Journal of Mathematical Biology (2019) 79:1–29
https://doi.org/10.1007/s00285-019-01348-1

Mathematical Biology

# Low-dimensional representation of genomic sequences

Richard C. Tillquist[1] · Manuel E. Lladser[2]

Received: 30 April 2018 / Revised: 12 November 2018 / Published online: 30 March 2019
© Springer-Verlag GmbH Germany, part of Springer Nature 2019

## Abstract

Numerous data analysis and data mining techniques require that data be embedded in a Euclidean space. When faced with symbolic datasets, particularly biological sequence data produced by high-throughput sequencing assays, conventional embedding approaches like binary and $k$-mer count vectors may be too high dimensional or coarse-grained to learn from the data effectively. Other representation techniques such as Multidimensional Scaling (MDS) and Node2Vec may be inadequate for large datasets as they require recomputing the full embedding from scratch when faced with new, unclassified data. To overcome these issues we amend the graph-theoretic notion of "metric dimension" to that of "multilateration." Much like trilateration can be used to represent points in the Euclidean plane by their distances to three non-colinear points, multilateration allows us to represent any node in a graph by its distances to a subset of nodes. Unfortunately, the problem of determining a minimal subset and hence the lowest dimensional embedding is NP-complete for general graphs. However, by specializing to Hamming graphs, which are particularly well suited to representing biological sequences, we can readily generate low-dimensional embeddings to map sequences of arbitrary length to a real space. As proof-of-concept, we use MDS, Node2Vec, and multilateration-based embeddings to classify DNA 20-mers centered at intron–exon boundaries. Although these different techniques perform comparably, MDS and Node2Vec potentially suffer from scalability issues with increasing sequence length whereas multilateration provides an efficient means of mapping long genomic sequences.

**Keywords** Feature extraction · Graph embeddings · Hamming graph · Metric dimension · Reads · Resolving set

**Electronic supplementary material** The online version of this article (https://doi.org/10.1007/s00285-019-01348-1) contains supplementary material, which is available to authorized users.

✉ Manuel E. Lladser
manuel.lladser@colorado.edu

1 Department of Computer Science, The University of Colorado, Boulder, CO 80309-0526, USA

2 Department of Applied Mathematics, The University of Colorado, Boulder, CO 80309-0526, USA

## 1 Introduction

Symbolic information plays a prominent role in modern biology. In contrast, many of the most powerful, well-studied machine learning tools available, including $k$-nearest neighbors (KNN) (Fix and Hodges Jr 1951) and support vector machines (SVM) (Cortes and Vapnik 1995), assume that feature vectors reside in $\mathbb{R}^d$, for some integer $d \geq 1$. It is not always clear, however, how to embed symbolic data into a real-vector space to leverage the power of these algorithms. Ideally, such embeddings should accurately reflect relevant relationships between different examples and should be low-dimensional to allow for fast classification and to avoid overfitting. When dealing with large datasets, it is essential that the process of generating an embedding be fast, preferably linear, with respect to input size. Furthermore, new, unclassified examples should be readily embeddable.

Many strategies for representing genomic sequences rely on features derived from knowledge of various biochemical properties of nucleotides or amino acids, including thermodynamic stability, structural flexibility (Sciabola et al. 2012), hydrophobicity, charge, and polarity among others (Sarda et al. 2005; Cai et al. 2003a; Bock and Gough 2001). The most common methods which rely solely on the symbols comprising a sequence are $k$-mer count vectors (Leslie et al. 2002), which record the occurrences of all $k$-mers in a sequence, and binary vectors (Cai et al. 2003b), which indicate the presence or absence of each nucleotide or amino acid at each position within a sequence. These embeddings can be, however, unnecessarily high-dimensional.

Other more sophisticated embedding techniques include Fisher kernels (Jaakkola et al. 1999) and neural networks in the BioVec tool (Asgari and Mofrad 2015). Fisher kernel methods apply the results of a trained generative model, typically a hidden Markov model (Baum and Petrie 1966), to the creation of features for use in a discriminative model, such as an SVM. Instead, BioVec embeds sequences by considering the contexts in which they appear, an approach based on Word2Vec (Mikolov et al. 2013; Mikolov et al. 2013). Techniques related to BioVec have been used to generate embeddings of DNA sequences of varying length (Ng 2017) and to take advantage of known protein properties, like membrane localization and thermostability, in producing amino acid sequence embeddings (Yang et al. 2018). All these methods require a substantial amount of data nevertheless to learn appropriate, useful embeddings for all examples.

Multidimensional Scaling (MDS) (Krzanowski 2000), also known as principal coordinates analysis (PCoA, not to be confused with PCA), assumes a metric to embed $n$ examples into $\mathbb{R}^{n-1}$. This technique has seen multiple applications for comparing and visualizing biological communities in metagenomic studies (Lozupone and Knight 2005; Lozupone et al. 2011). To the best of our knowledge, MDS has not been used for generating features of biological sequence data. This may be for good reasons. Indeed, unless the mapped examples happen to be near a low-dimensional subspace, classifiers based on MDS will be computationally demanding. Moreover, MDS does not handle new points efficiently: given a new set of examples to be included in the embedding, MDS must be rerun from scratch.

More recent algorithms such as DeepWalk (Perozzi et al. 2014) and Node2-Vec (Grover and Leskovec 2016), which were originally devised to embed general

graphs into real space, may also be used to generate features for sequence data. Though such techniques have not been applied to this task before, interpreting biological sequences as existing in a discrete abstract metric space allows us to depict such sequences as graphs. These algorithms use random walks to generate node "contexts" and take advantage of the skip-gram neural network architecture used in Word2Vec to represent vertices in a graph as real vectors. The resulting embeddings have been shown to be quite useful for graph visualization, node clustering, and node classification tasks with very large networks (Perozzi et al. 2014). Despite the scalability of these techniques for relatively large graphs, they too must be rerun from scratch when facing new nodes in a graph, with prohibitive time or memory requirements for a full embedding.

The chief goal of this manuscript is to develop a methodology for embedding symbolic datasets and specifically biological sequence data into Euclidean space, making systematic and efficient use of machine learning algorithms to learn classifiers from such datasets. The basis of our method is *multilateration*, an amendment of the notion of *metric dimension* in graph theory (Slater 1975; Harary and Melter 1976). Broadly speaking, we seek to uniquely represent elements of a set in a way analogous to how trilateration can be used to represent points in the plane by their distances to three non-colinear points. In abstract metric spaces, however, more than three points may be needed to represent each element in the space uniquely. Moreover, determining the smallest number of points required to accomplish this task, or even approximating this number with guarantees, in a large finite metric space is a computationally challenging problem in general.

We argue that multilateration-based embeddings have various desirable features for classification. Indeed, they tend to be comparatively low-dimensional and preserve the identity of examples (i.e., are one-to-one). They are also able to handle new instances effortlessly. Furthermore, despite rarely being an isometry, embeddings based on multilateration map similar examples to closeby points in Euclidean space (i.e., are in some sense continuous).

