0 & \eta \end{bmatrix} \begin{bmatrix} \mathbf{e_k}{}^\mathsf{T} \\ \mathbf{u}^\mathsf{T} \end{bmatrix},$$

where $\mathbf{u}$ is a column vector that equals the column k of the gradient matrix of the objective Eq. (3), $\mathbf{e_k}$ equals an elementary vector for selecting a column k of a matrix. Using the Woodbury matrix identity gives

$$W^{new\,-1} = W^{-1} - W^{-1}U(\eta^{-1}I_2 + VW^{-1}U)^{-1}VW^{-1} \quad .$$

Appendix C. Analysis of computational complexity

We first evaluate the computational complexity of a single coordinate step of Eq. (3), which includes computing the gradient and updating W, W^{-1} and the Cholesky decomposition of W.

Consider first the computation of the gradient. For the hinge-loss case l_W^h, each element $\delta_{i,j}$ of the gradient matrix in Eq. (3) equals

$$\delta_{(i,j)} = \sum_{t\in\mathcal{T}} [\tfrac{1}{2}[(\mathbf{q}_t)_i(\Delta\mathbf{p}_t{}^{\mathsf{T}})_j + (\Delta\mathbf{p}_t{}^{\mathsf{T}})_i(\mathbf{q}_t)_j{}^{\mathsf{T}}] \cdot \mathbf{1}(\lambda_W^t) - \alpha \cdot W_{i,j}^{-1} + \beta \cdot W_{i,j}, \tag{12}$$

where $\lambda_W^t \doteq 1 + \mathbf{q}_t{}^{\mathsf{T}} W \Delta\mathbf{p}_t$ is the linear part a triplet loss

For dense data, evaluating the sum over T triplets costs $O(T)$ operations. However, for $gamma$-sparse data with a sparsity coefficient $0 < \gamma < 1$, this cost can be reduced to $O(\gamma^2 T)$ operations on average, by accumulating only elements that are non-zero both in $(\mathbf{q}_t)_i$ and in $(\Delta\mathbf{p}_t{}^{\mathsf{T}})_j$ and likewise for $(\mathbf{q}_t)_j$ and $(\Delta\mathbf{p}_t{}^{\mathsf{T}})_i$. To efficiently evaluate the indicator functions $\{\mathbf{1}(\lambda_W^t)\}_{t\in T}$ on Eq. (12), we keep an array of the linear terms $\{\lambda_W^t\}_{t\in T}$. Computing all the gradient elements $\delta_{(k,1:d)}$ in a single row k costs $O(d \cdot \gamma^2 T)$. Maintaining and updating W^{-1} and the Cholesky decomposition of W, and computing the optimal step size following equations on Appendix B, each costs $O(d^2)$ operations. To conclude, the total computational complexity per an active block-coordinate step is $O(\gamma^2 dT + d^2)$. For dense COMET, when taking Nd coordinate steps, the overall complexity of dense COMET is

$$O(N \cdot (\gamma d)^2 T + N \cdot d^3) \quad . \tag{13}$$

For sparse-COMET, when taking ρNd active coordinate steps and $(1-\rho)Nd$ zero-update coordinate steps, the overall complexity of sparse-COMET is

$$O(N \cdot (\gamma d)^2 T + \rho N \cdot d^3). \tag{14}$$

In our experiments dense-COMET converged within $6d$ to $8d$ coordinate steps and sparse COMET within $8d$ to $11d$ with sparse COMET (d being the data dimensinality). As a comparison, consider using SGD or mini-batches for the objective of Eq. (2) (with $\alpha = 0$) and projecting onto the PD cone every P triplets ($P \ll T$), as proposed in Chechik et al. (2010); Qian et al. (2014a). The computational complexity per data pass becomes $O((\gamma d)^2 T + \frac{T}{P} d^3)$. This approach is slower than COMET and only reaches the complexity of COMET when projections are very rare. For example, Qian et al. (2014a) used mini-batches of $P = 10$ triplets. When the number of triplets is $T = 100k$, the resulting computational complexity is on the order of 1000 times larger than with COMET.

