Adaptive Distance Metric Selection for Supervised Classification

Krishnan Kumaran and Dimitri J. Papageorgiou

Corporate Strategic Research

ExxonMobil Research and Engineering Company

1545 Route 22 East, Annandale, NJ 08801 USA

dimitri.j.papageorgiou@exxonmobil.com

February 14, 2017

1 Introduction and motivation

Clustering (also known as unsupervised classification) and Classification (commonly understood to be supervised, i.e., based on training data) are common ways of performing data analysis to perform a range of functions like anomaly detection and diagnosis, data segmentation and model development/refinement. Consequently, there is a large body of research on these topics (see for example [1] and references therein for a survey of currently used methods) offering different solutions ranging from K-means, spectral methods and other more complex methods for Clustering, as well as Support Vector Machines, Artificial Neural Networks (ANN), their non-linear Kernel-based variants and others for Classification.

All clustering methods require a similarity/distance metric between data points. While there are typically a few reasonable choices for most data, the results can depend strongly on this choice. The main goal of this project is to extend existing clustering work (e.g., [2,3,4]) to a semi-supervised classification method that is capable of learning the optimal distance/similarity metric directly from training data. The clustering problem can be formulated as an optimization problem (a mixed-integer linear program in its simplest form) that can be hard to solve for realistic data sizes. This approach using an adaptive distance metric has the potential to provide results of better quality than state-of-the-art techniques like Support Vector Machines.

2 Problem statement

Suppose we are given N points $\mathbf{v}_i \in \mathbb{R}^p$ with class membership $\mathcal{C}_i$ for $i \in \mathcal{N} = \{1, \ldots, N\}$. Our goal is to find a distance metric $\mathbb{D} : \mathbb{R}^p \mapsto \mathbb{R}$ such that the following conditions hold:

1. $\mathbb{D}(\mathbf{v}_i, \mathbf{v}_j) = \mathbf{a}^\top(\mathbf{v}_i - \mathbf{v}_j) + (\mathbf{v}_i - \mathbf{v}_j)^\top \mathbf{B}(\mathbf{v}_i - \mathbf{v}_j) \quad \forall i, j \in \mathcal{N}$, for some $\mathbf{a} \in \mathbb{R}^p$ and $\mathbf{B} \in \mathbb{R}^{p \times p}$ and symmetric
2. $\mathbb{D}(\mathbf{v}_i, \mathbf{v}_j) \geq 0 \quad \forall i, j \in \mathcal{N}$
3. $\min_{j \in \mathcal{C}_i} \mathbb{D}(\mathbf{v}_i, \mathbf{v}_j) < \min_{k \notin \mathcal{C}_i} \mathbb{D}(\mathbf{v}_i, \mathbf{v}_j) \quad \forall i \in \mathcal{N}$

Note that this is not a true "distance metric" since symmetry and the triangle property are not enforced. The first condition could be extended to include higher order terms. Let $\boldsymbol{\delta}_{ij} = \mathbf{v}_i - \mathbf{v}_j$ for all $i, j \in \mathcal{N}$. Finding such a distance metric can be expressed as the following optimization problem:

$$\max_{\lambda,\mathbf{a},\mathbf{B},\mathbf{d}} \quad \lambda \tag{1a}$$

$$\text{s.t.} \quad \min_{j\in\mathcal{C}_i} d_{ij} + \lambda \leq \min_{k\notin\mathcal{C}_i} d_{ik} \qquad \forall i \in \mathcal{N} \tag{1b}$$

$$d_{ij} = \mathbf{a}^\top \boldsymbol{\delta}_{ij} + \boldsymbol{\delta}_{ij}^\top \mathbf{B} \boldsymbol{\delta}_{ij} \qquad \forall i, j \in \mathcal{N} \tag{1c}$$

$$d_{ij} \geq 0 \qquad \forall i, j \in \mathcal{N} \tag{1d}$$

$$\mathbf{a} \in \mathbb{R}^p \tag{1e}$$

$$\mathbf{B} \in \mathbb{R}^{p\times p}, \text{symmetric} \tag{1f}$$

Let $\lambda_i = [\min_{k\notin\mathcal{C}_i} d_{ik}] - [\min_{j\in\mathcal{C}_i} d_{ij}]$ be the difference between the separation/distance from point i to its nearest nonneighbor and the separation/distance from point i to its nearest neighbor. Ideally, we would like $\lambda_i > 0$ for all i. However, this might not be possible when restricted to a set of distance functions. In formulation (1), we would like to maximize the minimum separation over all points as can be seen by re-writing constraint (1b) as

$$\lambda \leq \lambda_i \qquad \forall i \in \mathcal{N}$$

and noting that the objective function is to maximize λ.

2.1 Potential loss functions

Let $L(\boldsymbol{\lambda}) = L(\lambda_1, \ldots, \lambda_n)$ be a loss function based on the separation of each point i. Let $f = -L$ be the negative loss function. In formulation (1), we are interested in maximizing the function

$$f(\boldsymbol{\lambda}) = -L(\boldsymbol{\lambda}) = \min_{i\in\mathcal{N}} \lambda_i \ .$$

Of course, other loss functions are possible and should potentially be considered.