**Organization of the paper** A fair amount is known concerning the metric dimension of certain families of graphs (Chartrand et al. 2000). In this work, we are most concerned with three main results. First, finding the metric dimension of a general graph is an NP-complete problem (Cook 1971). This has been established via reduction from the 3-SAT (Khuller et al. 1996) and 3-dimensional matching (Gary and Johnson 1979) problems. The similarities between metric dimension and multilateration strongly suggest that this is also the case for multilateration (Sect. 2.1). Second, metric dimension is closely related to a number of other NP-complete problems including $k$-embeddability (Blumenthal 1953) and the test set problem (Berman et al. 2005). It is from these relationships that we are able to derive further intuition for multilateration along with an approximation algorithm (Sect. 2.2).

Third, we are able to generalize a bound on the metric dimension of Cartesian products of connected graphs and $K_2$, the complete graph on two vertices (Chartrand et al. 2000), to arbitrary Hamming graphs (Hamming 1950) (Sect. 3). The proof is constructive and yields an algorithm quadratic in the length of underlying strings for generating resolving, though not always optimal, sets which grow logarithmically in

the number of nodes. In contrast, the so-called Information Content Heuristic (ICH) algorithm (Hauptmann et al. 2012) has exponential complexity. As illustrations, we consider the problems of embedding all octapeptides (Sect. 4) as well as DNA codons (Sect. 5) into a real space.

Finally, we use features derived via multilateration to classify intron–exon boundaries in the genome of *Drosophila melanogaster* (Sect. 6). Features based on MDS and Node2Vec among other embeddings are applied and compared on the same task.

## 2 Multilateration

We begin by introducing the concept of multilateration in an abstract but general setting.

**Definition 1** (*Multilateration problem*) Given $I$, a finite set of items, and $F$, a set of functions defined over $I$, determine a minimal set $R \subseteq F$ such that the vectors $(r(i))$, with $r \in R$, uniquely identify each item $i \in I$. Equivalently, if $\mathbf{M}$ is a matrix with rows indexed by $I$ and columns indexed by $F$ such that $\mathbf{M}(i, f) = f(i)$, determine a minimal subset of columns needed to uniquely identify each row.

At a high level, multilateration may be thought of as an amendment to or an extension of the graph theoretic notion of metric dimension (Slater 1975; Harary and Melter 1976). Given a graph $G = (V, E)$, a set $R \subseteq V$ is called *resolving* if for all $u, v \in V$, with $u \neq v$, there is an $r \in R$ such that $d(r, v) \neq d(r, u)$. Here $d(i, j)$ is the length of a shortest path between two nodes $i$ and $j$ in the graph. The *metric dimension* of $G$, denoted as $\beta(G)$, is then defined as the size of a minimal resolving set of $G$.

In essence, metric dimension asks about the least number of nodes in a graph required to uniquely identify all points in the graph based on the distances to said nodes. When applied to the distance matrix of a graph, multilateration answers the same question as metric dimension. However, while metric dimension assumes an underlying graph structure, multilateration does not. In particular, multilateration does not require entries in the matrix of interest to abide by the triangle inequality or, indeed, to be numeric at all. Borrowing terminology, we refer to a smallest set $R$ solving the multilateration problem as an *optimal resolving* or *minimal multilateration* set.

### 2.1 NP-completeness

Here we show that multilateration is NP-complete via reduction from the *set cover* problem (Karp 1972). This problem can be described as follows. Given $U$, a finite set, and $S = \{S_1, \ldots, S_m\}$, a finite cover of $U$, namely a collection of subsets of $U$ such that $\cup_{i=1}^{m} S_i = U$, determine a smallest sub-collection of $S$ whose union is still $U$.

**Theorem 1** *The general multilateration problem is NP-complete.*

***Proof*** Consider a finite set $U$ and a finite cover $S = \{S_1, \ldots, S_m\}$ of $U$. Let $a$ be any element outside $U$; for instance, one could take $a = \{U\}$. Furthermore, consider the matrix $\mathbf{M}$ with rows indexed by $U \cup \{a\}$, columns indexed by the indices in the set $\{1, \ldots, m\}$, and with entries defined as follows:

$$\mathbf{M}(i, j) := \begin{cases} i, & \text{if } i \in S_j; \\ a, & \text{otherwise.} \end{cases}$$

Note that $\mathbf{M}(a, j) = a$ for $j \in \{1, \ldots, m\}$ because $a \notin U$.

We claim the following (see Appendix 8.1 for its proof). □

**Lemma 1** *Let $J \subset \{1, \ldots, m\}$. The columns with indices in $J$ resolve the rows of $\mathbf{M}$ if and only if $U = \cup_{j \in J} S_j$.*

Let $n = |U| \cdot |J|$. To complete the proof of the theorem, we must verify that the multilateration problem is in NP. Namely, we must show that there is a non-deterministic algorithm capable of solving the problem in polynomial time. To do this we need only show that the validity of a solution can be checked in polynomial time. The brute force approach suffices in this context. Extract the sub-matrix $\mathbf{M}'$ of $\mathbf{M}$ with rows indexed by $U$ and columns indexed by $J$. This takes $O(n)$ steps. Next, compare each pair of rows in $\mathbf{M}'$. If any two of its rows are equal, then $J$ is not a valid solution to the instance of the multilateration problem. Otherwise it is a valid solution. Since there are a total of $\binom{|U|}{2}$ comparisons to make, the total time to check a solution is $O(n) + \binom{|U|}{2} \cdot |J| = O(|U|^2|J|) = O(n^2)$. Furthermore, the construction of the matrix $\mathbf{M}$ takes polynomial time in $|U| \cdot |S| = O(n)$. The set cover is therefore polynomial time reducible to multilateration. As a result, Lemma 1 implies that the multilateration problem is NP-complete.

### 2.2 Approximation algorithm

Many approximation and heuristic algorithms designed for estimating solutions to the metric dimension problem of a graph $G = (V, E)$ may be applied directly to multilateration. Three of the most popular approaches for approximating $\beta(G)$ utilize genetic algorithms (Kratica et al. 2009), a variable neighborhood search (Mladenović et al. 2012), and a greedy selection criterion (Hauptmann et al. 2012). The latter is based on the so-called Information Content Heuristic (ICH), which was first used for generating approximate solutions to instances of the test set problem (Berman et al. 2005). The ICH algorithm has the significant advantage of guaranteeing an approximation ratio of no more than $1 + (1 + o(1)) \cdot \ln |V|$, the best possible ratio for metric dimension (Hauptmann et al. 2012).