As another comparison, consider Liu et al. (2014). The fast heuristic version of HDSL costs $O(M\gamma d + Tk)$ per coordinate step, where M is the size of mini-batch, k is the iteration number. This is summed over $k = 1, ..., O(d^2)$ iterations, since each HDSL step considers a single pair of features, updating 4 matrix entries as opposed to $2d - 1$ entries in COMET. Overall, this yields

$O(M\gamma d^3 + Td^4)$ computations for HDSL, compared with Eq. (13) of COMET. Since both M and N are typically small, this means HDSL is more costly than COMET by a factor of $\frac{d^2}{N}$. In our experiments, HDSL sometimes uses much less than $O(d^2)$ iterations, but then achieved a significantly inferior test error. We believe that HDSL would excel in cases where the true optimum is very sparse, and there is no need to go over the entire set of $\binom{d}{2}$ coordinate pairs.

Finally, we compare with the complexity of LEGO (Jain et al., 2009). LEGO requires $O(d^2)$ computation per constraint. Thus for N passes over T triplets LEGO's complexity is $O(N \cdot d^2 \cdot T)$. Assuming an equal number of passes over the triplet, we find that COMET's complexity Eq. (13) is asymptotically better than LEGO as long as $N \cdot T \cdot (1 - \gamma^2) > d^2$. That is, whenever the data is even moderately sparse, and the number of triplets is larger than the number of matrix parameters.

dense and sparse COMET Memory footprint: Keeping the data triplets in memory takes $O(\gamma dT)$ elements and holding W and W^{-1} and the Cholesky decomposition costs $O(d^2)$. The total memory usage is $O(\gamma dT + d^2)$.

Appendix D. Convergence proofs

There is a well established body of work showing that with non-overlapping blocks, block-coordinate descent iterates converge w.h.p. in a linear rate to the optimum value (Nesterov, 2012; Richtárik and Takáč, 2014). However, the blocks we use in our method are overlapping - for example the $(1, 2)$ coordinate of the matrix is a part of both the 1st and the 2nd column-row. To address this case, we use a more general convergence result applicable to overlapping blocks, given by Richtárik and Takáč (2013). Richtárik and Takáč give a very general result, suitable for *any* distribution over the set of coordinate subsets. Specifically of interest to us, Richtárik and Takáč give sufficient conditions for a linear convergence rate of overlapping block-coordinate descent with a strongly convex smooth objective. We use a relatively simple distribution over coordinate subsets: we have d overlapping blocks corresponding to the column-rows of the matrix, each sampled with a probability p_i, $i = 1 \dots d$ (in the experiments below we used a uniform $p_i = \frac{1}{d}$).

The step sizes implied by the convergence theory presented in this section are conservative underestimates, especially since many of the constants involved in obtaining the step-sizes cannot be evaluated exactly but can only be upper-bounded. In practice, we found that much faster convergence is gained using larger steps while staying within the PD cone, using the Schur complement driven procedure described in detail in section 4.1.

To show convergence, we must prove our objective satisfies two assumptions: Assumption 1, called "Expected Separable Overapproximation", is that in expectation over the choice of blocks the function is smooth w.r.t. an inner product given by the coordinate probabilities. Assumption 2 is that the objective is strongly convex. In addition, for technical reasons the objective must be differentiable. This means that technically our proof is only valid for the squared hinge-loss and log-loss, but not the non-differentiable hinge-loss.

To fulfill the conditions in (Richtárik and Takáč, 2013), we must slightly modify the objective function $L(W)$. The objective $L(W)$ is strongly convex but is not smooth, since the gradient of the $\log\det$ term is unbounded near the envelope of the PD cone. Let $\tilde{L}(W) = L(W + \kappa I_d)$ be a modified version of the loss in Eq. 2, where I_d is the $d \times d$ identity matrix, and $\kappa > 0$ is a fixed parameter. Note that our algorithm can easily minimize $\tilde{L}$, the only difference being that we now need to maintain and update both W^{-1} and $(W + \kappa I_d)^{-1}$, which does not change the asymptotic computational complexity. The additional κI_d term acts as a bias term, where we add a constant

Euclidean distance term to the distance we learn. We note that in practice we found that simply setting $\kappa = 0$ had no detrimental effect on our performance.