Option 1: Assume f is linear and separable:

$$f(\boldsymbol{\lambda}) = \sum_{i\in\mathcal{N}} f_i(\lambda_i) \tag{2}$$

Option 2: Assume f is piecewise linear increasing and separable:

$$f^{\text{PWL-Cont}}(\lambda_i) = \begin{cases} \alpha\lambda_i & \lambda_i \geq 0 \\ \beta\lambda_i & o.w. \end{cases} \text{ with } \alpha \leq \beta \tag{3}$$

Option 3: Assume f is piecewise linear, increasing up to a and separable:

$$f^{\text{PWL-DisCont}}(\lambda_i) = \begin{cases} \alpha\lambda_i & \lambda_i \geq 0 \\ \beta & o.w. \end{cases} \text{ with} \tag{4}$$

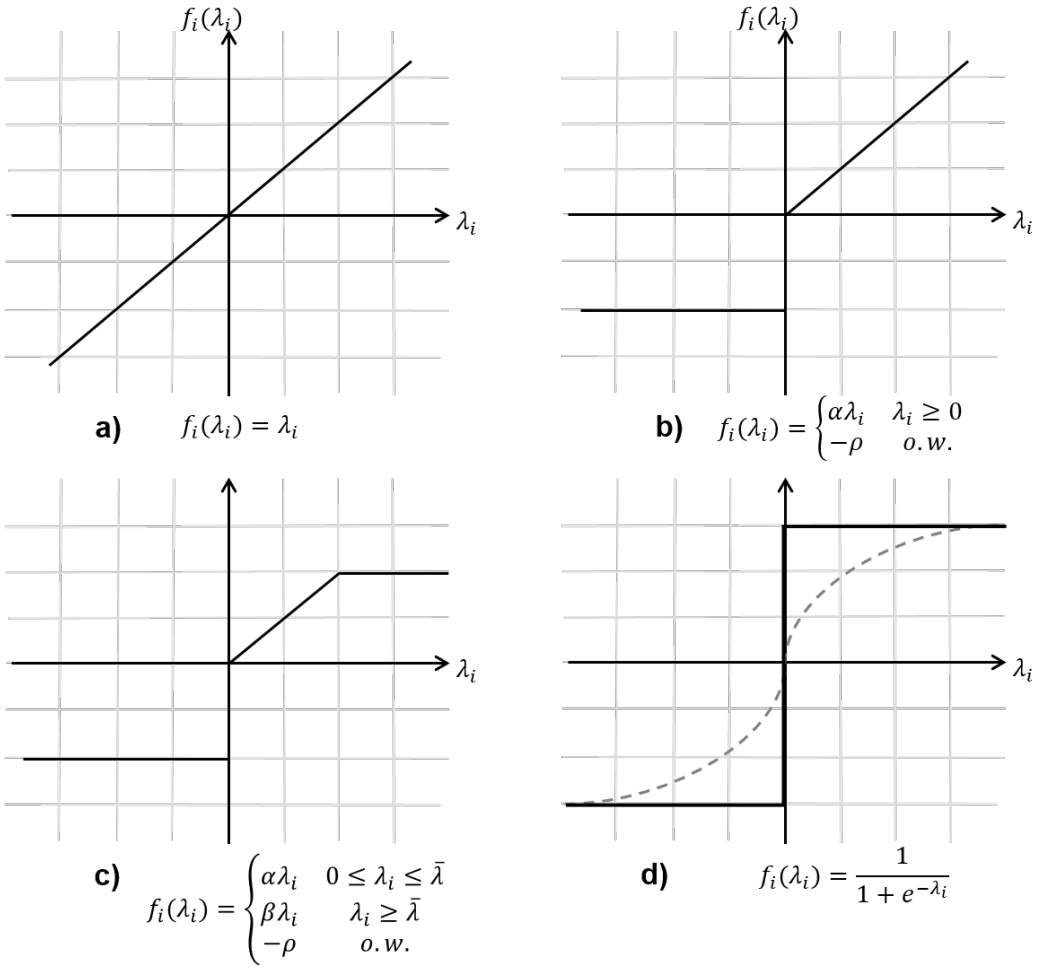

Figure 1: Potential loss functions

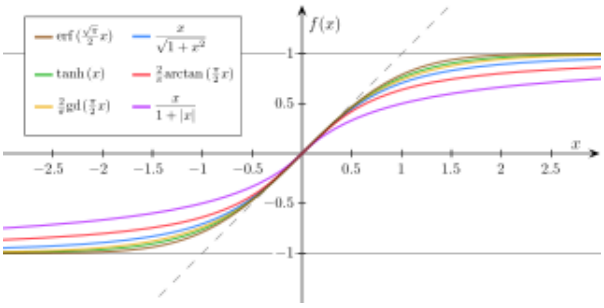

Figure 2: Sigmoid function. [Source: wikipedia]

2.2 Imposing a $K > 1$ nearest neighbor condition

Let $d_{i(j)}$ denote the distance of the jth closest point to point $i \in \mathcal{N}$, i.e., $d_{i(1)} \leq d_{i(2)} \cdots \leq d_{i(|\mathcal{C}_i|)}$. Then, inequality (1b) can be rewritten as

$$d_{i(1)} + \lambda \leq d_{ik} \forall i \in \mathcal{N}, k \notin \mathcal{C}_i \tag{5}$$

A more restrictive requirement

$$d_{i(t)} + \lambda \leq d_{ik} \forall i \in \mathcal{N}, k \notin \mathcal{C}_i \tag{6}$$