Consider a matrix $\mathbf{M}$ with $n$ rows and $m$ columns. Assume that altogether its columns are resolving; in particular, the multilateration problem has at least one solution. Next, we describe how the ICH algorithm finds a resolving set (though not necessarily optimal) of $\mathbf{M}$. Notice that an equivalence relation on rows is defined by any subset of columns $R$: say that two rows $u$ and $v$ are in the same equivalence class (with respect to $R$) when $\mathbf{M}(u, s) = \mathbf{M}(v, s)$, for every $s \in R$. The ICH algorithm interprets the equivalence classes defined by a particular $R$ as outcomes of a (discrete) probability distribution and seeks to find an $R$ of maximal entropy $H(R)$. Since the uniform distribution uniquely achieves the maximum entropy over $n$ items, a set $R$ of maximal entropy must be resolving.

The ICH algorithm is greedy. At each iteration, the entropies of all possible probability distributions generated by adding single unused columns to a growing set of columns are computed. A single column associated with the maximum calculated entropy is then added to the ever increasing set. More formally, let $R_0 = \emptyset$ and define the set $R_i$ of columns of $\mathbf{M}$ recursively as follows: $R_i = R_{i-1} \cup \{C^*\}$, where $C^*$ is any column $C$ of $\mathbf{M}$ which satisfies:

$$\begin{aligned} C^* &= \arg\max_C \ H(R_{i-1} \cup \{C\}), \\ &= \arg\max_C \ H(R_{i-1} \cup \{C\}) - H(R_{i-1}). \end{aligned}$$

This process repeats until $R_i$ imposes a uniform distribution over the rows of $M$ i.e. $H(R_i) = \log(n)$.

The computational complexity of this algorithm is $O(nm^2)$ as the $i$th iteration must consider $(m - i + 1)$ columns of length $n$, with a maximum of $m$ iterations. This is reasonably effective for up to medium-sized matrices. For larger matrices, however, the ICH algorithm may be prohibitively slow. In the context of biological sequence analysis where $n$ and $m$ often exceed $10^3$ or $10^6$, a linear or sublinear algorithm would be vastly preferable. This motivates the study of multilateration on specific families of matrices in the hopes that more efficient approximation methods can be constructed.

## 3 Specialization to Hamming graphs

Due to the prevalence of $k$-mers (i.e. genomic sequences of length $k$) in biological contexts, we address the multilateration problem for $k$-mers with respect to the Hamming distance (Hamming 1950).

Let $A$ be a finite set of certain size $a > 0$. We call this set the reference alphabet. In what follows, $H_{k,A}$ denotes the Hamming graph of $k$-mers formed using characters in $A$. The vertex set of $H_{k,A}$ is by definition $A^k$; in particular, it has $a^k$ nodes. Furthermore, two vertices $u = (u_1, \ldots, u_k)$ and $v = (v_1, \ldots, v_k)$ are adjacent (i.e. joined by an edge) if and only if there exists a unique $1 \leq i \leq k$ such that $u_i \neq v_i$. Since the distance between two vertices of $H_{k,A}$ coincides with their Hamming distance, solving the multilateration problem for the distance matrix of $H_{k,A}$ is equivalent to determining its metric dimension as a graph.

Since the graph structure of a Hamming graph depends only on the size of its reference alphabet, we write $H_{k,a}$ as shorthand for the Hamming graph with reference alphabet $\{0, \ldots, a-1\}$ (see Fig. 1).

Before continuing we revise a notion of product between graphs. The *(Cartersian) product* between $G_1 = (V_1, E_1)$ and $G_2 = (V_2, E_2)$ is the graph denoted $G_1 \,\square\, G_2$ with vertex set $V_1 \times V_2$. Two vertices $(x_1, x_2), (y_1, y_2) \in V_1 \times V_2$ are said adjacent if and only if $x_1$ and $y_1$ are adjacent in $G_1$ and $x_2 = y_2$, or $x_1 = y_1$ and $x_2$ and $y_2$ are adjacent in $G_2$.

It follows that

$$H_{k,a} = H_{1,a} \,\square\, H_{k-1,a}, \tag{1}$$

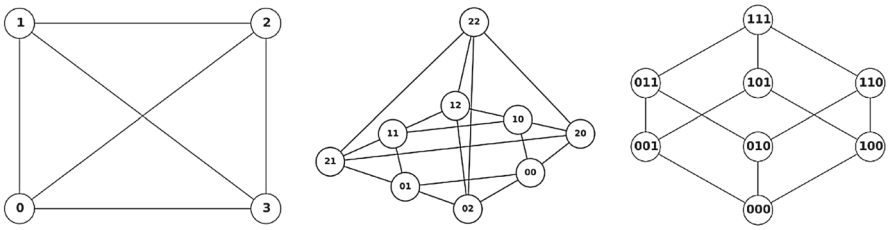


**Fig. 1** From left to right, visual representation of vertices and edges in the Hamming graphs $H_{1,4}$, $H_{2,3}$, and $H_{3,2}$

where $H_{1,a}$ is isomorphic to $K_a$ (i.e. the complete graph with vertex set $\{0, \ldots, a-1\}$). This recursive structure for Hamming graphs suggests that the metric dimension of $H_{k,a}$ is related to that of $H_{k-1,a}$. In fact, it is known that $\beta(G) \leq \beta(G \,\square\, K_2) \leq \beta(G)+1$, for any connected graph $G$ (Chartrand et al. 2000). This in turn implies that $\beta(H_{k-1,2}) \leq \beta(H_{k,2}) \leq \beta(H_{k-1,2})+1$. Our next result generalizes these inequalities for an arbitrary alphabet size.

**Theorem 2** $\beta(H_{k-1,a}) \leq \beta(H_{k,a}) \leq \beta(H_{k-1,a}) + \lfloor a/2 \rfloor$

According to the theorem:

$$\beta(H_{k,a}) \leq (a-1) + (k-1)\lfloor a/2 \rfloor,$$

i.e., the metric dimension of $H_{k,a}$ grows at most linearly with $k$; in particular, we can embed all $a^k$ $k$-mers into an $O(k)$-dimensional Euclidean space using multilateration. This is crucial for the practical use of multilateration. Moreover, because the proof of Theorem 2 is constructive, from any given resolving set $R_{k-1}$ of $H_{k-1,a}$, we can construct a resolving set for $H_{k,a}$ in $O(|R_{k-1}|)$ time (see Algorithm 3). In particular, since $H_{1,a} = K_a$ and any subset of $(a-1)$ nodes in $K_a$ is resolving, repeated applications of Algorithm 3 allow us to construct resolving sets for $H_{k,a}$ in polynomial $O(ak^2)$ time. In contrast, the time complexity of the ICH algorithm would be $O(a^{3k})$ i.e. exponential in $k$.

We note that tighter bounds on the metric dimension of Hamming graphs are possible (Cáceres et al. 2007), for example, in the case where $a$ is small in comparison to $k$. However, these bounds proceed from non-constructive probabilistic arguments (Chvátal 1983). Also, while the metric dimension of Hamming graphs has been studied in some depth, it remains unknown whether the problem is NP-complete or not for this family of graphs.