We show that the modified objective $\tilde{L}$ obeys Assumptions 1 and 2 of Richtárik and Takáč (2013). Thereby, according to Theorem 3 of Richtárik and Takáč, Algorithm 1 converges with high probability to the optimum value in a linear rate:

Theorem 1 *Let W^t be the t-th iterate of Algorithm 1 with objective function $\tilde{L}(W)$, sampling each column-row i with probability p_i. Let $\tilde{L}^*$ be the optimal value of $\tilde{L}(W)$ on the PD cone. Let $\beta^* \geq \beta$ be the strong convexity parameter of $\tilde{L}(W)$, M^1 a constant depending on the norm of the dataset, $\Lambda = \max_i \frac{1}{p_i}$, $\rho > 0, \epsilon > 0$. Then:*

If $t > \frac{\Lambda(M^1+\alpha d(1/\kappa)^2+\beta)}{\beta^*} log\left(\frac{\tilde{L}(W^0)-\tilde{L}^*}{\epsilon\rho}\right)$ then $Prob(\tilde{L}(W^t) - \tilde{L}^* \leq \epsilon) \geq 1 - \rho$.

We first establish two auxiliary lemmas, then proceed to the proof of Theorem 1.

Lemma 2 (Smooth objective) *Let:*

$\tilde{L}(W) = \sum\limits_{t \in T} l_{W+\kappa I}(\mathbf{q}_t, \mathbf{p}_t^+, \mathbf{p}_t^-) - \alpha \cdot \log\det(W + \kappa I) + \frac{\beta}{2} \cdot \|W + \kappa I\|_F^2$, *where $l_{W+\kappa I}$ is either the squared hinge loss or the log-loss, and $\tilde{L}(W)$ is defined over the positive semidefininte cone. Let $H^i \in \mathbb{R}^{d \times d}$, $i = 1 \ldots d$, be a symmetric matrix with non-zero entries only on the i-th row and column. For any W and H^i such that $W + H^i$ is PSD, there exists a positive constant M_i such that:*

$$\Delta L \leq \langle \frac{\partial \tilde{L}(W)}{W}, H^i \rangle + \frac{M_i}{2}\|H^i\|_F^2 = \sum_{k,l=1}^{d} \frac{\partial \tilde{L}(W)}{W_{kl}} H^i_{kl} + \frac{M_i}{2} \sum_{k,l=1}^{d} (H^i_{kl})^2,$$

with the constant $M_i \leq 2\sum_{t=1}^{T}(\mathbf{q}_{ti}^2 + \Delta\mathbf{p}_{ti}^2) + \frac{\alpha d}{\kappa^2} + \beta$ and $\Delta L = \tilde{L}(W + H^i) - \tilde{L}(W)$.

Proof The objective $\tilde{L}(W)$ is comprised of three terms: (1) the sum of loss terms, (2) the $\log\det$ term, and (3) the Frobenius regularization term. We will bound each of the separately, denoting the positive bounding constants M_i^1, M_i^2 and M_i^3, respectively.

Assuming the instances $\mathbf{q}_t$ and $\mathbf{p}_t$ are unit normalized, straightforward computation shows that for the term (1), inequality 15 holds true for $M_i^1 \leq 2\sum_{t=1}^{T}(\mathbf{q}_{ti}^2 + \Delta\mathbf{p}_{ti}^2)$.

To show that 15 is true for the $-\log\det$ term, we bound the maximal eigenvalue of its Hessian $\mathcal{H}$, which upper bounds M_i^2 by convexity and standard use of a Taylor expansion. The Hessian is a $d^2 \times d^2$ PSD matrix, due to convexity and twice-differentiability of $-\log\det$. At every point $X = W + \kappa I, W \succ 0$, the Hessian $\mathcal{H}(X)$ defines a bilinear form $\mathcal{B}_X(P, Q)$ on the set of symmetric $d \times d$ matrices. This bilinear form is $\mathcal{B}_X(P, Q) = tr\left(X^{-1}PX^{-1}Q\right)$ (Boyd and Vandenberghe, 2004, Appendix A). We then have:

$$\begin{aligned}
&\max eig(\mathcal{H}) = \max_{\|P\|_F=1} \mathcal{B}_X(P, P) = \\
&\max_{\|P\|_F=1} tr\left(X^{-1}PX^{-1}P\right) \leq \\
&\max_{\|P\|_F=1} \|X^{-1}P\|_F^2 \leq \|X^{-1}\|_F^2 \leq \\
&d\|X^{-1}\|^2 = \frac{d}{\|X\|^2} \leq \frac{d}{\kappa^2},
\end{aligned}$$

where in the last line we denote the spectral norm (maximum singular value) of X by $\|X\|$. The last inequality is due to the fact that $X = W + \kappa I$, $W \succ 0$. We therefore have a bound $M_i^2 \leq \frac{\alpha d}{\kappa^2}$. Finally, the constant M_i^3 for the Frobenius regularization is immediately seen to be β.

Collecting all the terms together, we obtain an overall bound on the constant: $M_i \leq M_i^1 + M_i^2 + M_i^3 \leq M_i^1 + \frac{\alpha d}{\kappa^2} + \beta$. ■

Let us define a matrix $P \in \mathbb{R}^{d \times d}$ such that $P_{ij} = p_i + p_j$ for $i \neq j$, $P_{ii} = p_i$. P is defined such that P_{ij} is the probability of updating the (i, j) entry of the matrix W at any given iteration. To show our method converges in a linear rate, we must show that $\tilde{L}(W)$, P and the constants M_i satisfy the "Expected Separable Overapproximation" assumption presented by Richtárik and Takáč (2013).

Lemma 3 (Expected Separable Overapproximation) *For any symmetric $H \in \mathbb{R}^{d \times d}$ such that $W + H$ is PSD, let $H^i \in \mathbb{R}^{d \times d}$, $i = 1 \dots d$ be identical to H on the i-th row and column, and 0 elsewhere. Then:*

$$\mathbb{E}_{i \sim Mult(p_1,\dots,p_d)} \left[\tilde{L}(W + H^i) \right] \leq \tilde{L}(W) + \sum_{k,l=1}^{d} \frac{\partial \tilde{L}(W)}{W_{kl}} H_{kl} P_{kl} + \frac{1}{2} \sum_{k,l=1}^{d} M_k (H_{kl})^2 P_{kl}, \quad (15)$$

where i is sampled from a multinomial distribution with parameters $(p_1, \dots, p_d)$.

Proof

$$\mathbb{E}_{i \sim Mult(p_1,\dots,p_d)} \left[\tilde{L}(W + H^i) \right] = \sum_{i=1}^{d} p_i \tilde{L}(W + H^i) \overset{(a)}{\leq}$$

$$\sum_{i=1}^{d} p_i \left(\tilde{L}(W) + \sum_{k,l=1}^{d} \frac{\partial \tilde{L}(W)}{W_{kl}} H_{kl}^i + \frac{M_i}{2} \sum_{k,l=1}^{d} (H_{kl}^i)^2 \right) \overset{(b)}{=}$$

$$\tilde{L}(W) + \sum_{k,l=1}^{d} \frac{\partial \tilde{L}(W)}{W_{kl}} \sum_{i=1}^{d} p_i H_{kl}^i + \sum_{k,l=1}^{d} \sum_{i=1}^{d} p_i \frac{M_i}{2} (H_{kl}^i)^2 \overset{(c)}{=}$$

$$\tilde{L}(W) + \sum_{k,l=1}^{d} \frac{\partial \tilde{L}(W)}{W_{kl}} H_{kl} P_{kl} + \frac{1}{2} \sum_{k,l=1}^{d} M_k (H_{kl})^2 P_{kl}.$$

Inequality (a) is due to Lemma 2. Equality (b) is by changing the order of summation and since the p_i sum to 1. Equality (c) is by a simple counting argument, and the fact that H^i is the restriction of H to its i-th row and column. Each off-diagonal element H_{kl} appears twice in the sum over i: when $i = k$ and $i = l$. This is accounted for by the elements $P_{kl} = p_k + p_l$. ■

Proof [Theorem 1] We show that Algorithm 1 with objective function $\tilde{L}(W)$ squared-hinge loss or log loss), sampling each column-row i with probability $p_i > 0$, and using step sizes $\eta_i \leq \frac{1}{M_i}$, follows Assumption 1 and Assumption 2 of Richtárik and Takáč (2013). From this the convergence result follows from Richtárik and Takáč, Theorem 3, plugging in our bounds regarding the smoothness and strong convexity of $\tilde{L}(W)$.