The above model can be generalized so that the requirement in inequality (1b) holds for the K closest points $j \in \mathcal{C}_i$, and

3 A linear programming approach to determine the existence of a feasible distance metric

Let $\mathcal{F}$ denote the feasible region associated with the distance metric parameters, i.e.,

$$\mathcal{F} = \{(\mathbf{a}, \mathbf{B}, \mathbf{d}) \in \mathbb{R}^p \times \mathbb{R}^{p \times p} \times \mathbb{R}_+^{N \times N} : \mathbf{B} \text{ symmetric}, d_{ij} = \mathbf{a}^\top \boldsymbol{\delta}_{ij} + \boldsymbol{\delta}_{ij}^\top \mathbf{B} \boldsymbol{\delta}_{ij} \quad \forall i, j \in \mathcal{N}\} \tag{7}$$

or

$$\mathcal{F} = \Bigg\{ \mathbf{a}, \mathbf{B}, \mathbf{d} : \tag{8a}$$

$$d_{ij} = \sum_{k=1}^{p} \delta_{ijk} a_k + \sum_{k=1}^{p} \delta_{ijk}^2 b_{kk} + 2 \sum_{k=1}^{p-1} \sum_{\ell=k+1}^{p} \delta_{ijk} \delta_{ij\ell} b_{k\ell} \qquad \forall i, j \in \mathcal{N} \tag{8b}$$

$$0 \leq d_{ij} \leq 1 \qquad \forall i, j \in \mathcal{N} \tag{8c}$$

$$\mathbf{a} \in \mathbb{R}^p \tag{8d}$$

$$b_{k\ell} \in \mathbb{R} \qquad \forall k = 1, \ldots, p, \ell = k, \ldots, p \Bigg\} \tag{8e}$$

The existence of a distance metric satisfying conditions 1-3 can be expressed as the following feasibility (nonlinear optimization) problem, where $\epsilon > 0$ is a given parameter:

$$\max_{\mathbf{a}, \mathbf{B}, \mathbf{d}} \quad 0 \tag{9a}$$

$$\text{s.t.} \quad \min_{j \in \mathcal{C}_i} d_{ij} + \epsilon \leq \min_{k \notin \mathcal{C}_i} d_{ik} \qquad \forall i \in \mathcal{N} \tag{9b}$$

$$(\mathbf{a}, \mathbf{B}, \mathbf{d}) \in \mathcal{F} \tag{9c}$$

Note that if we constrain the distances to be bounded (e.g., $d_{ij} \in [0, 1]$), problem (9) is not guaranteed to be feasible for an arbitrary choice of ϵ. To see this, let $\epsilon > 0$ and consider an example with three points and two classes. Point 1 is located at (0,0), point 2 at (1,1), and point 3 at $(\epsilon/2, \epsilon/2)$. Points 1 and 2 belong to class 1, while point 3 belongs to class 2. There is no way to create a distance metric that guarantees that $d_{13} > \epsilon$ and $d_{23} > \epsilon$.

There are several ways to proceed.

3.1 Multi-Objective approach

Let W be a positive scalar. We could solve the following LP that maximizes both ϵ and the minimum distance variables t_i:

$$\max_{\mathbf{a}, \mathbf{B}, \mathbf{d}, \epsilon, \mathbf{t}} \quad W\epsilon + \sum_{i \in \mathcal{N}} t_i \tag{10a}$$

$$\text{s.t.} \quad t_i + \epsilon \leq d_{ik} \qquad \forall i \in \mathcal{N}, k \in \mathcal{N} \backslash \mathcal{C}_i \tag{10b}$$

$$t_i \leq d_{ij} \qquad \forall i \in \mathcal{N}, j \in \mathcal{C}_i \tag{10c}$$

$$t_i \geq 0 \qquad \forall i \in \mathcal{N} \tag{10d}$$

$$(8) \tag{10e}$$

Proposition 1 *Problem* (10) *is always feasible.*

Proof The origin is always a feasible solution. That is, set $\mathbf{a} = \mathbf{0}, \mathbf{B} = \mathbf{0}, \mathbf{d} = \mathbf{0}, \epsilon = 0, \mathbf{t} = \mathbf{0}$. □

3.2 Iterative LP approach

First solve the following problem to determine the maximum non-neighbor separation ϵ that is possible.

$$\max_{\mathbf{a},\mathbf{B},\mathbf{d},\epsilon} \quad \epsilon \tag{11a}$$

$$\text{s.t.} \quad \epsilon \leq d_{ik} \qquad \forall i \in \mathcal{N}, k \in \mathcal{N} \backslash \mathcal{C}_i \tag{11b}$$

$$(8) \tag{11c}$$

Let $\epsilon^{\max}$ be the maximum non-neighbor separation, i.e., the optimal value of ϵ in (11). Note that this problem is identical to problem (10) after omitting the $\mathbf{t}$ decision variables. Hence, it is guaranteed to be feasible.

