Constraint-Aligned Subspace Transformation for Scalable Kernel Metric Learning in Clustering

Fei Wang*†
University of Eastern Finland
School of Computing
Joensuu, Finland
fei.wang@uef.fi

Le Li*
University of Eastern Finland
School of Computing
Joensuu, Finland
leli@cs.uef.fi

Pasi Fränti
University of Eastern Finland
School of Computing
Joensuu, Finland
franti@cs.uef.fi

Abstract

Kernel Distance Metric Learning (KDML) is a powerful tool for enhancing clustering performance by adapting the data geometry to correct structural mismatches. However, traditional global KDML methods suffer from severe scalability issues, often incurring $O(n^3)$ computational costs and $O(n^2)$ memory usage, where n is the sample size. Furthermore, these holistic deformations can lead to overfitting when supervision is sparse. To address these limitations, we propose Constraint-Aligned Subspace Transformation (CAST), a novel framework that restricts metric adaptation to a low-dimensional subspace strictly induced by pairwise constraints. By orthonormalizing constraint directions, we reduce the learning problem from estimating a full kernel matrix to a rank-efficient Linear Program (LP). We further derive an implicit update formulation that enables distance evaluation in $O(r)$ time, effectively reducing memory complexity to $O(nr)$. Extensive experiments on diverse datasets demonstrate that CAST matches or exceeds state-of-the-art KDML competitors in downstream clustering tasks while dramatically reducing optimization time, delivering 33×-259× speedups over the fastest baseline in the weak-supervision (1%) regime, while still achieving 3×–20× speedups at 20% supervision.

CCS Concepts

• **Computing methodologies** → **Kernel methods**; *Cluster analysis.*

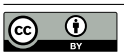
Keywords

Clustering, Kernel Distance Metric Learning, Linear Programming, Low-rank approximation, Scalability

ACM Reference Format:
Fei Wang, Le Li, and Pasi Fränti. 2026. Constraint-Aligned Subspace Transformation for Scalable Kernel Metric Learning in Clustering. In *Proceedings of the 32nd ACM SIGKDD Conference on Knowledge Discovery and Data Mining V.2 (KDD '26), August 09–13, 2026, Jeju Island, Republic of Korea.* ACM, New York, NY, USA, 12 pages. https://doi.org/10.1145/3770855.3818171

Resource Availability:
The source code of this paper has been made publicly available at https://doi.org/10.5281/zenodo.20417880.

*Fei Wang and Le Li contributed equally to this research.
†Correspondence to: Fei Wang



KDD '26, Jeju Island, Republic of Korea

ACM ISBN 979-8-4007-2259-2/2026/08
https://doi.org/10.1145/3770855.3818171

1 Introduction

Distance metric learning (DML) aims to learn a task-adaptive notion of similarity from limited supervision, typically in the form of class labels, triplets, or pairwise relations. By reshaping the geometry of the data, DML has become a standard ingredient in various tasks, including clustering [22], image retrieval [18], face recognition [14, 25], object detection [28], anomaly detection [20], recommendation systems [1], etc., where even sparse must-link (cannot-link) feedback can steer partitions toward semantically meaningful structures.

In practice, the supervision available for clustering is often weak and local: users provide only a small set of pairwise constraints, which encode partial and noisy knowledge about which points should (or should not) belong together. This makes how a metric is parameterized and where it is allowed to change the geometry central to both performance and robustness.

Most classical DML methods operate in the linear regime, learning a Mahalanobis metric in the input space. While efficient and well-understood, linear DML inherits an inherent representational bottleneck: a single global linear map cannot adequately reconcile complex latent structures that arise from nonlinear manifolds, curved decision boundaries, or higher-order feature interactions. As a result, when the cluster geometry is intrinsically non-Euclidean or varies across the space, purely linear formulations often fail to resolve the mismatch between the learned metric and the true semantic neighborhood structure, limiting downstream clustering quality and stability [4, 22].

Kernel-based distance metric learning (KDML) extends DML beyond this limitation through a two-stage design: nonlinear mapping followed by metric learning in the induced space [2, 9, 16, 23, 30]. First, data are implicitly embedded into a high-dimensional Reproducing Kernel Hilbert Space (RKHS), where nonlinear relationships can become linearly representable. Second, a metric (equivalently, a kernel transformation) is learned in this space, enabling flexible geometric adaptation that can capture non-spherical and highly curved cluster shapes.

However, the same design also amplifies the cost and risk of learning. Even forming the kernel matrix $K \in \mathbb{R}^{n\times n}$ can be prohibitive at scale, with quadratic storage and at least quadratic-time construction, motivating approximations[33] such as Nyström sampling [8] or random feature expansions [6, 19]. More critically, global KDML methods typically learn a high-capacity PSD operator in the RKHS whose degrees of freedom scale with the sample size, leading

to optimization costs that grow quadratically or cubically with n [12, 23]. From a statistical standpoint, this "holistic" deformation is also problematic under sparse supervision: when constraints cover only a small portion of the space, an unrestricted RKHS-level transformation can distort regions that are effectively unconstrained, producing brittle solutions that fit the provided constraints yet harm the intrinsic geometry elsewhere [2, 32]. These two issues, scalability and overfitting under sparse pairwise supervision, constitute a key barrier to deploying KDML as a practical component of weakly supervised clustering pipelines.

Motivated by these limitations, we propose **Constraint-Aligned Subspace Transformation (CAST)**, a kernel metric adaptation framework tailored for sparse pairwise supervision. The proposed framework makes three key contributions:

- **Structure-preserving subspace metric adaptation.** We introduce a constraint-aligned parameterization of kernel metrics in which admissible adaptations are confined to a low-dimensional subspace spanned by constraint-supported directions. This explicitly couples the degrees of freedom of the learned metric to the available supervision, reducing the tendency to distort unconstrained regions when constraints are sparse.
- **Rank-efficient linear programming.** By orthonormalizing this subspace and restricting the adaptation to within it, we reduce kernel metric learning from estimating an $O(n^2)$ PSD operator to learning r nonnegative scalar weights, where r is the intrinsic rank of the constraints. This yields a globally optimal linear programming formulation whose complexity depends on constraints rather than dataset size.
- **Scalable implicit inference.** We derive an implicit low-rank update scheme that enables distance evaluation in $O(r)$ time and reduces memory from $O(n^2)$ to $O(nr)$, making kernel metric adaptation practical when global KDML methods are computationally prohibitive.

Extensive experiments show that CAST achieves clustering performance comparable to or better than state-of-the-art KDML baselines while reducing optimization runtime by orders of magnitude.

2 Preliminaries and Global KDML

We consider a semi-supervised setting where a dataset $X = \{x_i\}_{i=1}^n \subset \mathbb{R}^d$ is accompanied by a limited set of pairwise constraints. Let $\mathcal{M}$ and C denote the sets of *must-link* and *cannot-link* pairs, respectively. The goal is to learn a distance metric that respects these constraints, pulling $\mathcal{M}$ pairs closer while pushing C pairs apart, to facilitate downstream tasks such as clustering.

2.1 Global Metric Learning in RKHS

To handle non-linear structures, we employ a positive semidefinite kernel function $k(x_i, x_j) = \langle \phi(x_i), \phi(x_j) \rangle_{\mathcal{H}}$, which implicitly maps data into a high-dimensional Hilbert space $\mathcal{H}$ via $\phi : \mathbb{R}^d \to \mathcal{H}$. We denote the data representation in this space as $\Phi = [\phi(x_1), \ldots, \phi(x_n)]$. The geometry of this space is captured by the kernel matrix $K \in \mathbb{R}^{n \times n}$, where $K_{ij} = k(x_i, x_j)$.

In this induced space, the squared Euclidean distance between two samples is given by:

$$d^2_{\mathcal{H}}(x_i, x_j) = \|\phi(x_i) - \phi(x_j)\|^2_{\mathcal{H}} = K_{ii} + K_{jj} - 2K_{ij}. \quad (1)$$

This distance relies solely on the pre-defined kernel K and acts as the baseline geometry before learning.

Standard KDML seeks to adapt this baseline geometry. A generalized Mahalanobis metric in $\mathcal{H}$ is defined by a positive semidefinite operator $M \succeq 0$. The generalized distance is:

$$d^2_M(x_i, x_j) = \langle \phi(x_i) - \phi(x_j),\, M\,(\phi(x_i) - \phi(x_j)) \rangle_{\mathcal{H}}. \quad (2)$$

Since any valid metric adaptation in the RKHS is intimately tied to the sample span, the operator M is typically parameterized via a kernel-level transformation, and then downstream tasks are performed under the adapted geometry.