We first note that our algorithm is indeed a special case of the algorithm presented in Richtárik and Takáč (2013). Specifically, our algorithm assigns probability $p_i > 0$ to each of the d column-rows of a matrix, and probability 0 to every other possible choice of coordinates. We update along

this block, and the $\log\det$ term acts as a barrier function assuring us we will stay within the PD cone.

Lemma 3 shows our objective is smooth and satisfies Assumption 1 of Richtárik and Takáč. Assumption 2 of Richtárik and Takáč is immediately satisfied because of the Frobenius regularization term, ensuring a strong convexity term $\beta^* \geq \beta > 0$. The result follows by considering that the probability P_{ij} of updating coordinate (i, j) obeys $P_{ij} \geq \min_i p_i$ and the values of M_i given in Lemma 2. ■

Appendix E. Selecting the step size for the sparse proximal step

First, we bring the closed-form solution of the proximal step of Eq. (8) for step size θ (Bach et al., 2012):

$$\begin{cases} H_k(\theta) \doteq (V_k - \theta G_k) \\ {V_k}^{new}(\theta) = H_k(\theta) \cdot [1 - \frac{\lambda}{||H_k(\theta)||_F}]_+ & \text{if } k \geq 1 \\ {V_k}^{new}(\theta) = H_k(\theta) & \text{if } k = 0, \end{cases} \tag{16}$$

where G_k is a matrix notation of the gradient step, as discussed in 4.

In section 4.1 we demonstrated how to bound the step size for a single row-column update given a PD matrix and and a single row-column update matrix G, using the quadratic inequality Eq. (11). However the same method cannot be directly applied to evaluate the proximal step size θ from 8, because as one can see above, the closed form solution for the proximal step ${V_k}^{new}$ is a sum of a gradient step and a shrinkage operation, and is therefore a more complicated function of θ.

We proceed as follows. Define an origin O which is the point where the proximal step will result with $\mathbf{0}$. For clarification, *this is not a zero update*. Instead it means that ${V_k}^{new}$ is all zeros. We also note that H_k represents the ordinary coordinate update, as in section 4.1. Finally, from the update equation 16 it is clear that the updated point ${V_k}^{new}$ is a convex combination of O and H_k. Therefore, from the convexity of the PSD cone, if both O and H_k are PSD, then the update will be PSD as well. We can determine a maximum θ_{max} for which $H_k(\theta_{max})$ will be PSD by inequality Eq. (11), and we can use inequality Eq. (11) again to determine whether O is within the PSD cone.

This leads to two possibilities: if O is within the PSD cone, then necessarily ${V_k}^{new}(\theta_{max})$ is PSD, since it is a convex combination of two PSD matrices, and we may use any step-size which is lesser than or equal to θ_{max}.

If, on the other hand, O is not within the PSD cone, we need to find a maximal scalar ρ_{max} such that $O' = W + \rho_{max}(W - O)$ is PSD. This is again done using the inequality Eq. (11). We can now perform a binary search on θ between 0 and θ_{max} and test whether ${V_k}^{new}(\theta)$ is within the triangle whose vertices are W, $W - V_k + H_k(\theta_{max})$, and O', where we know that this triangle is entirely within the PSD cone. This procedure still results in $O(d^2)$ computational complexity, albeit with a larger constant which may depend on the required precision. In practice, we found that even when we learn a moderately sparse metric (less that 30% sparsity), it is very rare that the O is not PD. Therefore, in our experiments we simply skip the update in case its O is not PD.