Given $\epsilon \in (0, \epsilon^{\max}]$, model (1) can be expressed as a linear programming feasibility problem:

$$\max_{\mathbf{a},\mathbf{B},\mathbf{d},\mathbf{t}} \quad \sum_{i \in \mathcal{N}} t_i \tag{12a}$$

$$\text{s.t.} \quad t_i + \epsilon \leq d_{ik} \qquad \forall i \in \mathcal{N}, k \in \mathcal{N} \backslash \mathcal{C}_i \tag{12b}$$

$$t_i \leq d_{ij} \qquad \forall i \in \mathcal{N}, j \in \mathcal{C}_i \tag{12c}$$

$$t_i \geq 0 \qquad \forall i \in \mathcal{N} \tag{12d}$$

$$(8) \tag{12e}$$

For $\epsilon \in (0, \epsilon^{\max}]$, model (12) is always feasible since the origin $\mathbf{0}$ is always feasible. Note that t_i represents the distance to point i's nearest neighbor since constraints (12c) ensure that $t_i \leq \min_{j \in \mathcal{C}_i} d_{ij}$, while the objective function encourages $t_i = \min_{j \in \mathcal{C}_i} d_{ij}$ for all i. Meanwhile, constraints (12b) ensure that the separation $[\min_{k \in \mathcal{N} \backslash \mathcal{C}_i} d_{ik}] - [\min_{j \in \mathcal{C}_i} d_{ij}]$ is at least ϵ. If model (12) returns a feasible solution with $t_i^* = \min_{j \in \mathcal{C}_i} d_{ij}^* > 0$ for all $i \in \mathcal{N}$, then the distance metric satisfies requirements 1-3. Otherwise there exists a subset $\mathcal{O}$ of outlier points such that $0 \leq t_i^* < \min_{j \in \mathcal{C}_i} d_{ij}^*$ for all $i \in \mathcal{O}$, which means that no such distance metric guaranteeing a minimum separation exists for the choice of ϵ.

The following algorithm can be used to explore potential distance metrics for different levels of separation.

Algorithm 1 Iterative LP Heuristic

1: Solve model (11). Let $\epsilon^{\max}$ be the optimal objective function value.
2: Let $\mathcal{E} = \{i/\epsilon^{\max} : i = 1, \ldots, I\}$ (e.g., $I = 10$)
3: **for** $\epsilon \in \mathcal{E}$ **do**
4: Solve model (12).
5: Store the outliers and distance metric
6: **end for**

4 Mixed-Integer Linear Optimization Approaches

4.1 Max-Min Approach: Maximize the minimum separation with outliers

Model (1) can be expressed as a huge mixed-integer linear programming problem:

$$\max_{\boldsymbol{\lambda},\mathbf{B},\mathbf{d},\mathbf{w},\mathbf{y},\mathbf{z}} \quad f(\boldsymbol{\lambda}) = \sum_{i\in\mathcal{N}} f_i(\lambda_i) = \sum_{i\in\mathcal{N}} (\lambda_i - \rho z_i) \tag{13a}$$

$$\text{s.t.} \quad \sum_{j\in\mathcal{C}_i} w_{ij} + \lambda_i \le d_{ik} + M_{ik} z_i \qquad \forall i \in \mathcal{N}, k \in \mathcal{N}\backslash\mathcal{C}_i \tag{13b}$$

$$\lambda_i \le 1 - z_i \qquad \forall i \in \mathcal{N} \tag{13c}$$

$$w_{ij} \le d_{ij} \qquad \forall i \in \mathcal{N}, j \in \mathcal{C}_i \tag{13d}$$

$$w_{ij} \le y_{ij} \qquad \forall i \in \mathcal{N}, j \in \mathcal{C}_i \tag{13e}$$

$$w_{ij} \ge d_{ij} + y_{ij} - 1 \qquad \forall i \in \mathcal{N}, j \in \mathcal{C}_i \tag{13f}$$

$$\sum_{j\in\mathcal{C}_i} y_{ij} = 1 \qquad \forall i \in \mathcal{N} \tag{13g}$$

$$d_{ij} = \sum_{k=1}^{p} \delta_{ijk}^2 b_{kk} + 2\sum_{k=1}^{p-1}\sum_{\ell=k+1}^{p} \delta_{ijk}\delta_{ij\ell} b_{k\ell} \qquad \forall i,j \in \mathcal{N} \tag{13h}$$

$$b_{k\ell} \in \mathbb{R} \qquad \forall k = 1,\ldots,p, \ell = k,\ldots,p \tag{13i}$$

$$d_{ij} \ge 0 \qquad \forall i,j \in \mathcal{N} \tag{13j}$$

$$w_{ij} \ge 0 \qquad \forall i \in \mathcal{N}, j \in \mathcal{C}_i \tag{13k}$$

$$y_{ij} \in \{0,1\} \qquad \forall i \in \mathcal{N}, j \in \mathcal{C}_i \tag{13l}$$

$$z_i \in \{0,1\} \qquad \forall i \in \mathcal{N} \tag{13m}$$

$$\lambda_i \ge 0 \qquad \forall i \in \mathcal{N} \tag{13n}$$

Model (13) assumes a linear reward for positive separation and a fixed (constant) penalty ρ for each outlier. Outliers are handled through the binary z_i decision variables. If $z_i = 1$, then constraint (13c) ensures that the separation λ_i is 0 and hence ignored; otherwise (if $z_i = 0$), this constraint is effectively ignored. Constraints (13d)-(13f) are the so-called McCormick envelopes needed to linearize the bilinear equation $w_{ij} = d_{ij}y_{ij}$.

TODO: Modify to incorporate assumption that $0 \le d_{ij} \le 1$ for all i and j.