2.2 The Global Optimization Bottleneck

Most KDML methods learn a *global* deformation of the RKHS geometry by optimizing a full-capacity PSD operator. This approach introduces two fundamental limitations:

1. Computational and memory scaling dominated by n. A global RKHS metric typically requires manipulating kernel-level matrices (via inversion, eigendecomposition, or matrix functions), leading to quadratic memory $O(n^2)$ and time complexity that is cubic $O(n^3)$ in the worst case. Therefore, the optimization cost is governed by the dataset size n, rather than the supervision size $|\mathcal{M}| + |C|$ that actually drives the learning signal.

2. Cost-capacity mismatch under sparse supervision. Global KDML deforms the geometry everywhere in the RKHS, even in regions devoid of constraints. When supervision is sparse, such holistic deformations can overfit to a small set of pairs, unintentionally distorting the intrinsic structure of the data and harming interpretability.

Below, we categorize representative global KDML strategies and explain how they suffer from these bottlenecks.

2.2.1 Iterative Global Solvers (LogDet, Dual Programs). This category includes methods that solve for the kernel transformation via iterative optimization techniques. **Bregman Projections:** Approaches such as [2, 13] minimize a LogDet divergence subject to constraints. They update the kernel via iterative projections. While flexible, the process repeatedly operates on $n \times n$ matrices, becoming expensive when n is large or constraints are conflicting. **Dual Convex Programs:** Methods like the kernel metric learning for supervised k-means [22] formulate a global convex program and solve it via Lagrange duality. Although the dual form can be more efficient than generic PSD optimization, the solution remains a global operator in the RKHS sample span, inheriting the $O(n^2)$ memory footprint.

2.2.2 Direct Matrix Solvers (Statistical, Geometric, and Spectral). These methods avoid iterative loops by deriving an analytic or spectral solution. However, they invariably rely on expensive matrix operations (inversion, square root, or eigendecomposition) on the full $n \times n$ kernel matrix. **Distributional Divergence :** Methods like [24] and [23] maximize the separability between constraint-induced distributions. They derive closed-form metrics involving the difference of inverse covariance matrices, incurring a fixed $O(n^3)$ cost for

inversion. **Riemannian Geometry:** Kernel GMML [9] computes the metric as a geometric mean on the SPD manifold. This requires expensive matrix functions (square roots, logarithms) of size $n \times n$. **Discriminative Scatter:** Recent discriminative methods like [35] maximize a trace ratio of between-class to within-class scatter. In the kernel setting, this results in a generalized eigenproblem of size $n \times n$, inheriting global scaling costs.

Whether iterative or direct, all these approaches are dominated by manipulating full $n \times n$ kernel objects. This yields unfavorable scaling and a structural mismatch between global capacity and local supervision. These limitations motivate the proposed CAST framework, which restricts adaptation to a low-dimensional subspace strictly induced by supervision.

2.3 Relation to Low-rank Kernel Approximation

Low-rank kernel approximation methods, such as Nyström [7] sampling and random feature expansions [6, 19], primarily act at the **kernel construction stage**: they approximate the kernel-induced representation to reduce the cost of forming, storing, or multiplying the baseline kernel matrix K [8, 27]. In contrast, KDML targets the **kernel metric adaptation** stage, where a data-dependent geometric deformation (often expressed as a kernel-level operator) is learned from supervision.

These two directions are complementary rather than competing. A low-rank approximation of K can reduce shared kernel overhead and can be used as input to a KDML method. However, kernel approximation alone does not address the *cost-capacity mismatch* of global KDML under sparse constraints, where the optimization still targets a global transformation with degrees of freedom coupled to n.

In this work, we address this bottleneck by restricting the admissible metric adaptation to a supervision-induced subspace, thereby tying the learnable degrees of freedom to the constraint evidence rather than to the dataset size.

3 Constraint-Aligned Subspace Metric Adaptation

As illustrated in Fig. 1, the proposed CAST framework learns a metric by operating within a constraint-defined low-rank subspace. Strategically scaling the subspace to satisfy constraints causes the transformation of feature space, resulting in the learned metric. The following section provides a detailed introduction to the mathematical formulas of the CAST framework.

3.1 Constraint-Aligned Scaling Transformation

We derive the proposed Constraint-Aligned Subspace Transformation (CAST) from a structural requirement: the learned representation correction should act only on directions supported by pairwise constraints, and remain an identity mapping elsewhere.

Constraint-induced subspace. For each constrained pair (i, j), define the indicator difference vector $z_{ij} = e_i - e_j \in \mathbb{R}^n$. We collect these into the indicator matrices $Z_{\mathcal{M}} = [z_{ij}]_{(i,j)\in\mathcal{M}}$ and $Z_{\mathcal{C}} = [z_{ij}]_{(i,j)\in\mathcal{C}}$, forming the full constraint matrix:

$$Z = [Z_{\mathcal{M}}, Z_{\mathcal{C}}] \in \mathbb{R}^{n\times m}, \qquad m = |\mathcal{M}| + |\mathcal{C}|. \tag{3}$$

In the RKHS $\mathcal{H}$, each pair induces a difference vector $v_{ij} = \phi(x_i) - \phi(x_j) = \Phi z_{ij}$. The collection of all such vectors forms the constraint directions $V = \Phi Z$, spanning the subspace:

$$\mathcal{U} = \mathrm{span}(V) \subseteq \mathcal{H}. \tag{4}$$

Design principle: identity outside constraint evidence. We enforce the restriction that the operator Q acts as the identity on the orthogonal complement of the constraint subspace:

$$Qu = u, \quad \forall u \in \mathcal{U}^{\perp}. \tag{5}$$

Lemma 3.1 (A constraint-supported update family). *Let $\mathcal{U} = \mathrm{span}(V)$. Any operator of the form*

$$Q = I - VAV^{\top}, \tag{6}$$

for some matrix $A \in \mathbb{R}^{m\times m}$, satisfies (5).

Proof. For any $u \in \mathcal{U}^{\perp}$, we have $V^{\top}u = 0$. Thus,

$$(I - VAV^{\top})u = u - VA(V^{\top}u) = u - 0 = u.$$

□

Orthonormalizing constraint directions. To eliminate redundancy caused by correlated constraints, we work in the orthogonal coordinates of the constraint Gram matrix:

$$G = V^{\top}V = Z^{\top}KZ = U\Lambda U^{\top}. \tag{7}$$

Let $U_+ \in \mathbb{R}^{m\times r}$ collect eigenvectors for the $r = \mathrm{rank}(G)$ non-zero eigenvalues, and $\Lambda_+ \in \mathbb{R}^{r\times r}$ be the corresponding diagonal matrix.

Lemma 3.2 (Rank-r reduction and CAST parameterization). *Restricting Q to the family* (6)*, we can parameterize $A = U_+BU_+^{\top}$ for some $B \in \mathbb{R}^{r\times r}$, yielding*

$$Q = I - VU_+BU_+^{\top}V^{\top}. \tag{8}$$

In CAST, we restrict B to be diagonal, $B = \Theta = \mathrm{diag}(\theta_1, \ldots, \theta_r)$, resulting in:

$$Q(\Theta) = I - VU_+\Theta U_+^{\top}V^{\top}. \tag{9}$$

Proof. Let U_0 span the nullspace of $V^{\top}V$. Since $V^{\top}VU_0 = 0$, it follows that $VU_0 = 0$. Thus, any component of A involving U_0 vanishes in the product $VAV^{\top}$. We strictly parameterize the active component B on the range of U_+. CAST further restricts B to a diagonal Θ to perform axis-aligned scaling in the orthogonalized coordinate system, reducing degrees of freedom to r. □

3.2 Distance Modeling

The transformation $Q(\Theta)$ induces a valid PSD metric operator $M(\Theta) = Q(\Theta)^{\top}Q(\Theta)$. This metric decomposes into a fixed projection and a learnable subspace term.