*Proof of Theorem 2* Let $\mathbf{d}$ denote the distance matrix of $H_{k-1,a}$ when vertices are sorted lexicographically. Furthermore, let $\mathbf{1}$ be the matrix of dimension $(a^{k-1} \times a^{k-1})$ with all entries equal to one. For each integer $0 \leq i \leq (a-1)$, let $V_i$ denote the set of all $k$-mers that start with the character $i$. Sorting the vertices of $H_{k,a}$ also lexicographically, the distance matrix $\mathbf{D}$ of $H_{k,a}$ has the following block structure:

$$\mathbf{D} = \begin{array}{c} \\ V_0 \\ V_1 \\ \vdots \\ V_{a-1} \end{array} \begin{array}{c} \begin{array}{cccc} V_0 & V_1 & \cdots & V_{a-1} \end{array} \\ \begin{pmatrix} \mathbf{d} & \mathbf{d+1} & \cdots & \mathbf{d+1} \\ \mathbf{d+1} & \mathbf{d} & \cdots & \mathbf{d+1} \\ \vdots & \vdots & \ddots & \vdots \\ \mathbf{d+1} & \mathbf{d+1} & \cdots & \mathbf{d} \end{pmatrix} \end{array}, \tag{2}$$

where the rows and columns are indexed by the sets $V_0, \ldots, V_{a-1}$.

To begin, we show that $\beta(H_{k-1,a}) \leq \beta(H_{k,a})$, i.e. $\beta(H_{k,a})$ is an increasing function of $k$ for fixed $a$. For this let $\mathbf{S}_0$ be the sub-matrix of $\mathbf{D}$ associated with rows labeled by $V_0$. Namely:

$$\mathbf{S_0} = \begin{array}{c} \\ V_0 \end{array} \begin{array}{c} \begin{array}{cccc} V_0 & V_1 & \cdots & V_{a-1} \end{array} \\ \begin{pmatrix} \mathbf{d} & \mathbf{d+1} & \cdots & \mathbf{d+1} \end{pmatrix} \end{array}. \tag{3}$$

In addition, for each $x \in H_{k,a}$, let $\mathbf{S}_0(\cdot, x)$ denote the column of $\mathbf{S}_0$ labeled by $x$. Consider the following equivalence relation over $H_{k,a}$. Say that $x$ is equivalent to $y$ if and only if the vector $\mathbf{S}_0(\cdot, x) - \mathbf{S}_0(\cdot, y)$ has identical entries. The following intermediate result is useful (see Appendix 8.2).

**Lemma 2** *(a) Each $x \in H_{k,a}$ is equivalent to a unique element in $V_0$. (b) If $R$ is a resolving set of $\mathbf{S}_0$ and $x, y \in H_{k,a}$ are equivalent then $(R \setminus \{x\}) \cup \{y\}$ also resolves $\mathbf{S}_0$.*

Suppose that $R_k$ is a minimal resolving set of $H_{k,a}$. Then $R_k$ must also resolve the rows of $\mathbf{S}_0$. Further, due to Lemma 2(a), for each $x \in R_k$ there is a unique $y(x) \in V_0$ such that $x$ and $y(x)$ are equivalent. Define $R'_k := \cup_{x \in R_k} \{y(x)\}$; in particular, $|R'_k| \leq |R_k|$. Due to Lemma 2(b), $R'_k$ is also a resolving set of $\mathbf{S}_0$. Furthermore, because $R'_k \subset V_0$, $R'_k$ resolves the rows of $\mathbf{d}$. As a result $\beta(H_{k-1,a}) \leq |R'_k| \leq |R_k| = \beta(H_{k,a})$, as claimed.

Next, we show that $\beta(H_{k,a}) \leq \beta(H_{k-1,a}) + \lfloor a/2 \rfloor$. For each integer $0 \leq i \leq (a-1)$, let $\mathbf{S}_i$ denote the sub-matrix of $\mathbf{D}$ associated with rows labeled by $V_i$. Consider the following finer equivalence relation over $H_{k,a}$. Say that $x$ and $y$ are equivalent if and only if, for each $0 \leq i \leq (a-1)$, the vector $\mathbf{S}_i(\cdot, x) - \mathbf{S}_i(\cdot, y)$ has identical entries. Before continuing, we state two intermediate results that are used to construct a resolving set for $H_{k,a}$ from a resolving set for $H_{k-1,a}$ (see Appendices 8.3 and 8.4).

**Lemma 3** *(a) If $R_{k-1}$ is a resolving set for $H_{k-1,a}$ then, for each $0 \leq i \leq (a-1)$, $\{0\} \times R_{k-1}$ is a resolving set for $\mathbf{S}_i$. (b) If for each $0 \leq i \leq (a-1)$, $R_k$ is a resolving set for $\mathbf{S}_i$, and $x, y \in H_{k,a}$ are equivalent, then $(R_k \setminus \{x\}) \cup \{y\}$ is also a resolving set for each $\mathbf{S}_i$.*

**Lemma 4** *If $0 \leq i < j \leq (a-1)$ and $u, v, x \in H_{k-1,a}$ then (a) $\mathbf{D}((i,u),(i,x)) \neq \mathbf{D}((j,v),(i,x))$ or $\mathbf{D}((i,u),(j,x)) \neq \mathbf{D}((j,v),(j,x))$, and (b) if there is $0 \leq m \leq (a-1)$ different from $i$ and $j$ then $\mathbf{D}((i,u),(i,x)) \neq \mathbf{D}((j,v),(i,x))$ or $\mathbf{D}((i,u),(m,x)) \neq \mathbf{D}((j,v),(m,x))$.*

Let $R_{k-1} = \{x_1, \ldots\}$ be a minimal resolving set of $H_{k-1,a}$. Furthermore, let $R'_k$ and $R''_k$ be as given in Algorithm 3. It follows that

$$R'_k = \{(i, x_{\lfloor i/2 \rfloor + 1}) \text{ with } 0 \leq i \leq (a-1)\};$$
$$R''_k = \{(0, x_j) \text{ with } \lfloor (a-1)/2 \rfloor + 1 < j \leq |R_{k-1}|\}.$$

We note that the set $R'_k$ is well-defined because

$$\lfloor (a-1)/2 \rfloor + 1 \leq (a-1) = \beta(H_{1,a}) \leq \beta(H_{k,a}) = |R_{k-1}|.$$

We claim that $R_k = (R'_k \cup R''_k)$ resolves $H_{k,a}$. To verify this, it suffices to show that (1) for each $i$, $R_k$ resolves $\mathbf{S}_i$; and (2) for each $i \neq j$, $R_k$ differentiates rows in $\mathbf{S}_i$ from rows in $\mathbf{S}_j$.