5 Generalized disjunctive programming (GDP) formulations

This section presents several generalized disjunctive programming formulations.

Robust GDP formulation with no outliers

$$\max_{\lambda,\mathbf{a},\mathbf{B},\mathbf{d}} \quad \lambda \tag{14a}$$

$$\text{s.t.} \quad \bigvee_{j\in\mathcal{C}_i} [d_{ij} + \lambda \leq d_{ik} \quad \forall k \notin \mathcal{C}_i] \qquad \forall i \in \mathcal{N} \tag{14b}$$

$$(\mathbf{a}, \mathbf{B}, \mathbf{d}) \in \mathcal{F} \tag{14c}$$

Constraints (14b) read: "the distance between point i and its in-class neighbor $j_1 \in \mathcal{C}_i$ plus the separation λ must not exceed the distance between point i and all of its non-neighbors" or "the distance between point i and its in-class neighbor $j_2 \in \mathcal{C}_i$ plus λ must not exceed ... " or ...

Robust GDP formulation with outliers

$$\max_{\lambda,\mathbf{a},\mathbf{B},\mathbf{d}} \quad \lambda - \rho \sum_{i\in\mathcal{N}} z_i \tag{15a}$$

$$\text{s.t.} \quad \left[\bigvee_{j\in\mathcal{C}_i} [d_{ij} + \lambda \leq d_{ik} \quad \forall k \notin \mathcal{C}_i] \right] \veebar [z_i = 1] \qquad \forall i \in \mathcal{N} \tag{15b}$$

$$(\mathbf{a}, \mathbf{B}, \mathbf{d}) \in \mathcal{F}, \mathbf{z} \in \mathbb{R}^N_+ \tag{15c}$$

GDP formulation with outliers

$$\max_{\boldsymbol{\lambda},\mathbf{a},\mathbf{B},\mathbf{d}} \quad \sum_{i\in\mathcal{N}} \lambda_i \tag{16a}$$

$$\text{s.t.} \quad \left[\bigvee_{j\in\mathcal{C}_i} [d_{ij} + \lambda_i \leq d_{ik} \quad \forall k \notin \mathcal{C}_i; \lambda_i \geq 0] \right] \veebar [\lambda_i = -\rho] \qquad \forall i \in \mathcal{N} \tag{16b}$$

$$(\mathbf{a}, \mathbf{B}, \mathbf{d}) \in \mathcal{F}, \lambda_i \geq -\rho \tag{16c}$$

6 Iterative LP Algorithm with Row and Column Generation

A potential hurdle in the LP above is the huge number of constraints. For example, an instance with 1000 points, 10 classes, and 100 points per class would require 900,000 constraints for each (i,k) pair, while an instance with 10,000 points, 10 classes, and 1000 points per class would require 9,000,000 constraints for each (i,k) pair. Meanwhile, the fundamental theorem of linear programming asserts that if an optimal solution exists, there exists an optimal basic feasible solution that will involve n active constraints, where $n = N + (p+1)p/2$ and $m = N^2$ ($\mathcal{O}(N^2)$). Note that $n << m$. There is likely to be a large amount of degeneracy.

Our goal is to significantly reduce the number of constraints that must be explicitly included in the LP. Let $\mathcal{N}_A \subseteq \mathcal{N}$ be the set of active points considered in the LP. Let $\mathcal{S}_i \subseteq \mathcal{C}_i$ and $\bar{\mathcal{S}}_i \subseteq \bar{\mathcal{C}}_i$ for each point $i \in \mathcal{N}$. Here, $\mathcal{S}_i$ ($\bar{\mathcal{S}}_i$) represents the set of co-class (non-class) neighbors that are currently considered in the LP.

Let W be a positive scalar. Consider the following reduced LP that maximizes both ϵ and the minimum distance variables t_i:

$$\max_{\mathbf{a},\mathbf{B},\mathbf{dt}} \quad \sum_{i \in \mathcal{N}_A} t_i \tag{17a}$$

$$\text{s.t.} \quad t_i + \epsilon \leq d_{ik} \qquad \forall i \in \mathcal{N}_A, k \in \bar{\mathcal{S}}_i \tag{17b}$$

$$t_i \leq d_{ij} \qquad \forall i \in \mathcal{N}_A, j \in \mathcal{S}_i \tag{17c}$$

$$t_i \geq 0 \qquad \forall i \in \mathcal{N}_A \tag{17d}$$

$$d_{ij} = \sum_{k=1}^{p} \delta_{ijk} a_k + \sum_{k=1}^{p} \delta_{ijk}^2 b_{kk} + 2 \sum_{k=1}^{p-1} \sum_{\ell=k+1}^{p} \delta_{ijk} \delta_{ij\ell} b_{k\ell} \qquad \forall i,j \in \mathcal{N} \tag{17e}$$

$$0 \leq d_{ij} \leq 1 \qquad \forall i,j \in \mathcal{N} \tag{17f}$$

$$\mathbf{a} \in \mathbb{R}^p \tag{17g}$$

$$b_{k\ell} \in \mathbb{R} \qquad \forall k = 1,\ldots,p, \ell = k,\ldots,p \tag{17h}$$