Lemma 3.3 (Projection special case). *Setting $\Theta = \Lambda_+^{-1}$ reduces $Q(\Theta)$ to the orthogonal projector P_V onto $\mathcal{U}^{\perp}$:*

$$P_V = I - VU_+\Lambda_+^{-1}U_+^{\top}V^{\top} = I - V(V^{\top}V)^{\dagger}V^{\top}. \tag{10}$$

Proof. Substituting $\Theta = \Lambda_+^{-1}$ and using $(V^{\top}V)^{\dagger} = U_+\Lambda_+^{-1}U_+^{\top}$ yields the standard projector form. By definition, $P_VV = 0$. □

Corollary 3.4. *For any constrained pair (i, j), the difference vector $v_{ij} \in \mathcal{U}$ satisfies $P_Vv_{ij} = 0$.*

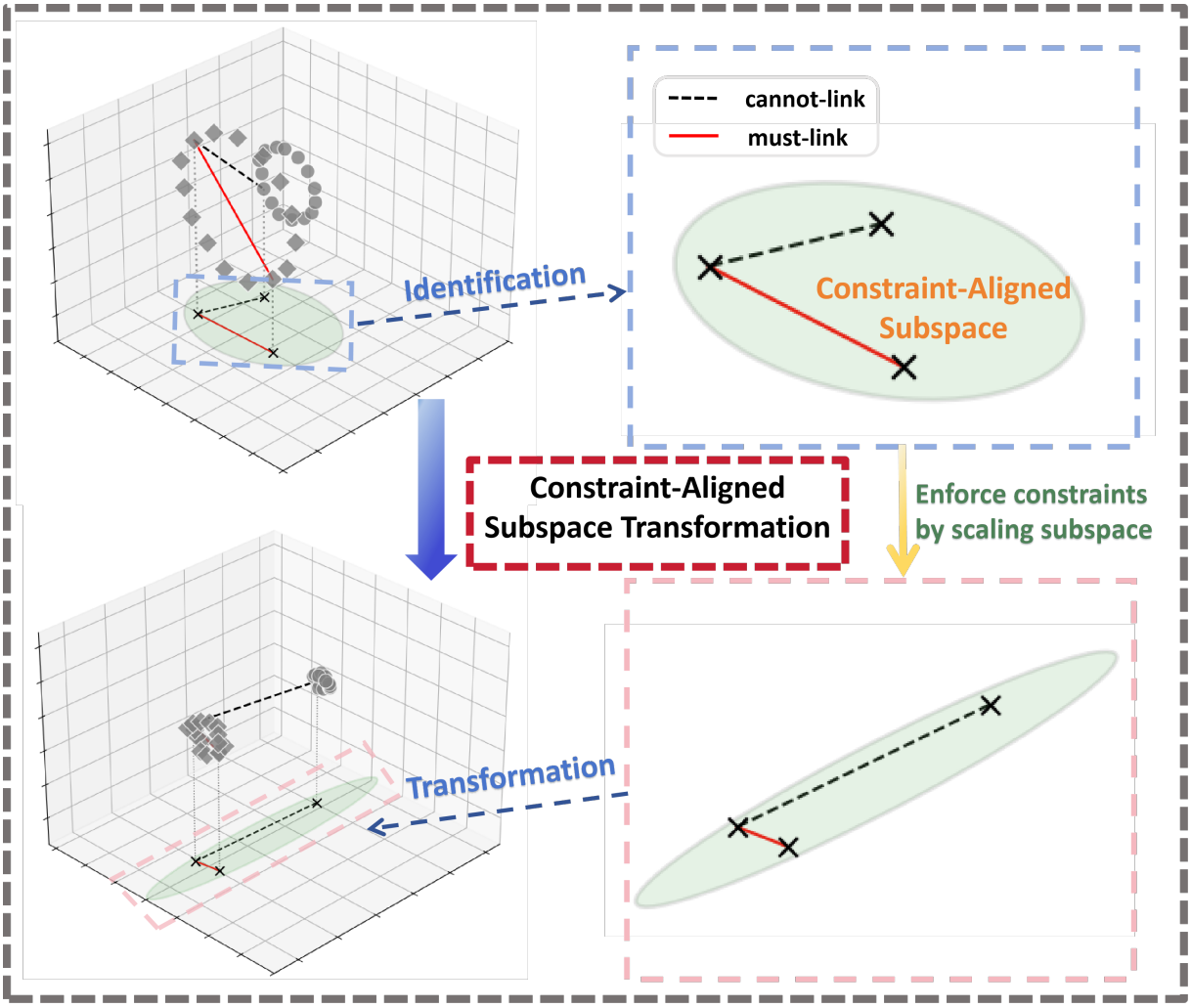


Figure 1: Illustration of the CAST framework for metric learning. The process first identifies a low-rank constraint-induced subspace, with which the CAST operator $Q(\Theta)$ is defined (Eq. 9). Scaling this subspace to enforce all constraints corresponds to learning a Θ; applying this learned transformation $Q(\Theta)$ yields the final learned metric.

Lemma 3.5 (Decomposition of $M(\Theta)$). *Let* $f(\Theta) = \Lambda_+\big(\Theta - \Lambda_+^{-1}\big)^2 \succeq 0$. *The induced metric decomposes as:*

$$M(\Theta) = P_V + M_{\mathcal{U}}(\Theta), \tag{11}$$

where $M_{\mathcal{U}}(\Theta) := VU_+ f(\Theta) U_+^\top V^\top$.

Proof. Expanding $Q(\Theta)^\top Q(\Theta)$ and using $U_+^\top V^\top V U_+ = \Lambda_+$:

$$\begin{aligned} Q(\Theta)^\top Q(\Theta) &= I - 2VU_+\Theta U_+^\top V^\top + VU_+\Theta(\Lambda_+)\Theta U_+^\top V^\top \\ &= I + VU_+(\Theta\Lambda_+\Theta - 2\Theta)U_+^\top V^\top. \end{aligned} \tag{12}$$

Completing the square elementwise, $\Theta\Lambda_+\Theta - 2\Theta = \Lambda_+(\Theta - \Lambda_+^{-1})^2 - \Lambda_+^{-1}$. Substituting this back yields $P_V + VU_+ f(\Theta) U_+^\top V^\top$. □

Crucially, for any constrained pair, the contribution of P_V vanishes. The distance depends solely on the subspace metric:

$$d^2_{M(\Theta)}(x_i, x_j) = \langle v_{ij}, M_{\mathcal{U}}(\Theta) v_{ij} \rangle. \tag{13}$$

3.3 Subspace Metric Learning via Linear Programming

We formulate the learning problem directly over the weights of the constraint-aligned metric $M_{\mathcal{U}}(\Theta)$. Let $w \in \mathbb{R}^r_{\geq 0}$ be the diagonal elements of $f(\Theta)$. The squared distance in Eq. (13) becomes linear in w:

$$\begin{aligned} d^2_{ij}(w) &= \langle v_{ij}, VU_+ \mathrm{diag}(w) U_+^\top V^\top v_{ij} \rangle \\ &= \sum_{k=1}^{r} w_k \underbrace{\left(u_k^\top Z^\top K z_{ij}\right)^2}_{a_{ij,k}}. \end{aligned} \tag{14}$$

The coefficients $a_{ij,k}$ are fully kernelized, depending only on K and constraint indicators. This allows us to formulate the optimization

Algorithm 1 CAST-LP

Require: Kernel $K \in \mathbb{R}^{n\times n}$, Indicators $Z_{\mathcal{M}}, Z_C$.
Ensure: Implicit params $B \in \mathbb{R}^{n\times r}, D \in \mathbb{R}^{r\times r}$.

1. Construct Constraint Basis
Form $Z \leftarrow [Z_{\mathcal{M}}, Z_C] \in \mathbb{R}^{n\times m}$
$G \leftarrow Z^\top KZ$
2. Subspace Identification
$[U, \Lambda] \leftarrow \text{eig}(G)$
Extract r components U_+, Λ_+ where $\Lambda_{kk} > \epsilon$ (Emperically, 1×10^{-6} works well)
3. Compute LP Coefficients
Initialize $A_{LP} \in \mathbb{R}^{m\times r}$
for $p \leftarrow 1$ **to** m **do**
  $g_p \leftarrow G_{:,p}$
  $y_p \leftarrow U_+^\top g_p$
  $A_{LP}[p,:] \leftarrow y_p \odot y_p$
end for
4. Solve Optimization
$c \leftarrow A_{LP}[1:|\mathcal{M}|,:]^\top \mathbf{1}$
Define constraints for C using A_{LP}
$w \leftarrow \text{solve\_LP}(c, \text{constraints})$
5. Output
$B \leftarrow KZU_+$
$D \leftarrow \text{diag}(w) - \Lambda_+^{-1}$
return B, D

as a standard Linear Program (LP):

$$\begin{aligned} \min_{w\in\mathbb{R}^r_{\geq 0}} \quad & \sum_{(i,j)\in\mathcal{M}} d_{ij}^2(w) \\ \text{s.t.} \quad & d_{ij}^2(w) \geq 1, \quad \forall (i,j) \in C. \end{aligned} \tag{15}$$

This LP formulation reduces the optimization variables to the intrinsic rank r, so the optimization stage scales with r, and admits a globally optimal solution.