We stress that the bound in Eq. (11) is evaluated twice on an active step of sparse COMET. Hence, it would take approximately twice the time in comparison to a step of dense COMET. Therefore, in the case it is required to learn a dense matrix, dense COMET would be faster.

To conclude, we demonstrated how to evaluate a bound on the step size that maintains the proximal step inside the PD cone, such that during training we can adaptively set the step size accordingly.

Appendix F. Supplemental Results

Precision at k for three datasets: *Reuters CV1* 1K features, *Caltech-256* 249 categories and *Protein*.

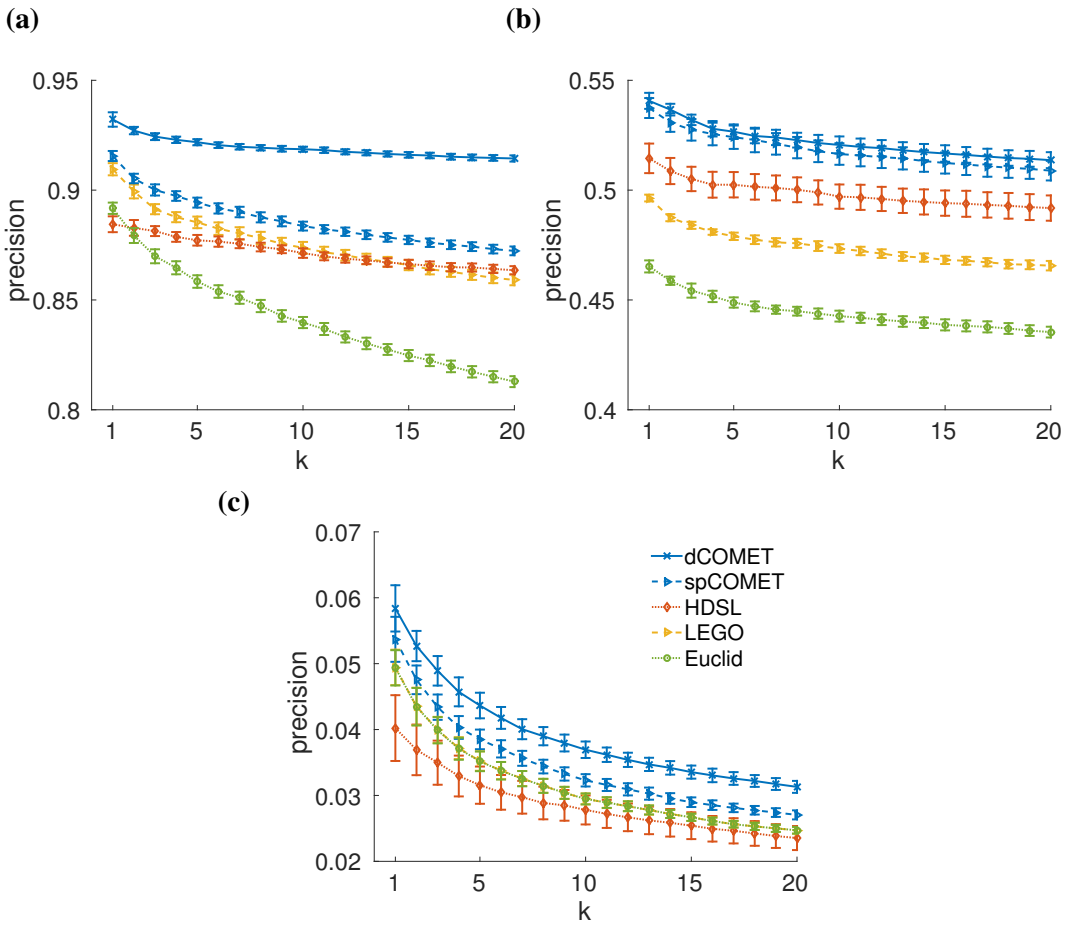

Figure 4: *precision-at-top-k* for three datasets. **(a)** REUTERS CV1 with 1000 features. **(b)** Protein (LIBSVM) with 357 features. **(c)** Caltech256 (249 Categories) with 1000 features.