Algorithm 2 Iterative LP via row and column generation

Initialize: $\mathcal{N}_A = \mathcal{N}$; $\mathcal{O} = \emptyset$; $\mathcal{S}_i = ?$; $\bar{\mathcal{S}}_i = ?$; $\mathcal{P} = ?$; Fix ϵ;
`DONE = False`
while `not DONE` **do**
  `DONE = True`
  (Re-)Solve LP (17) for $\mathbf{B}^*, d_{ij}^*, t_i^*$. Fix the distance metric to d_{ij}^*.
  Step 1: Create set of alleged outliers. Let $\mathcal{O}^{\text{alleged}} = \{i \in \mathcal{N}_A : t_i^* < \min_{j \in \mathcal{S}_i} d_{ij}^*\}$.
  Step 2: Check for violated constraints.
  for $i \in \mathcal{N}_A$ (Note: Can be done in parallel) **do**
    Set $\mathcal{V}_i = \{j \in \mathcal{C}_i \backslash \mathcal{S}_i : t_i^* > d_{ij}^*\}$.
    Set $\bar{\mathcal{V}}_i = \{k \in \bar{\mathcal{C}}_i \backslash \bar{\mathcal{S}}_i : t_i^* + \epsilon > d_{ik}^*\}$.
    Set $\mathcal{D}_i = \{j \in \mathcal{C}_i \backslash \mathcal{S}_i : d_{ij}^* > 1\}$.
    if $|\mathcal{V}_i| > 0$, **then** $\hat{j} = \arg\max\{t_i^* - d_{ij}^* : j \in \mathcal{V}_i\}$ and set $\mathcal{S}_i = \mathcal{S}_i \cup \{\hat{j}\}$; `DONE = False`.
    else $(|\mathcal{V}_i| = 0)$, **if** $i \in \mathcal{O}^{\text{alleged}}$, **then** i is a true outlier: $\mathcal{N}_A = \mathcal{N}_A - \{i\}$ and $\mathcal{O} = \mathcal{O} \cup \{i\}$.
    if $|\bar{\mathcal{V}}_i| > 0$, **then** $\hat{k} = \arg\max\{t_i^* + \epsilon - d_{ik}^* : k \in \bar{\mathcal{V}}_i\}$ and set $\bar{\mathcal{S}}_i = \bar{\mathcal{S}}_i \cup \{\hat{k}\}$; `DONE = False`.
    if $|\bar{\mathcal{V}}_i| > 0$, **then** $\hat{j} = \arg\max\{d_{ij}^* : j \in \bar{\mathcal{V}}_i\}$ and set $\bar{\mathcal{S}}_i = \bar{\mathcal{S}}_i \cup \{\hat{j}\}$; `DONE = False`.
  end for
  Step 3: Attempt to re-insert outliers.
  Step 4: Remove outliers: $\mathcal{N}_A = \mathcal{N}_A - \{O\}$
end while

7 Outlier re-insertion

We assume that the distance metric d_{ij} is fixed for the remainder of this section. Let $\mathcal{N}$ be the set of all points, $\mathcal{C}_i$ the set of co-class neighbors of point i, $\bar{\mathcal{C}}_i$ the set of non co-class neighbors of point i, and $\mathcal{O}$ the current set of outliers. Let $\mathcal{IK} = \{(i,k) \in \mathcal{N} \times \mathcal{N} : \nexists\, j \in \mathcal{C}_i \backslash \mathcal{O} : d_{ij} + \epsilon \leq d_{ik}, k \in \bar{\mathcal{C}}_i\}$. Set $M_{ik} = 1 - d_{ik}$. (We are making use of our normalization $d_{ij} \in [0,1]$ here.) For any outlier penalty parameter $\rho > 0$ (I suggest $\rho = 1$), the following mixed-integer linear program attempts to minimize the number of outliers given a fixed distance metric.

$$\min_{\mathbf{y},\mathbf{z}} \quad \rho \sum_{i \in \mathcal{O}} z_i \tag{18a}$$

$$\text{s.t.} \quad \sum_{j \in \mathcal{C}_i} d_{ij} y_{ij} + \epsilon \leq d_{ik} + M_{ik} z_k \qquad \forall (i,k) \in \mathcal{IK} \tag{18b}$$

$$\sum_{j \in \mathcal{C}_i} y_{ij} = 1 \qquad \forall i \in \mathcal{N} \backslash \mathcal{O}, j \in \mathcal{C}_i \tag{18c}$$

$$\sum_{j \in \mathcal{C}_i} y_{ij} = 1 - z_i \qquad \forall i \in \mathcal{O}, j \in \mathcal{C}_i \tag{18d}$$

$$y_{ij} \leq 1 - z_j \qquad \forall i \in \mathcal{N}, j \in \mathcal{C}_i \tag{18e}$$

$$y_{ij} \in \{0,1\} \qquad \forall i \in \mathcal{N}, j \in \mathcal{C}_i \tag{18f}$$

$$z_i \in \{0,1\} \qquad \forall i \in \mathcal{O} \tag{18g}$$

$$z_i = 0 \qquad \forall i \in \mathcal{N} \backslash \mathcal{O} \tag{18h}$$

Several observations:

1. Bolun wrote the RHS of constraint (18b) as $M_i(z_i + z_k)$. This is correct, but the formulation above is tighter, meaning the linear programming relaxation provides a better lower bound, and thus optimizers believe it will be better. Why can we replace $M_i(z_i + z_k)$ with $M_{ik}z_k$? Now that we have constraints (18d), which effectively set the LHS of (18b) to ϵ if point i is deemed an outlier ($z_i = 1$), there is no reason to include $M_{ik}z_i$ on the RHS of (18b) as well.