3.4 Efficient Distance Evaluation

Explicitly updating the $n \times n$ kernel matrix is unnecessary. The updated kernel K' admits a low-rank correction form:

$$K' = K + BDB^\top, \tag{16}$$

where $B = KZU_+ \in \mathbb{R}^{n\times r}$ and $D = \text{diag}(w) - \Lambda_+^{-1}$.

The squared distance for any pair (i, j) can be computed on demand in $O(r)$ time:

$$d'^2(i,j) = \underbrace{(K_{ii} + K_{jj} - 2K_{ij})}_{\text{original distance}} + \underbrace{\sum_{k=1}^{r} D_{kk}\,(b_{ik} - b_{jk})^2}_{\text{rank-}r\text{ correction}},$$

where b_i, b_j are rows of B. This implicit evaluation significantly reduces memory overhead from $O(n^2)$ to $O(nr)$.

3.5 Complexity and scalability analysis

We analyze the computational efficiency of CAST-LP as presented in Algorithm 1, distinguishing between the *preparation phase*, the *learning phase*, and the *inference phase*. Table 1 summarizes these costs compared to standard global kernel DML (See section 2.2).

Preparation and Subspace Construction. Steps 1 and 2 involve constructing the constraint geometry. While global kernel methods typically operate on the full $n\times n$ kernel matrix K, CAST-LP operates on the projected constraint Gram matrix $G = Z^\top KZ \in \mathbb{R}^{m\times m}$. Since the indicator matrix Z is sparse (two non-zeros per column), each entry $G_{pq} = z_p^\top K z_q$ is computed using only constant-time kernel lookups. Thus, building G costs $O(m^2)$ when K is accessible. The eigendecomposition of G (Step 2) depends only on the number of constraints, costing $O(m^3)$. Crucially, since $m \ll n$ in weakly supervised settings, this step is negligible compared to the $O(n^3)$ cost of full kernel operations or global spectral methods.

LP Optimization. Steps 3 and 4 constitute the learning phase. Calculating the coefficients A_{LP} (Step 3) is efficient because we reuse the columns of G. For each of the m constraints, we project a vector of size m onto the r-dimensional subspace, costing $O(m^2 r)$ total. The optimization in Step 4 is a Linear Program with r variables and $|C|$ inequality constraints. Given that $r \leq m$, the complexity is polynomial in m, which is significantly faster than the iterative Semidefinite Programming (SDP) or gradient descent required by methods that learn a full $n \times n$ transformation matrix.

Implicit Inference. Step 5 and the subsequent distance evaluation (Section 3.4) highlight the memory advantage. Instead of storing a dense update matrix (n^2 entries), CAST stores the low-rank basis $B \in \mathbb{R}^{n\times r}$ and diagonal $D \in \mathbb{R}^r$. This reduces space complexity from $O(n^2)$ to $O(nr)$. Furthermore, evaluating a learned distance using the implicit formula requires only $O(r)$ operations per pair, enabling fast retrieval and clustering applications.

4 Experimental Setup

We evaluate the proposed method on diverse benchmark datasets covering multiple application purposes and clustering regimes. We provide the implementation and experimental details via an anonymous link.

4.1 Datasets

Image related datasets. We include COIL20 [21], YaleFace [11, 17], as well as Scene from UCI [3]. We use pixel representations for the image datasets to enable a clear and controlled comparison among kernel metric learning methods under the same input representation.

Text-related datasets. We consider 20Newsgroups and Cnae9 from UCI [3]. For 20Newsgroups, documents are vectorized using TF-IDF features. Cnae9 contains company business descriptions across nine categories; it is preprocessed by retaining letters only, removing function words, normalizing words to canonical forms, and converting documents into frequency-based vectors. We also include Spambase from UCI [3], which originates from email spam filtering. It consists of frequency and statistics-based attributes extracted from email content.

Other tabular datasets. We further include kr-vs-kp, Gina-agnostic, and HAR [3]. These datasets cover binary and multi-class scenarios with low- to medium-dimensional feature vectors collected from diverse scientific scenarios.

Preprocessing and normalization. For raw-pixel image datasets, we rescale pixel intensities to $[0, 1]$. For numeric feature datasets

Table 1: Complexity comparison (n: samples, m: constraints, r: subspace rank, T: iterations). Note that $r \le m \ll n$.

Operation	Global Kernel DML	CAST (Ours)
Preprocessing	$O(n^3)$ (Full Decomp)	$O(m^2)$ (Constraint Gram)
Optimization	$O(T \cdot n^2)$ (SDP/Grad)	$O(\text{poly}(m))$ (LP)
Space cost	$O(n^2)$	$O(nr)$
Inference/Pair	$O(n)$	$O(r)$

(including Scene, SatelliteImage, and the tabular datasets), we apply per-feature z-score standardization. For text datasets, we use TF-IDF representations with ℓ_2 normalization.

4.2 Compared methods

Kernel distance metric learning (kernel DML). We include several kernel DML baselines that learn a data-dependent kernel metric from supervision: (i) *SKMS* [2, 13], which learns a supervision-adapted kernel via an iterative Bregman-projection procedure; (ii) *kDMLMJ* [24], which learns a kernel metric by maximizing the Jeffrey divergence; (iii) *kKISSME* [23], a kernel extension of KISSME [15] using pairwise constraints; (iv) *kDML-SKM* [22], which performs kernel-based metric learning tailored for supervised k-means clustering; (v) *kGMML* [9], a kernelized variant of geometric mean metric learning [34]; (vi) *kDML-CG*, which kernelizes the class-center guided Mahalanobis metric learned by DML-CG [35] to obtain a supervision-adapted kernel metric.

Joint multi-kernel update and clustering. As the only exception to the above two-stage pipeline, we include *CCMKL* [5], which jointly performs multiple kernel update and constrained clustering (with kernel k-means used within the joint optimization). This baseline is closely related to our setting in that it integrates kernel adaptation with clustering under pairwise constraints.

Reference baselines. Where applicable, we also report results of kernel k-means using the initial kernel (without kernel adaptation) to quantify the gain brought by learning.

4.3 Pairwise constraint generation

We adopt a standard semi-supervised setting where supervision is provided as pairwise must-link and cannot-link constraints. For each dataset and trial, we sample a labeled subset $\mathcal{L}$ of size $m = \lfloor pn \rfloor$ (with $p \in (0, 1)$ and n the dataset size) using stratified sampling; whenever feasible, $\mathcal{L}$ contains at least one instance from each class. Ground-truth labels are used *only* to form constraints and to evaluate clustering quality.

From the labeled subset, we generate a balanced constraint set of total size m. Must-link constraints $\mathcal{M}$ are formed by uniformly sampling labeled pairs $(i, j) \in \mathcal{L} \times \mathcal{L}$ with $y_i = y_j$, while cannot-link constraints $\mathcal{C}$ are formed by uniformly sampling labeled pairs with $y_i \neq y_j$, with $|\mathcal{M}| = |\mathcal{C}| = m/2$.

4.4 Implementation details and hyperparameter tuning

Kernel construction and initialization. We use the exact RBF kernel to keep the kernel construction stage fixed across methods and isolate differences in the kernel metric learning (adaptation) stage. Kernel approximation, such as Nyström or Random Fourier Features (RFF) is complementary and can be integrated on top of CAST, but is outside the scope of this study. For a fair comparison, all kernel-based methods start from the same initial RBF kernel $\mathbf{K}$ constructed from the preprocessed data (Section 4.1). Specifically, given feature vectors $\{x_i\}_{i=1}^n$, we compute

$$K_{ij} = \exp\left(-\frac{\|x_i - x_j\|_2^2}{2\sigma^2}\right). \tag{17}$$

The bandwidth σ is selected from a broad candidate range using only the pairwise constraints in each trial: for each candidate σ, we first rescale distances so that the minimum cannot-link distance equals 1, and then choose the σ that yields the smallest sum of distances over must-link pairs while maintaining cannot-link separation. This procedure provides a shared and constraint-aware kernel initialization for all kernel-based baselines.