Due to Lemma 3(a), for each $i$, $\{0\} \times R_{k-1}$ is a resolving set of $\mathbf{S}_i$. But note that for each $0 \leq j \leq (a-1)$, $(0, x_{\lfloor j/2 \rfloor + 1})$ is equivalent to $(j, x_{\lfloor j/2 \rfloor + 1})$. Thus, by repeated applications of Lemma 3(b), it follows that $(R'_k \cup R''_k)$ is a resolving set of $\mathbf{S}_i$. This shows the first claim. To show the second claim, assume without loss of generality that $0 \leq i < j \leq (a-1)$, and consider $(i, u) \in V_i$ and $(j, v) \in V_j$. Define $x := x_{\lfloor i/2 \rfloor + 1}$. We consider two cases. If $\lfloor i/2 \rfloor = \lfloor j/2 \rfloor$ then $(i, x), (j, x) \in R_k$. Hence, due to Lemma 4(a), $R_k$ differentiates row $(i, u)$ in $\mathbf{S}_i$ from row $(j, v)$ in $\mathbf{S}_j$. Otherwise, if $\lfloor i/2 \rfloor \neq \lfloor j/2 \rfloor$ then we must have $a \geq 3$. As a result, there is $m \neq i$ such that $\lfloor m/2 \rfloor = \lfloor i/2 \rfloor$; in particular, $m \neq j$ and $(i, x), (m, x) \in R_k$. Lemma 4(b) implies that $R_k$ also differentiates row $(i, u)$ in $\mathbf{S}_i$ from row $(j, v)$ in $\mathbf{S}_j$. This shows the second claim, and completes the proof of the Theorem. □

**Algorithm 3**
**Hamming Graph Resolving Set Construction**
Input: $R_{k-1}$ is a resolving set of $H_{k-1,a}$
Output: $R_k$ is a resolving set of $H_{k,a}$

```
1: function CONSTRUCTRESOLVINGSET(R_{k-1},a)
2:     R'_k ← {}
3:     R''_k ← {}
4:     i ← 0
5:     for w ∈ R_{k-1} do
6:         //Include pairs of columns to guarantee that Lemmas 4(a) and 4(b) apply
7:         //This differentiates rows in S_i and S_j with i ≠ j
8:         if i < a then
9:             R'_k ← R'_k ∪ {iw}
10:            if i < (a − 1) then
11:                R'_k ← R'_k ∪ {(i + 1)w}
12:            end if
13:        end if
14:        //Ensure that every element of R_{k-1} is represented
15:        //Thus, by Lemmas 3(a) and 3(b), every S_i is resolved
16:        if i ≥ a then
17:            R''_k ← R''_k ∪ {0w}
```

```
        end if
        i ← (i + 2)
    end for
    R_k ← (R'_k ∪ R''_k)
    return R_k
end function
```

We note that even though Algorithm 3 is deterministic and based on the proof of Theorem 2, it is easy to tweak in various ways to produce a number of different resolving sets for $H_{k,a}$ from a single resolving set for $H_{k-1,a}$.

### 3.1 Automorphisms of Hamming graphs

Given a resolving set, we may also generate other resolving sets by applying any automorphism (i.e., bijection from $\{0,\ldots,a-1\}^k$ to itself that preserves edges) in $H_{k,a}$.

We denote the automorphism group of $H_{k,a}$ as $\mathbb{A}(H_{k,a})$.

Because automorphisms of a graph preserve distances between vertices, if $R_k$ is a resolving set of $H_{k,a}$ then so is $\Gamma(R_k)$, for any $\Gamma \in \mathbb{A}(H_{k,a})$. The next result provides a handy representation of the elements in this group.

**Lemma 5** (Adapted from Chaouche and Berrachedi 2006) *If $\mathbb{S}'_k$ and $\mathbb{S}_k$ denote the group of permutations over $\{0,\ldots,k-1\}$ and $\{1,\ldots,k\}$, respectively, then $\mathbb{A}(H_{k,a}) = (\times_{i=1}^{k}\mathbb{S}'_a) \rtimes \mathbb{S}_k$.*

In the lemma, $\times$ denotes the operation of direct product between groups whereas $\rtimes$ denotes their semi-direct product. In particular, we may uniquely identify each automorphism $\Gamma$ of $H_{k,a}$ as a $(k+1)$-tuple of the form $(\sigma_1,\ldots,\sigma_k;\ \sigma)$, where $\sigma_1,\ldots,\sigma_k \in \mathbb{S}'_a$ and $\sigma \in \mathbb{S}_k$. Broadly speaking, this means that $\Gamma$ may be thought of as a permutation of characters in $k$-mers (described by $\sigma$), followed by the application of bijective maps at each character (the $i$th character is transformed by $\sigma_i$). To fix ideas, consider the case with $k=2$ and the alphabet $\{x,y,z\}$ (i.e. $a=3$). If for example:

$$\Gamma = \left(\begin{pmatrix} x & y & z \\ y & x & z \end{pmatrix}, \begin{pmatrix} x & y & z \\ z & x & y \end{pmatrix}; \begin{pmatrix} 1 & 2 \\ 2 & 1 \end{pmatrix}\right),$$

then $\Gamma(yz) = zx$ because $\sigma(yz) = zy$, $\sigma_1(z) = z$ and $\sigma_2(y) = x$. Likewise, $\Gamma(xx) = yz$, $\Gamma(xy) = xz$, $\Gamma(xz) = zz$, $\Gamma(yx) = yx$, $\Gamma(yy) = xx$, $\Gamma(yz) = zx$, $\Gamma(zy) = xy$, and $\Gamma(zz) = zy$.

Lemma 5 implies that $|\mathbb{A}(H_{k,a})| = (a!)^k k!$. Unfortunately, despite this rapidly growing number of automorphisms when $a > 1$, it is not necessarily true that any two resolving sets of the same size are equivalent up to an isomorphism. This has been noted in (Harrison 1963) for hypercubes (i.e. when $a=2$). More generally, say that two subsets of nodes $V_1$ and $V_2$ are equivalent if and only if there is $\Gamma \in \mathbb{A}(H_{k,a})$ such that $V_1 = \Gamma(V_2)$. An upper bound for the total number of subsets is the total number of equivalence classes times the maximum size of a single equivalence class. Since each subset of nodes is equivalent to at most $(a!)^k k!$ other sets, it follows that there are

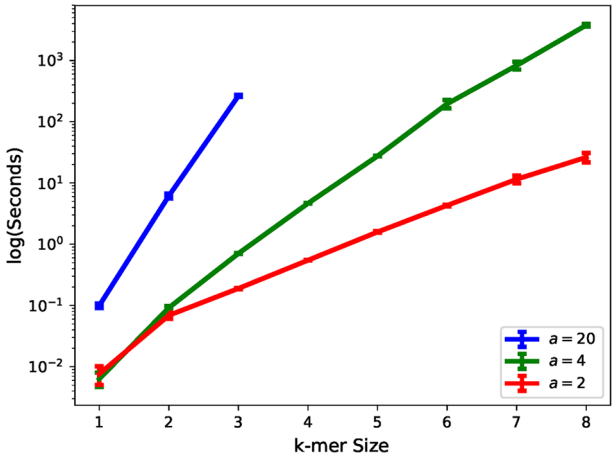


**Fig. 2** Mean time required to determine an embedding function of various Hamming graphs using Node2Vec for increasing $k$-mer sizes

at least $2^{a^k}/(k!(a!)^k)$ different equivalence classes. But note that $2^{a^k}/(k!(a!)^k) \gg a^k$ because

$$\lim_{k\to\infty} \frac{2^{a^k}}{k!(a!)^k a^k} = +\infty.$$

In particular, since the size of a subset of nodes in $H_{k,a}$ must be between 0 and $a^k$, for all $k$ sufficiently large, it must be that there are at least two subsets of the same size in different equivalence classes.