2. What is the set $\mathcal{IK}$ doing? What is its purpose? Given that the distance metric is fixed, for any pair of points $(i, k) : i \in \mathcal{N}, k \in \bar{\mathcal{C}}_i$, if

$$\min\{d_{ij} : j \in \mathcal{C}_i \backslash \mathcal{O}\} + \epsilon \leq d_{ik}$$

which holds if and only if

$$\exists j \in \mathcal{C}_i \backslash \mathcal{O} : d_{ij} + \epsilon \leq d_{ik} \ , \tag{19}$$

then the corresponding constraint (18b) is already satisfied (and will continue to be satisfied once outliers are considered) and is therefore redundant.

3. Is it really worth the trouble to create the set $\mathcal{IK}$? Won't the solver take care of this? While solvers are good at eliminating redundant constraints, it is best to help them eliminate constraints a priori especially when we can immediately identify them. This saves the solver time when building the optimization model and in preprocessing. Example: Suppose there are 10 classes, 100 points per class, for a total of 1000 points. For simplicity, suppose there are 3 outliers per class (that is, our algorithm has thus far identified 3 outliers per class so that $|\mathcal{O}| = 30$). There are 900,000 constraints that must be included if constraints (18b) are written "$\forall i \in \mathcal{N}, k \in \bar{\mathcal{C}}_i$" since, for every point i (of which there are 1,000), there are 900 non-neighbors. However, if we were to replace "$\forall i \in \mathcal{N}, k \in \bar{\mathcal{C}}_i$" with $\forall (i, k) \in \mathcal{IK}$, then at least 846,810 (94%) of the constraints of type (18b) can be eliminated before doing a single optimization. Why? In this example, there are 970 non-outliers, each of which has 9×97 non-neighbors that are not outliers. These 846,810 pairs of points have already been shown to satisfy constraints (18b) to arrive at the current distance metric. We write "at least" because there are likely many other pairs of points involving outliers (i.e., $k \in \mathcal{O}$) for which the condition (19) also holds.

7.1 AIMMS implementation

In AIMMS, there are several items to be aware of. First, define the set $\mathcal{IK}$ as follows:

$$\mathcal{IK} = \{(i, k) | k \in \bar{\mathcal{C}}_i \ \texttt{and not exists}(j \in \mathcal{C}_i \backslash \mathcal{O} | d_{ij} + \epsilon \leq d_{ik})\}$$

Second, z_i should only be defined for $i \in \mathcal{O}$. You should *not* define it for all $i \in \mathcal{N}$ and then set $z_i = 0$ for all $i \in \mathcal{N} \backslash \mathcal{O}$ as in (18h). Note that AIMMS understands that if z_i appears in a constraint written for all $i \in \mathcal{N}$, it will only include it when z_i exists.

Third, in summary, the following formulation should be used in AIMMS:

$$\min_{\mathbf{y},\mathbf{z}} \quad \rho \sum_{i \in \mathcal{O}} z_i \tag{20a}$$

$$\text{s.t.} \quad \sum_{j \in \mathcal{C}_i} d_{ij} y_{ij} + \epsilon \leq d_{ik} + M_{ik} z_k \qquad \forall (i,k) \in \mathcal{IK} \tag{20b}$$

$$\sum_{j \in \mathcal{C}_i} y_{ij} = 1 - z_i \qquad \forall i \in \mathcal{N}, j \in \mathcal{C}_i \tag{20c}$$

$$y_{ij} \leq 1 - z_j \qquad \forall i \in \mathcal{N}, j \in \mathcal{C}_i \tag{20d}$$

$$y_{ij} \in \{0,1\} \qquad \forall i \in \mathcal{N}, j \in \mathcal{C}_i \tag{20e}$$

$$z_i \in \{0,1\} \qquad \forall i \in \mathcal{O} \tag{20f}$$

8 Iterative MILP approach with row and column generation: Maximize the minimum separation with outliers

Model (1) can be expressed as a huge mixed-integer linear programming problem:

$$\max_{\boldsymbol{\lambda},\mathbf{B},\mathbf{d},\mathbf{w},\mathbf{y},\mathbf{z}} \quad \lambda - \rho \sum_{i\in\mathcal{N}} z_i \tag{21a}$$

$$\text{s.t.} \quad \sum_{j\in\mathcal{S}_i} w_{ij} + \lambda \le d_{ik} + M_{ik} z_i \qquad \forall i \in \mathcal{N}, k \in \bar{\mathcal{S}}_i \tag{21b}$$

$$w_{ij} \le d_{ij} \qquad \forall i \in \mathcal{N}, j \in \mathcal{S}_i \tag{21c}$$

$$w_{ij} \le y_{ij} \qquad \forall i \in \mathcal{N}, j \in \mathcal{S}_i \tag{21d}$$

$$w_{ij} \ge d_{ij} + y_{ij} - 1 \qquad \forall i \in \mathcal{N}, j \in \mathcal{S}_i \tag{21e}$$

$$\sum_{j\in\mathcal{S}_i} y_{ij} = 1 \qquad \forall i \in \mathcal{N} \tag{21f}$$