Hyperparameter tuning protocol. Hyperparameters of all compared methods are selected using only the supervision available in each trial (i.e., the labeled subset and generated constraints), without using ground-truth class labels beyond evaluation. For each method, we perform a grid search over the recommended ranges in the original papers, and select the configuration that optimizes a constraint-based validation criterion computed on a held-out subset of the generated pairwise constraints. The final clustering results are then reported on the full dataset using the selected configuration.

CCMKL requires a set of candidate kernels as input. To keep the comparison controlled and to avoid introducing additional kernel families, we construct a small kernel set by varying the RBF bandwidth around the selected default σ_0: $\sigma \in \{0.5\sigma_0,\ \sigma_0,\ 2\sigma_0,\ 4\sigma_0\}$, where σ_0 denotes the bandwidth chosen by the shared initialization strategy above. CCMKL is then tuned using the same constraint-only protocol as other methods.

LP solver choice. The LP in Step 4 (Algorithm 1) is solved using SciPy `linprog`. We use `interior-point` by default. An ablation across four `linprog` solvers on controlled synthetic regimes shows that solver choice yields nearly identical optima (objective ranges near numerical precision) while affecting runtime by up to orders of magnitude; details are reported in Appendix A.

In this study, all computational experiments were conducted on a system equipped with an AMD R9 7940HX processor and 64 GB of RAM.

5 Experimental Results

We compare all methods under different supervision levels p. Given a specific p (Section 4.3), each method learns an adapted kernel (or an equivalent kernel-induced distance). Unless otherwise stated, we

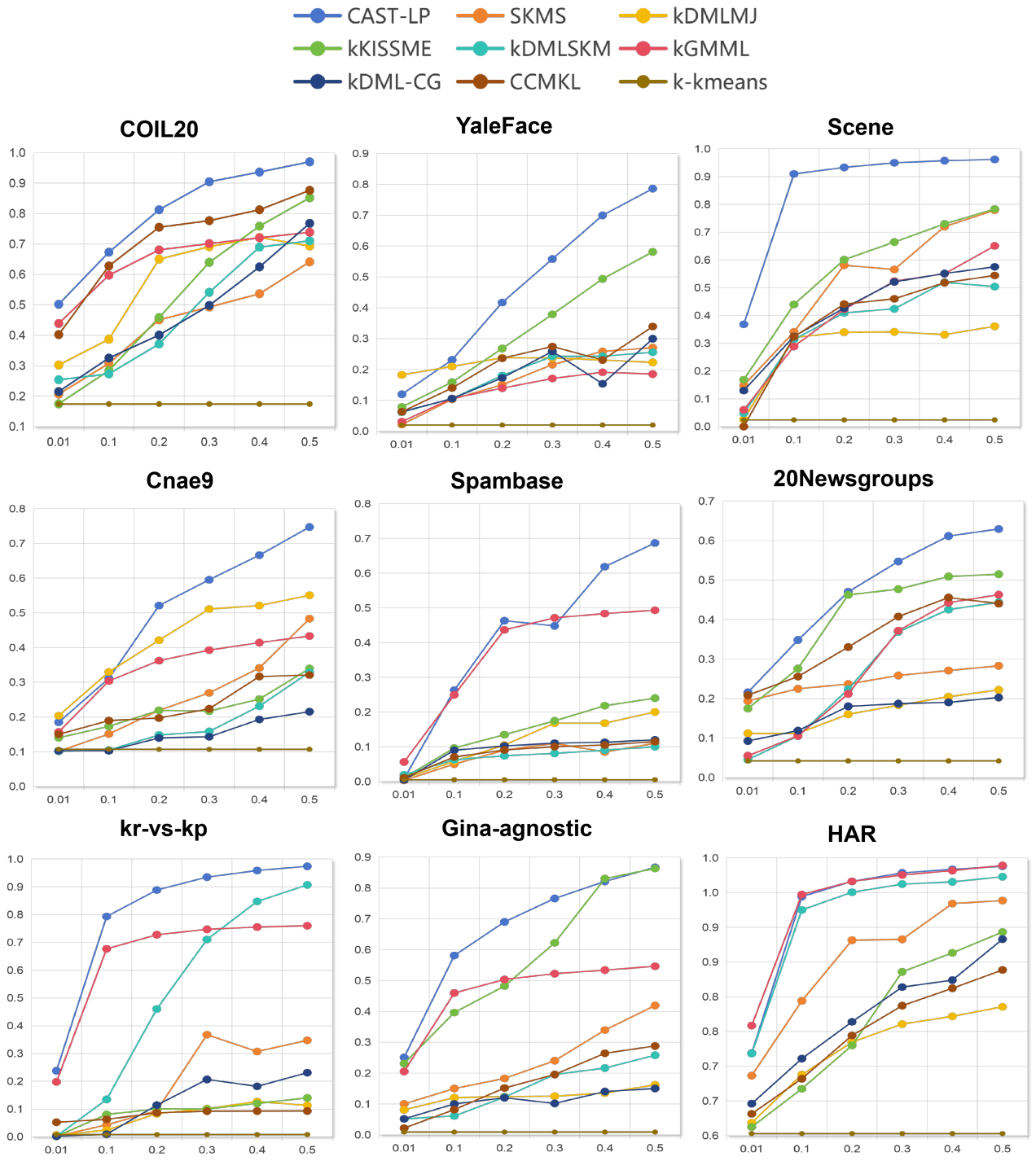


Figure 2: Comparison of clustering performance measured by ARI across 9 datasets. Each plot depicts the clustering results obtained by running k-means with learned kernel metrics for each dataset. The x-axis represents the supervision level p, while the y-axis displays the corresponding ARI scores. k-kmeans represents the plain kernel kmeans without kernel learning.

then apply the same downstream clustering procedure, i.e., kernel k-means, using the learned kernel as input (or equivalently, k-means using the kernel-induced metric). The only exception is CCMKL, which jointly performs kernel learning and constrained clustering within a unified optimization process. For each labeled fraction p, we repeat the sampling and constraint generation process for 20 trials. All reported results are averaged over the 20 trial.

5.1 Clustering performance

Clustering quality is evaluated by comparing the predicted cluster assignments with the ground-truth class labels using standard external measures: adjusted Rand index (ARI) [29] and clustering accuracy (ACC) [10]. ARI evaluates pairwise agreement while correcting for chance, while ACC is computed after aligning cluster indices with class labels using the Hungarian algorithm. The results of ACC are provided in the Appendix (see Fig. 3).

Fig. 2 and 3 summarize clustering performance across 9 datasets as the supervision level increases. Overall, CAST-LP achieves the most consistent improvement trend and ranks among the top methods on the majority of datasets in both ARI and ACC. This behavior matches the design goal of CAST-LP: instead of learning an unrestricted kernel transformation with high risk of overfitting, it performs a targeted, constraint-aligned kernel adaptation in a low-dimensional supervision-induced subspace, which preserves the original geometry outside the constraint-supported directions.

Image datasets. On COIL20 and YaleFace, CAST-LP shows clear and monotonic gains as more constraints are provided, and achieves the highest ARI (ACC) at medium-to-high supervision.

On Scene, CAST-LP exhibits particularly strong gains: it rapidly improves with limited supervision and remains the best performer across most supervision levels, suggesting that constraint-aligned scaling is effective at correcting structural mismatch in these datasets.

Text datasets. On 20Newsgroups, Cnae9 and Spambase, CAST-LP consistently improves with supervision and achieves the highest final ARI (ACC) among compared approaches. The advantage is most pronounced in ARI, indicating better pairwise consistency of assignments, which is aligned with the fact that the learning objective is driven directly by pairwise constraints.

Other datasets. On kr-vs-kp, CAST-LP yields the strongest improvement curves and achieves the best overall performance, especially under moderate-to-high supervision. On Gina-agnostic, CAST-LP remains competitive and matches the top performance at higher supervision. On HAR, CAST-LP performs on par with the best method and remains consistently strong across supervision levels.

Across diverse regimes (image, text, and tabular data), CAST-LP demonstrates a robust trend: performance improves smoothly as supervision increases, while remaining competitive with the strongest kernel DML baselines.

5.2 Runtime comparison and practical scalability

To evaluate practical efficiency, we report only the runtime of the *kernel learning stage*, corresponding to the steps of Algorithm 1. Kernel construction and downstream clustering are excluded, as these steps are shared across all methods in our experimental framework and are not affected by the learning formulation. This separation ensures that the empirical results directly reflect the complexity analysis in Section 3.5 and highlight the efficiency gains of the CAST-LP framework.