## 4 Illustrative example

Consider the problem of embedding all octapeptides (i.e., 8-mers of amino acids) into a real space. Such an embedding may be valuable for characterizing targets of the Dengue virus protease (Aguirre et al. 2012; Yu et al. 2012), a task which has proven difficult (Li et al. 2005).

Since there are about twenty-five billion octapeptides, MDS cannot be used to produce an embedding as this would require storing in memory and manipulating a non-sparse distance matrix of dimensions $20^8 \times 20^8$ (i.e., with more than six hundred quintillion entries). Similarly, and although algorithms like DeepWalk and Node2Vec only involve the incidence matrix implicitly, with a state space this large, the memory and time complexity of such algorithms to generate a complete embedding is unworkable. In fact, as seen in Fig. 2, the time required to fully embed a Hamming graph using Node2Vec grows rapidly with $k$—even for much smaller alphabet sizes. This figure displays the mean time necessary to generate an embedding function $f : \{0, \ldots, a-1\}^k \to \mathbb{R}^d$, where $d = (a-1)+(k-1)\lfloor a/2 \rfloor$ (based on Algorithm 3), using Node2Vec over ten replicates. We used the Python Node2Vec implementation

from (Grover and Leskovec 2016) and assumed edge list files had already been created. For $a = 20$, the time requirements are large enough to prevent measurements for $k > 3$ from being taken.

A possible way to overcome these memory and time bottlenecks would be to regard octapeptides as nodes in the Hamming graph $H_{8,20}$. Certainly, the size of this graph makes approximating its metric dimension via the ICH algorithm impractical. However, one can first use the ICH to find a resolving set of size 32 for $H_{3,20}$ and then apply Algorithm 3 five times to find that the following set of dimension 82 is resolving for $H_{8,20}$ (see Appendix 9):

$$R_8 = \left\{ \begin{array}{cccc} \text{AAAAAAAA}, & \text{AAAAAAAR}, & \text{AAAAAARA}, & \text{AAAAARAA}, \\ \text{AAAARAAA}, & \text{AAARAAAA}, & \text{ARWAAAAA}, & \text{CCCHHHHH}, \\ \text{CCCHHHHI}, & \text{CCCHHHIA}, & \text{CCCHHIAA}, & \text{CCCHIAAA}, \\ \text{CCCIAAAA}, & \text{CNSAAAAA}, & \text{DDDEEEEE}, & \text{DDDEEEEG}, \\ \text{DDDEEEGA}, & \text{DDDEEGAA}, & \text{DDDEGAAA}, & \text{DDDGAAAA}, \\ \text{DHFAAAAA}, & \text{EAGAAAAA}, & \text{EEEFAAAA}, & \text{EEEMFAAA}, \\ \text{EEEMMFAA}, & \text{EEEMMMFA}, & \text{EEEMMMMF}, & \text{EEEMMMMM}, \\ \text{FFFAAAAA}, & \text{GGGPPPPP}, & \text{GGGPPPPS}, & \text{GGGPPPSA}, \\ \text{GGGPPSAA}, & \text{GGGPSAAA}, & \text{GGGSAAAA}, & \text{HHHTTTTT}, \\ \text{HHHTTTTW}, & \text{HHHTTTWA}, & \text{HHHTTWAA}, & \text{HHHTWAAA}, \\ \text{HHHWAAAA}, & \text{HPVAAAAA}, & \text{IIIVAAAA}, & \text{IIIYVAAA}, \\ \text{IIIYYVAA}, & \text{IIIYYYVA}, & \text{IIIYYYYV}, & \text{IIIYYYYY}, \\ \text{KKKAAAAA}, & \text{KLQAAAAA}, & \text{LLLAAAAA}, & \text{MKYAAAAA}, \\ \text{MMMAAAAA}, & \text{NNNCCCCC}, & \text{NNNCCCCQ}, & \text{NNNCCCQA}, \\ \text{NNNCCQAA}, & \text{NNNCQAAA}, & \text{NNNQAAAA}, & \text{NSTAAAAA}, \\ \text{PPPAAAAA}, & \text{QPKAAAAA}, & \text{QQQKAAAA}, & \text{QQQLKAAA}, \\ \text{QQQLLKAA}, & \text{QQQLLLKA}, & \text{QQQLLLLK}, & \text{QQQLLLLL}, \\ \text{QYEAAAAA}, & \text{RRRDAAAA}, & \text{RRRNDAAA}, & \text{RRRNNDAA}, \\ \text{RRRNNNDA}, & \text{RRRNNNND}, & \text{RRRNNNNN}, & \text{SISAAAAA}, \\ \text{SVTAAAAA}, & \text{TTCAAAAA}, & \text{VFRAAAAA}, & \text{WMPAAAAA}, \\ \text{WWDAAAAA}, & \text{YGLAAAAA} & & \end{array} \right\}. \tag{4}$$

In other words, we can readily represent any octapeptide as an 82-dimensional real vector based on its Hamming distance to each 8-mer in the above set. For instance, using this representation:

$$\text{YAPSQYRR} \longleftrightarrow (7, 6, 6, 7, \ldots, 8, 7),$$

because the Hamming distance of the octamer YAPSQYRR to AAAAAAAA, AAAAAAAR, AAAAAARA, AAAAARAA, WWDAAAAA, and YGLAAAAA is 7, 6, 6, 7, 8, and 7, respectively.

In comparison, $k$-mer count vectors and binary vectors are unnecessarily high dimensional. For instance, using a sliding window with $k = 3$, a 3-mer count vector representation would use $20^3$ i.e. $8,000$ features. In contrast, the embedding based on metric dimension needs 98.9% fewer dimensions. With binary vectors, only 160 features are required. While vectors of this length are sparse and manageable, the embedding based on metric dimension needs at least 48.75% fewer dimensions.

## 5 Metric distortion

A critical feature of any embedding is how closely distances in the original space match those after the embedding. This motivated us to test the effect of metric dimension based embeddings on Hamming distances between nodes.

We first considered the Hamming graph $H_{3,4}$. In a biological context, this would be the graph of interest to embed codons, as done e.g. in Sect. 6. Using the ICH algorithm, we found that the set $R_3 = \{GGC, GGT, GCG, GTG, CGG, TGG\}$ is resolving. Since it can be verified that $\beta(H_{3,4}) = 6$ via an exhaustive search (see Online Resource 4), $R_3$ is also minimal.