$$d_{ij} = \sum_{k=1}^{p} \delta_{ijk}^2 b_{kk} + 2 \sum_{k=1}^{p-1} \sum_{\ell=k+1}^{p} \delta_{ijk}\delta_{ij\ell} b_{k\ell} \qquad \forall i,j \in \mathcal{P} \tag{21g}$$

$$b_{k\ell} \in \mathbb{R} \qquad \forall k = 1,\dots,p, \ell = k,\dots,p \tag{21h}$$

$$d_{ij} \ge 0 \qquad \forall i,j \in \mathcal{P} \tag{21i}$$

$$w_{ij} \ge 0 \qquad \forall i \in \mathcal{N}, j \in \mathcal{S}_i \tag{21j}$$

$$y_{ij} \in \{0,1\} \qquad \forall i \in \mathcal{N}, j \in \mathcal{S}_i \tag{21k}$$

$$z_i \in \{0,1\} \qquad \forall i \in \mathcal{N} \tag{21l}$$

$$\lambda \ge 0 \tag{21m}$$

9 Target neighbors

A fundamental distinction that makes our approach more general than the distance metric learning approaches of ... is that ours does not rely on auxiliary input information in the form of so-called target neighbors. The target neighbors of $\mathbf{v}_i$ are the co-class points that the user desires to be closest to $\mathbf{v}_i$. Target neighbor-based methods fix a priori a set of points and attempt to learn a linear transformation of the input space such that the resulting nearest neighbors of $\mathbf{v}_i$ are indeed its target neighbors. Unfortunately, in many applications target neighbors or triples are not available. In the absence of prior target neighbor knowledge, Weinberger and Saul cite suggest using the K nearest neighbors with the same class label, as determined by Euclidean distance. Figure ?? illustrates the poor distance metrics that may arise when target neighbors are not available at the outset and Euclidean distance is used instead. It is worth emphasizing that Weinberger and Saul cite and Ying and Li () rely on Euclidean distance in their computational experiments.

Another possibility is that target neighbors and/or triples are available with an initial data set, but when additional data becomes available, the process of updating the target neighbors and/or triples is time-consuming and prone to errors. Finally,

Consequently, we believe that there is a

Specifically, the distance metric optimization frameworks of ... all require the a priori identification of target neighbors for each input $\mathbf{v}_i$ at the outset of learning. The target neighbors of $\mathbf{v}_i$ are those that we desire to be closest to $\mathbf{v}_i$; in particular, we attempt to learn a linear transformation of the input space such that the resulting nearest neighbors of xi are indeed its target neighbors. We emphasize that target neighbors are fixed a priori and do not change during the learning process. This step significantly simplifies the learning process by specifying a priori which similarly labeled inputs to cluster together. In many applications, there may be prior knowledge or auxiliary information (e.g., a similarity graph) that naturally identifies target neighbors. In the absence of prior knowledge, the simplest prescription is to compute the k nearest neighbors with the same class label, as determined by Euclidean distance

Weinberger and Saul's LMNN formulation: Let $\mathcal{T}_i$ be the target co-class neighbors of point i; $\mathcal{T} = \{(i,j) : i \in \mathcal{N}, j \in \mathcal{T}_i\}$ be the set of target pairs; $\mathcal{U} = \{(i,j,k) : i \in \mathcal{N}, j \in \mathcal{T}_i, k \in \bar{\mathcal{C}}_i\}$ be the triplets for which a large margin of separation is desired. Let $\mu \in [0,1]$ be a scalar weight.

$$\min_{\mathbf{B},\mathbf{s}} \quad (1-\mu) \sum_{(i,j)\in\mathcal{T}} \boldsymbol{\delta}_{ij}^\top \mathbf{B} \boldsymbol{\delta}_{ij} + \mu \sum_{(i,j,k)\in\mathcal{U}} s_{ijk} \tag{22a}$$

$$\text{s.t.} \quad \boldsymbol{\delta}_{ij}^\top \mathbf{B} \boldsymbol{\delta}_{ij} + 1 \leq \boldsymbol{\delta}_{ik}^\top \mathbf{B} \boldsymbol{\delta}_{ik} + s_{ijk} \qquad \forall (i,j,k) \in \mathcal{U} \tag{22b}$$

$$s_{ijk} \geq 0 \qquad \forall (i,j,k) \in \mathcal{U} \tag{22c}$$

$$\mathbf{B} \succeq \mathbf{0} \tag{22d}$$

10 Nomenclature

Decision variable	Definition
d_{ij}	distance between points i and j
w_{ij}	$d_{ij}y_{ij}$
y_{ij}	1 if point j is the closest neighbor to i; 0 otherwise
z_i	1 if point i is an outlier; 0 otherwise
λ_i	separation (difference) between closest non-neighbor and closest neighbor
λ_i	separation (difference) between closest non-neighbor and closest neighbor
$\mathbf{B}$	$\frac{p(p-1)}{2}$

Proposition 2 *Note that we may always assume that the data (input vectors) v_i have been normalized to reside in the unit hypercube as follows: $v_{ij} \leftarrow (v_{ij} - m)/(M - m)$ where $m = \min\{v_{ij} : i \in \mathcal{N}, j = 1, \ldots, D\}$ and $M = \max\{v_{ij} : i \in \mathcal{N}, j = 1, \ldots, D\}$.*