Table 2 reports the optimization runtime under three supervision levels (1%, 10%, 20%). Across all datasets and supervision ratios, CAST-LP is consistently the fastest method. On small and medium-sized datasets such as COIL20, YaleFace, and Cnae9, the optimization completes within a few hundredths of a second. On larger datasets, HAR and 20Newsgroups, CAST-LP remains efficient, while competing kernel DML methods incur substantially higher costs, often exceeding CAST-LP by several orders of magnitude depending on the supervision level.

Under the *sparsest* supervision regime (1%), CAST-LP achieves substantial speedups even against the *fastest* competitor: the speedup ranges from **33**× on Cnae9 to **259**× on kr-vs-kp (see the **Speedup (Min/Max)** row). Similar gaps hold on other datasets, e.g., **59**× on COIL20, **109**× on YaleFace and **83**× on 20Newsgroups. As supervision increases, the absolute runtime of CAST-LP grows monotonically on every dataset (e.g., from 0.46 s to 48.06 s on HAR, and from 2.31 s to 83.87 s on 20Newsgroups when moving from 1% to 20%), which is expected because our protocol increases the constraint budget with the labeled fraction (Section 4.3). Nevertheless, CAST-LP remains below all competitors across the entire supervision range, and the advantage remains substantial at 20% supervision (e.g., **8**× on 20Newsgroups, and **3**× on HAR; see Table 2).

Notably, these gains are obtained while using the same downstream clustering procedure and the same kernel initialization as all baselines, indicating that the runtime advantage is attributable to the CAST-LP learning formulation rather than shared overhead. The results validate the core claim of CAST-LP: the kernel learning stage scales primarily with the supervision budget (constraint volume) rather than the dataset size. In contrast, most kernel DML baselines optimize global kernel-level objectives whose cost scales with n or n^2, which explains their rapidly increasing runtime on larger datasets (e.g., HAR and 20Newsgroups). The results also points to a practical limitation: CAST-LP's advantage may narrow not because of n, but because the supervision-driven subspace itself grows.

Overall, CAST-LP achieves a favorable balance between expressiveness and computational cost: by confining metric adaptation to a constraint-induced subspace and solving for globally optimal weights via LP, it delivers strong practical speedups under limited supervision while maintaining competitive clustering performance.

6 Conclusion and Future Work

In this work, we presented CAST, a geometrically motivated approach to semi-supervised clustering that addresses the computational tractability of kernel learning stage. Unlike standard methods that learn a global transformation of the feature space, CAST operates on the principle of minimal intervention, modifying the metric only along supervision-induced subspace while preserving the original manifold structure elsewhere. Theoretical analysis and empirical results confirm that this targeted adaptation strategy effectively resolves structural mismatches without the prohibitive costs associated with full kernel matrix operations. By formulating

Table 2: Comparison of the runtime of various methods across 9 datasets with different supervision levels (1%, 10%, and 20%). Speedup (Min/Max) indicates the runtime comparison between the fastest and slowest competitors against CAST-LP.

Datasets	COIL20			YaleFace			Scene		
Supervision Level	1%	10%	20%	1%	10%	20%	1%	10%	20%
CAST-LP	**0.02**	**0.08**	**0.21**	**0.03**	**0.15**	**0.29**	**0.02**	**0.15**	**0.54**
SKMS	7.77	19.40	31.21	49.95	123.69	156.58	28.91	86.32	126.29
kDMLMJ	0.59	0.93	1.38	3.89	5.71	7.87	3.55	4.95	6.61
kKISSME	3.82	8.88	15.05	21.76	57.41	108.08	29.11	56.10	92.07
kDMLSKM	0.93	1.39	2.89	3.51	6.94	14.74	3.65	5.21	11.62
kGMML	1.20	1.40	1.89	5.24	6.30	6.98	4.59	5.87	8.28
kDMLCG	0.89	0.96	1.58	3.28	4.31	6.20	2.47	7.50	11.91
CCMKL	9.25	9.71	10.33	41.18	50.92	57.08	31.02	33.31	33.60
Speedup (Min/Max)	59/925	23/485	9/222	109/1665	28/824	21/539	123/1551	33/575	12/233
Datasets	Cnae9			Spambase			20Newsgroups		
Supervision Level	1%	10%	20%	1%	10%	20%	1%	10%	20%
CAST-LP	**0.01**	**0.05**	**0.08**	**0.09**	**0.91**	**3.94**	**2.31**	**17.94**	**83.87**
SKMS	3.86	10.72	20.18	147.39	319.06	516.72	3003.63	4958.18	7642.64
kDMLMJ	0.39	0.57	0.83	29.81	32.17	57.65	645.02	662.47	700.34
kKISSME	1.87	6.71	9.05	108.13	264.22	495.93	1538.06	2763.05	6423.96
kDMLSKM	0.50	1.17	1.95	18.89	35.09	68.18	192.22	436.94	932.99
kGMML	0.69	0.82	0.79	38.35	41.92	53.39	657.41	695.99	739.34
kDMLCG	0.33	0.60	0.71	20.63	28.37	38.63	543.50	598.46	673.50
CCMKL	2.57	3.40	4.06	226.16	230.02	266.67	2155.64	2458.15	2575.15
Speedup (Min/Max)	33/386	11/214	8/252	209/2512	31/350	9/131	83/1300	24/276	8/91
Datasets	kr-vs-kp			Gina-agnostic			HAR		
Supervision Level	1%	10%	20%	1%	10%	20%	1%	10%	20%
CAST-LP	**0.02**	**0.27**	**1.19**	**0.03**	**0.35**	**1.48**	**0.46**	**8.60**	**48.06**
SKMS	36.58	111.94	187.67	63.47	190.28	318.11	472.12	846.70	1719.61
kDMLMJ	6.76	11.19	20.90	6.11	12.74	14.31	133.05	139.84	151.40
kKISSME	31.75	80.60	129.34	40.65	97.05	153.01	419.00	752.65	1207.25
kDMLSKM	5.19	13.52	25.09	5.06	11.62	20.29	104.99	142.85	161.37
kGMML	12.85	15.05	17.85	9.92	13.35	17.92	119.40	154.32	179.13
kDMLCG	8.26	10.47	12.25	7.62	14.87	15.62	107.40	132.30	155.02
CCMKL	77.41	81.14	119.01	133.38	138.26	142.68	1078.44	1173.56	1246.84
Speedup (Min/Max)	259/3870	38/414	10/157	168/4446	33/354	9/214	228/2344	15/136	3/35

the weight learning as a Linear Program over the intrinsic rank of the constraint set, CAST achieves deterministic convergence and practical scalability for large datasets where traditional kernel methods fail.

To further improve scalability in dense constraint scenarios, future research could investigate the integration of CAST with approximation methods like Nyström or Random Features, as well as the strategic combination of multiple subspaces.

Acknowledgments

This work was supported by the Finnish Doctoral Program Network in Artificial Intelligence, AI-DOC (decision number VN/3137/2024-OKM-6).

References

[1] Sultan Alfarhood and Meshal Alfarhood. 2024. CAML: A Context-Aware Metric Learning approach for improved recommender systems. *Alexandria Engineering Journal* 100 (2024), 53–60. doi:10.1016/j.aej.2024.05.025

[2] Saket Anand, Sushil Mittal, Oncel Tuzel, and Peter Meer. 2014. Semi-supervised kernel mean shift clustering. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 36, 6 (2014), 1201–1215. doi:10.1109/TPAMI.2013.190

[3] Arthur Asuncion and David Newman. 2007. UCI machine learning repository.