Motivated by the use of Lipschitz constants to evaluate BioVec (Asgari and Mofrad 2015), we define

$$M(u, v) := \frac{d(u, v)}{||f(u) - f(v)||_2},$$

where $d(\cdot, \cdot)$ is the Hamming distance between 3-mers, and $f : \{A, C, G, T\}^3 \to \mathbb{R}^6$ is the embedding induced by the resolving set $R_3$. We call $M(u, v)$ the *(metric) distortion* of the embedding between $u$ and $v$.

As seen on the top-left of Fig. 3, a metric dimension based embedding produces a fairly diverse set of distortion-values centered near 1 (mean = 0.953, sd = 0.303). Also, a significant portion of the values seems to fall below one, suggesting that this embedding does a reasonable job of maintaining pairwise distances between nodes and, for many, is an expansion.

The nodes of $H_{3,4}$ were also embedded using MDS and Node2Vec (see top-right and bottom of Fig. 3). In each case, and for a reliable comparison, the dimension of the embedding was set to six to match that of the metric dimension based embedding.

The objective of metric MDS is to place a set of points in a real space of fixed dimension while minimizing "stress," i.e. the sum of squared differences between the true distance and the embedded distance of every pair of points. We used the metric MDS implementation of Python's scikit-learn library (Pedregosa et al. 2011) with the distance matrix of $H_{3,4}$ to produce representations of DNA 3-mers in $\mathbb{R}^6$. To obtain a scaled stress value, we divided it by the total number of pairwise distances. Since scikit-learn uses a stochastic iterative method for minimization, we embedded $H_{3,4}$ into $\mathbb{R}^6$ 50-times and obtained a mean scaled stress of $0.1704 \pm 8.5 \times 10^{-5}$.

Node2Vec, following the concept of Word2Vec, embeds the nodes of $H_{3,4}$ based on context. The context of a node $u \in V$ is defined as a set of sequences of nodes visited by a second order random walk starting at $u$ and guided by two parameters, $p$ and $q$. To generate contexts, we consider 30 walks of length 6 taken from each node. To guide the walks, we set $p = 1$ and $q = 1$, a parameterization making Node2Vec equivalent to DeepWalk, as well as $p = 0.0545$ and $q = 1.22$. This second pair of values was found via a simple grid search in an effort to minimize the squared difference between average distortion values of Node2Vec and multilateration embeddings. Since Node2Vec embeddings are based on random walks (and hence are themselves random), the bottom plots in Fig. 3 display the distribution of distortion-values over three replicates.

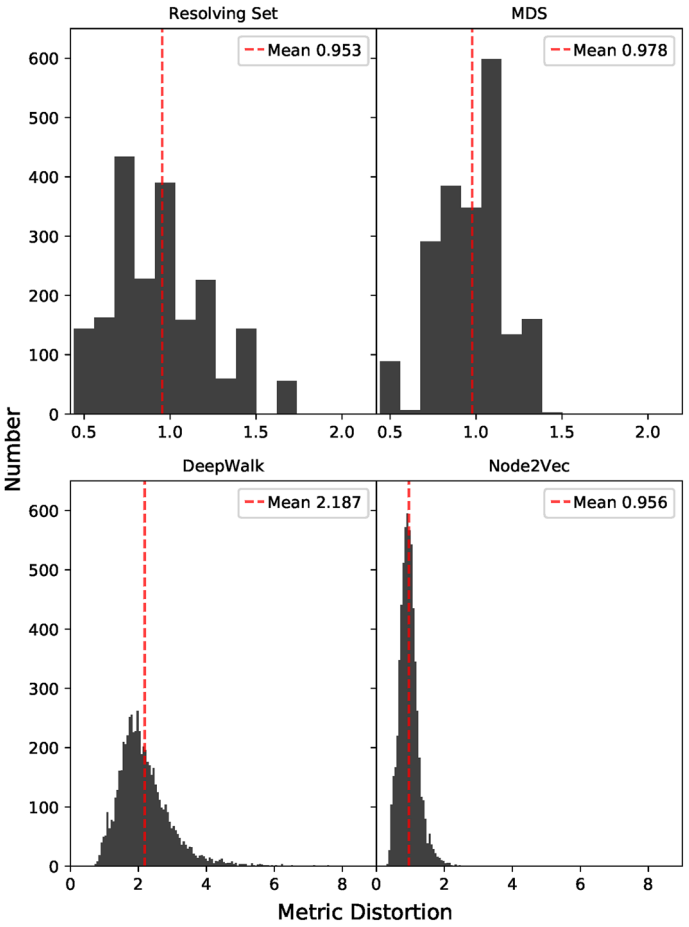


**Fig. 3** Distributions of metric distortion values in embeddings of $H_{3,4}$ based on metric dimension, MDS, and two parameterizations of Node2Vec

Metric distortion values for the MDS embedding are more tightly clustered near 1 (mean = 0.978, sd = 0.193). This is not surprising as MDS explicitly seeks to minimize the "stress" of an embedding, taking the distances between nodes into account directly. For DeepWalk, distortion-values have a mean of 2.187 (sd = 0.809), with a maximum distortion of 8.340. Such an embedding greatly distorts the distances between nodes—contracting the space overall. Instead, for Node2Vec with $p = 0.0545$ and $q = 1.22$ the mean and standard deviation are practically identical (mean = 0.956, sd = 0.283) to those found for the metric dimension based embedding. This suggests that embeddings with this parameterization also do a good job of maintaining distance relationships between nodes.

We also tested the distortions produced by metric dimension based embeddings for relatively large $k$-mers with alphabet sizes relevant for epigenetics (a = 2), genomic sequences (a = 4), and proteins (a = 20). As seen in Fig. 4, distortions in the associated

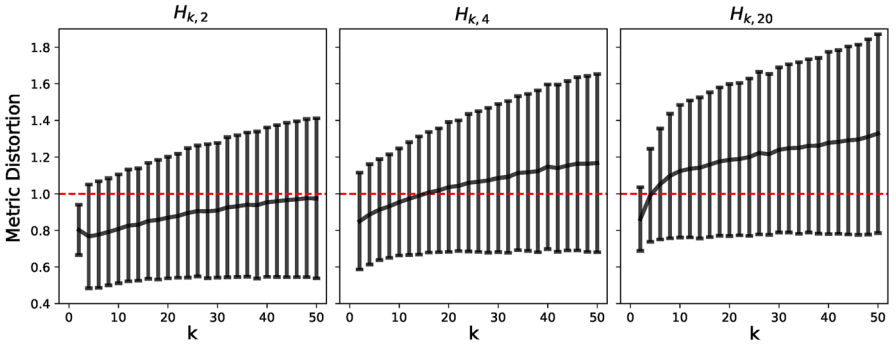


**Fig. 4** Metric distortion values for metric dimension based embeddings of the Hamming graphs $H_{k,2}$, $H_{k,4}$, and $H_{k,20}$. Error bars represent 95% confidence intervals over 10,000 random sample distances drawn from each graph

Hamming graphs have similar characteristics to those observed in $H_{3,4}$. To generate the figure, for each $a \in \{2, 4, 20\}$ and even $1 \leq k \leq 50$, a set of size $(a - 1)$ was chosen uniformly at random as the resolving set $H_{1,a}$, and then Algorithm 3 was applied iteratively to produce a resolving set for $H_{k,a}$. While many of these embeddings were expansions of the underlying space, the average distortion increased slowly with $k$ at a rate that seemed to depend on $a$. Interestingly, and even though the range of values for metric distortion increased with repeated applications of Algorithm 3, the lower-bound of the confidence intervals remained almost constant and well below one. This suggests that for a considerable number of pairs of nodes in $H_{k,a}$ the metric dimension based-embedding is nearly an isometry. In conclusion, for alphabet sizes of relevance to modern biology and for relatively large $k$-mers, metric dimension based embeddings tend to induce sensible distortions to Hamming distances despite the fact that positions under such embeddings are not chosen with the goal of maintaining pairwise Hamming distances between $k$-mers.