[4] Ghojogh Benyamin, Ghodsi Ali, Karray Fakhri, and Crowley Mark. 2022. Spectral, Probabilistic, and Deep Metric Learning: Tutorial and Survey. arXiv:arXiv:2201.09267

[5] Benedikt Boecking, Vincent Jeanselme, and Artur Dubrawski. 2024. Constrained clustering and multiple kernel learning without pairwise constraint relaxation. *Advances in Data Analysis and Classification* 18 (2024), 309–324. doi:10.1007/s11634-022-00507-5

[6] Yifan Chen, Ethan N. Epperly, Joel A. Tropp, and Robert J. Webber. 2025. Randomly Pivoted Cholesky: Practical Approximation of a Kernel Matrix with Few Entry Evaluations. *Communications on Pure and Applied Mathematics* (2025). doi:10.1002/cpa.22234

[7] Petros Drineas and Michael W. Mahoney. 2005. On the Nyström Method for Approximating a Gram Matrix for Improved Kernel-Based Learning. *Journal of Machine Learning Research* 6 (2005), 2153–2175.
[8] Yu Feng, Weixuan Liang, Xinhang Wan, Jiyuan Liu, Suyuan Liu, Qian Qu, Renxiang Guan, Huiying Xu, and Xinwang Liu. 2025. Incremental Nyström-based Multiple Kernel Clustering. In *Proceedings of the AAAI Conference on Artificial Intelligence*. Article 1852, 9 pages. doi:10.1609/aaai.v39i16.33825
[9] Zixin Feng, Teligeng Yun, Yu Zhou, Ruirui Zheng, and Jianjun He. 2023. Kernel Geometric Mean Metric Learning. *Applied Sciences* 13, 21 (2023), 12047. doi:10.3390/app132112047
[10] Pasi Fränti and Sami Sieranoja. 2024. Clustering Accuracy. *Applied Computing and Intelligence* 4, 1 (2024), 24–44. doi:10.3934/aci.2024003
[11] Athinodoros S Georghiades, Peter N Belhumeur, and David J Kriegman. 2001. From Few to Many: Illumination Cone Models for Face Recognition Under Variable Lighting and Pose. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 23 (2001), 643–660. Issue 6. doi:10.1109/34.927464
[12] Benyamin Ghojogh, Fakhri Karray, and Mark Crowley. 2019. Fisher and kernel Fisher discriminant analysis: Tutorial. *arXiv preprint arXiv:1906.09436* (2019). arXiv:1906.09436
[13] Prateek Jain, Brian Kulis, Jason V. Davis, and Inderjit S. Dhillon. 2012. Metric and kernel learning using a linear transformation. *J. Mach. Learn. Res.* 13 (2012), 519–547.
[14] Minchul Kim, Anil K. Jain, and Xiaoming Liu. 2022. AdaFace: Quality Adaptive Margin for Face Recognition. In *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 18729–18738. https://api.semanticscholar.org/CorpusID:247940176
[15] Martin Kostinger, Martin Hirzer, Paul Wohlhart, Peter M. Roth, and Horst Bischof. Jun. 2012. Large scale metric learning from equivalence constraints. In *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.* 2288–2295. doi:10.1109/CVPR.2012.6247939
[16] Haanvid Lee, Tri Wahyu Guntara, Jongmin Lee, Yung-Kyun Noh, and Kee-Eung Kim. 2024. Kernel Metric Learning for In-Sample Off-Policy Evaluation of Deterministic RL Policies. In *International Conference on Learning Representations (ICLR)*. doi:10.48550/arXiv.2405.18792
[17] Kuang Chih Lee, Jeffrey Ho, and David J Kriegman. 2005. Acquiring linear subspaces for face recognition under variable lighting. *IEEE Trans. Pattern Anal. Mach. Intell.* 27 (2005), 684–698. Issue 5. doi:10.1109/TPAMI.2005.92
[18] Christopher Liao, Theodoros Tsiligkaridis, and Brian Kulis. 2023. Supervised Metric Learning to Rank for Retrieval via Contextual Similarity Optimization. In *Proceedings of the 40th International Conference on Machine Learning*, Vol. 202. 20906–20938.
[19] Fanghui Liu, Xiaolin Huang, Yudong Chen, and Johan A. K. Suykens. 2022. Random Features for Kernel Approximation: A Survey on Algorithms, Theory, and Beyond. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 44, 10 (2022), 7128–7148. doi:10.1109/TPAMI.2021.3097011
[20] Yongheng Liu, Xiangdong Gao, James Zhiqing Wen, and Huiyuan Luo. 2023. Unsupervised Image Anomaly Detection and Localization in Industry Based on Self-Updated Memory and Center Clustering. *IEEE Transactions on Instrumentation and Measurement* 72 (2023). doi:10.1109/TIM.2023.3271754
[21] Sameer A Nene, Shree K Nayar, and Hiroshi Murase. Tech. Rep., Columbia Univ. 1996. Columbia Object Image Library (COIL-20). (Tech. Rep., Columbia Univ. 1996).
[22] Bac Nguyen and Bernard De Baets. 2019. Kernel-Based Distance Metric Learning for Supervised k-Means Clustering. *IEEE Transactions on Neural Networks and Learning Systems* 30, 10 (2019), 3084–3095. doi:10.1109/TNNLS.2018.2890021
[23] Bac Nguyen and Bernard De Baets. Feb. 2019. Kernel distance metric learning using pairwise constraints for person re-identification. *IEEE Trans. Image Process.* 28, 2 (Feb. 2019), 589–600. doi:10.1109/TIP.2018.2870941
[24] Bac Nguyen, Carlos Morell, and Bernard De Baets. 2017. Supervised distance metric learning through maximization of the Jeffrey divergence. *Pattern Recognition* 64 (2017), 215–225. doi:10.1016/j.patcog.2016.11.010
[25] Tetsushi Ohki, Yuya Sato, Masakatsu Nishigaki, and Koichi Ito. 2024. LabellessFace: Fair Metric Learning for Face Recognition without Attribute Labels. In *2024 IEEE International Joint Conference on Biometrics (IJCB)*. 1–8. doi:10.1109/IJCB62174.2024.10744511
[26] Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, Jake Vanderplas, Alexandre Passos, David Cournapeau, Matthieu Brucher, Matthieu Perrot, and Édouard Duchesnay. 2011. Scikit-learn: Machine Learning in Python. *The Journal of Machine Learning Research* 12 (2011), 2825–2830.
[27] Ali Rahimi and Benjamin Recht. 2007. Random Features for Large-Scale Kernel Machines. In *Advances in Neural Information Processing Systems 20 (NeurIPS 2007)*.
[28] Mingwen Shao and Zilu Peng. 2024. Distance metric-based learning for long-tail object detection. *Image and Vision Computing* 142 (2024), 104888. doi:10.1016/j.imavis.2023.104888
[29] Douglas Steinley. 2004. Properties of the Hubert-Arable adjusted rand index. *Psychol. Methods* 9, 3 (2004), 386.
[30] Gokcan Tatli, Yi Chen, Blake Mason, Robert D. Nowak, and Ramya Korlakai Vinayak. 2025. Metric Learning in an RKHS. In *Proceedings of the Forty-first Conference on Uncertainty in Artificial Intelligence (Proceedings of Machine Learning Research, Vol. 286)*. 4145–4164. doi:10.48550/arXiv.2508.04476
[31] Pauli Virtanen, Ralf Gommers, Travis E Oliphant, et al. 2020. SciPy 1.0: fundamental algorithms for scientific computing in Python. *Nat. Methods* 17, 3 (2020), 261–272.
[32] Nan Wang, Xibin Zhao, Yu Jiang, and Yue Gao. 2018. Iterative Metric Learning for Imbalance Data Classification. In *Proceedings of the 27th International Joint Conference on Artificial Intelligence*. 2805–2811.
[33] Tianbao Yang, Yu-feng Li, Mehrdad Mahdavi, Rong Jin, and Zhi-Hua Zhou. 2012. Nyström Method vs Random Fourier Features: A Theoretical and Empirical Comparison. In *Proc. Int. Conf. Neural Inf. Process. Syst.* 476–484.
[34] Pourya Habib Zadeh, Reshad Hosseini, and Suvrit Sra. 2016. Geometric Mean Metric Learning. In *Proc. Int. Conf. Mach. Learn.* 2464–2471.
[35] Shijie Zhao, Liang Cai, Fanshuai Meng, and Ronghua Yang. 2025. Discriminative distance metric learning via class-center guidance. *Appl. Soft Comput.* 182 (2025), 113552. doi:10.1016/J.ASOC.2025.113552

A Sensitivity Analysis and Ablation Studies on LP Solver

We analyze the sensitivity to the choice of the LP solver. We conducted a controlled ablation study on three synthetic regimes generated with `make_classification` [26].

The regimes are designed to decouple sample size (N) from feature dimensionality (D): (A) a baseline setting with $(N, D) = (5{,}000, 100)$, (B) a high-dimensional setting with $(N, D) = (5{,}000, 5{,}000)$, and (C) a larger-sample setting with $(N, D) = (50{,}000, 100)$.

All datasets use $n_{\text{classes}} = 10$ and $n_{\text{informative}} = 50$ to maintain comparable class structure across regimes. We vary the supervision ratio $p \in \{0.05, 0.10, 0.15\}$ and solve the resulting standardized LP using four solvers available through SciPy `linprog` [31]: `revised simplex`, `interior-point`, `highs-ds`, and `highs-ipm`.

For each configuration, we report the mean and standard deviation of the wall-clock runtime over 10 trials (different random seeds), measured for the LP optimization (Step 4 in Algorithm 1).

A.0.1 Runtime sensitivity across regimes. Table 3 summarizes the runtime results. Two consistent trends emerge.

First, increasing the ambient dimensionality from $D = 100$ to $D = 5{,}000$ (Dataset A → B) has a limited effect on LP runtime for the legacy SciPy solvers: `interior-point` remains within 4–12 ms and `revised simplex` within 8–20 ms across the tested p values. The HiGHS solvers operate at a higher runtime level in these regimes (approximately 24–196 ms), with runtime increasing as p grows.

Second, scaling the sample size by 10× (Dataset A → C) substantially increases runtime for all solvers, but the magnitude depends strongly on the solver implementation. The HiGHS solvers exhibit a pronounced overhead in the large-N regime, increasing from tens–hundreds of milliseconds (Datasets A/B) to seconds on Dataset C (e.g., $\approx$ 1.8 s at $p = 0.05$ and $\approx$ 17.1 s at $p = 0.15$). In contrast, the legacy SciPy solvers remain in the millisecond range: `revised simplex` increases from 13.8–19.8 ms (Dataset A) to 24.4–39.9 ms (Dataset C), while `interior-point` increases from 5.2–12.1 ms to 36.0–142.1 ms.

A.0.2 Numerical consistency of the optimum. To verify that solver choice affects only efficiency (and not the optimized solution), we additionally examine the numerical consistency of the final objective values. Table 3 reports, for each (dataset, p), the mean objective value across solvers and the range (max–min) of solver-specific objective values. The ranges are very small across all settings, from

Table 3: LP solver performance and numerical consistency on three synthetic regimes (10 trials). Times for LP optimization are in milliseconds (mean ± std). Objective (Obj) and Range (max–min) demonstrate solver consistency.

		Optimization Time (ms)				Numerical Consistency	
Dataset	***p***	**revised simplex**	**interior-point**	**HiGHS-DS**	**HiGHS-IPM**	**Obj**	**Obj. Range**
A: Baseline (5k, 100)	0.05	13.75±1.37	5.17±0.91	23.83±2.10	24.82±3.62	125.22	6.0×10^{-6}
	0.10	19.01±2.19	8.64±0.84	96.19±9.14	87.21±3.86	23.11	2.0×10^{-6}
	0.15	19.75±1.48	12.14±1.92	190.69±12.11	172.48±1.69	12.41	3.0×10^{-6}
B: High D (5k, 5k)	0.05	7.83±0.73	4.17±1.14	26.84±3.43	26.19±4.60	2.55	0.0
	0.10	8.70±1.98	6.06±0.54	82.50±1.65	87.50±3.49	1.13	0.0
	0.15	9.27±1.63	8.09±1.08	195.59±12.37	173.81±3.18	0.87	0.0
C: Large N (50k, 100)	0.05	24.42±2.27	36.04±4.71	1839.28±7.95	1822.80±8.90	4.37	8.0×10^{-5}
	0.10	32.03±3.64	87.48±7.18	7518.02±37.68	7478.50±42.66	2.83	3.56×10^{-4}
	0.15	39.89±6.90	142.05±13.87	17067.62±51.22	17071.82±42.83	2.12	2.36×10^{-3}

exactly zero in Dataset B to at most 2.361×10^{-3} in the largest regime (Dataset C, $p = 0.15$). This indicates that all solvers converge to essentially the same optimum up to numerical precision, and the observed differences are attributable to runtime and implementation overhead rather than solution quality.

Across the baseline and high-dimensional regimes (Datasets A and B), `interior-point` is consistently the fastest solver. In the large-N regime (Dataset C), `revised simplex` becomes faster than `interior-point`, while both remain orders of magnitude faster than the HiGHS solvers. Based on these results, we use `interior-point` as the default solver, and switch to `revised simplex` when the constraint volume becomes large, to keep the LP stage computationally negligible relative to shared costs such as kernel construction and downstream clustering.

B ACC performance

Fig. 3 reports the clustering performance evaluated by ACC.

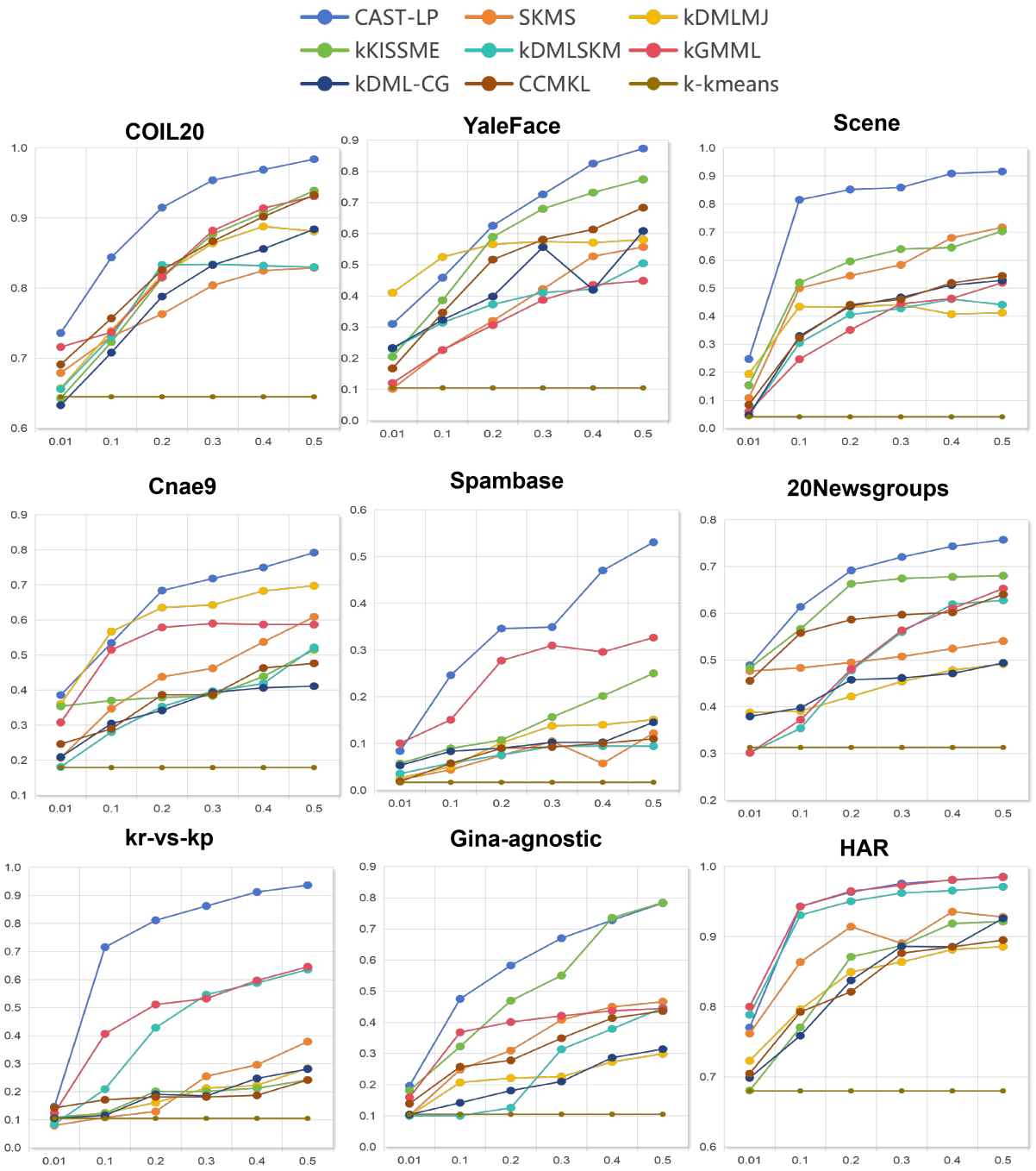


Figure 3: Comparison of clustering performance measured by ACC across 9 datasets. Each plot depicts the clustering results obtained by running k-means with learned kernel metrics for each dataset. The x-axis represents the supervision level p, while the y-axis displays the corresponding ACC scores.