## 6 Proof-of-concept: learning intron–exon boundaries

Gene identification is a fundamental and well studied problem in computational biology (Hayes and Borodovsky 1998; Hoff et al. 2008; Reese et al. 2000; Stanke et al. 2004). In this section, we showcase and evaluate embeddings based on multilateration, MDS, and Node2Vec on the task of identifying intron–exon boundaries. Our goal is to provide a proof-of-concept of how these alternative feature generation methods may be applied to problems involving biological sequence data and to compare their performance to that of $k$-mer counts and binary vector representations.

We gathered DNA 20-mers centered at annotated intron–exon boundaries in the genome of *Drosophila melanogaster* from Ensembl BioMart (Yates et al. 2016). 20-mers containing ambiguous bases were thrown out. This left a set of $\sim$87K positive examples, with $\sim$22K repeated and $\sim$65K unique 20-mers in the dataset.

To apply simple binary classification and to avoid any bias towards one class or the other, an equal number of negative examples was drawn from the genome itself. In particular, this set consists of $\sim$87K 20-mers drawn randomly from the positive and negative strands of the *D. melanogaster* genome. These negative examples were allowed to match annotated intron–exon boundaries (i.e., positive examples) on prefixes or suffixes of at most 10 bases. Negative examples were generated in this way, and not simply selected uniformly at random from the set of all possible 20-mers, in order to provide a realistic space for testing.

We used two separate datasets for testing. One with duplicate positive examples included (dataset 1, see Online resource 5), and one with a single copy of each positive example (dataset 2, see Online resource 6). We then compared the performance of a KNN classifier using features generated by various embeddings on these datasets.

### 6.1 Features generation

We considered five possible feature vector representations of 20-mers (see Table 1). These included 64-dimensional 3-mer count vectors and 80-dimensional binary vectors. We note that both of these representations are sparse. Features based on multilateration, MDS, and Node2Vec used embeddings of the Hamming graph $H_{3,4}$ in real spaces instead.

To generate feature vectors based on multilateration we used the resolving set (see Sect. 5):

$$R_3 = \{GGC, GGT, GCG, GTG, CGG, TGG\}. \tag{5}$$

To embed 20-mers based on this set, we used a sliding window of length three and appended the 6-dimensional coordinates associated with each window to construct a 108-dimensional representation. The same approach was used to generate 108-dimensional feature vectors based on MDS and Node2Vec. Namely, a 6-dimensional embedding of $H_{3,4}$ was determined (see Online Resources 1 and 2) and then used to describe full 20-mers based on the coordinates of each constituent 3-mer. For Node2Vec we set $p = 0.0545$ and $q = 1.22$ (see Sect. 5).

There are many ways in which features based on these embeddings could be generated. We choose to use a sliding window of length three for two key reasons. First, we have reason to believe that codons (i.e. sequences of three consecutive nucleotides, or 3-mers) will be relevant in this application. Second, by using local information, we can generate more fine-grained embeddings which in turn may capture more subtle aspects of intron–exon boundaries.

### 6.2 Results

To qualitatively evaluate the ability of the different embedding techniques to separate positive from negative examples, we used a t-distributed stochastic neighbor embedding (t-SNE) to project data points down to two dimensions (Maaten and Hinton 2008).

**Table 1** Summary of tested feature types for 20-mers

| Name | Dimensions | Feature description |
|---|---|---|
| 3-mer counts | 64 | Vector recording number of possibly overlapping occurrences of each codon in a 20-mer |
| Binary | 80 | Vector describing the presence or absence of each nucleotide at each position in a 20-mer |
| Multilateration | 108 | Vector associated with the embedding of each codon in a 20-mer via the multilateration of $H_{3,4}$ |
| MDS | 108 | Vector associated with the six-dimensional embedding of each codon in a 20-mer using MDS |
| Node2Vec | 108 | Vector associated with the six-dimensional embedding of each codon in a 20-mer using Node2Vec |

The scikit-learn (Pedregosa et al. 2011) t-SNE implementation with default parameters and a PCA based initialization was used to generate projections of 2000 randomly selected data points represented with features based on multilateration, MDS, Node2Vec, binary vectors, and 3-mer count vectors. As seen in Fig. 5, all representations except for 3-mer count vectors exhibit clustering of positive examples away from negative examples. This strongly suggests that most of these methods of representing 20-mers based on embeddings of $H_{3,4}$ can to some extent differentiate intron–exon boundaries from non-boundaries in the *D. melanogaster* genome.

The effectiveness of these different representations in classifying intron–exon boundaries was analyzed further in numerous ways.

First, using the KNN implementaton from scikit-learn, precision-recall curves were generated for each dataset using neighborhood sizes of 3, 5, and 7 as follows. For a fixed neighborhood size $s$, and using thresholds from 1 to $s$, precision-recall values were calculated over five splits of the data using 80% of the examples for training and 20% for testing. As seen in Fig. 6, curves associated with 3-mer count vectors are substantially lower than those generated using other methods. While features based on multilateration, Node2Vec, MDS, and binary vector embeddings all achieve comparable precision-recall values, the precision-recall curve for multilateration is generally lower than others. Receiver operating characteristic (ROC) curves generated following an analogous procedure show a similar relationship between the different feature types (see Fig. 7). This is not surprising given the connection between the two representations (Davis and Goadrich 2006).

The mean and standard deviation of accuracy, sensitivity, and specificity using five nearest neighbors over 50 random subsamples of the data were also collected and analyzed. This process was repeated with different proportions (from 95 to 5%, in 5% increments) of the dataset being held out for testing. Further, to compare the performance of KNN under different feature vectors, we interpreted the average accuracy, sensitivity, and specificity over 50 replicates as the success probability of a Binomial distribution with a number of trials equal to the number of labels generated. That is, $n = (\text{number of replicates}) \times (\text{size of testing set}) = 50 \cdot |D| \cdot (1 - f)$, where $|D|$ is the total size of the dataset, and $f$ is the fraction of examples used for training. This

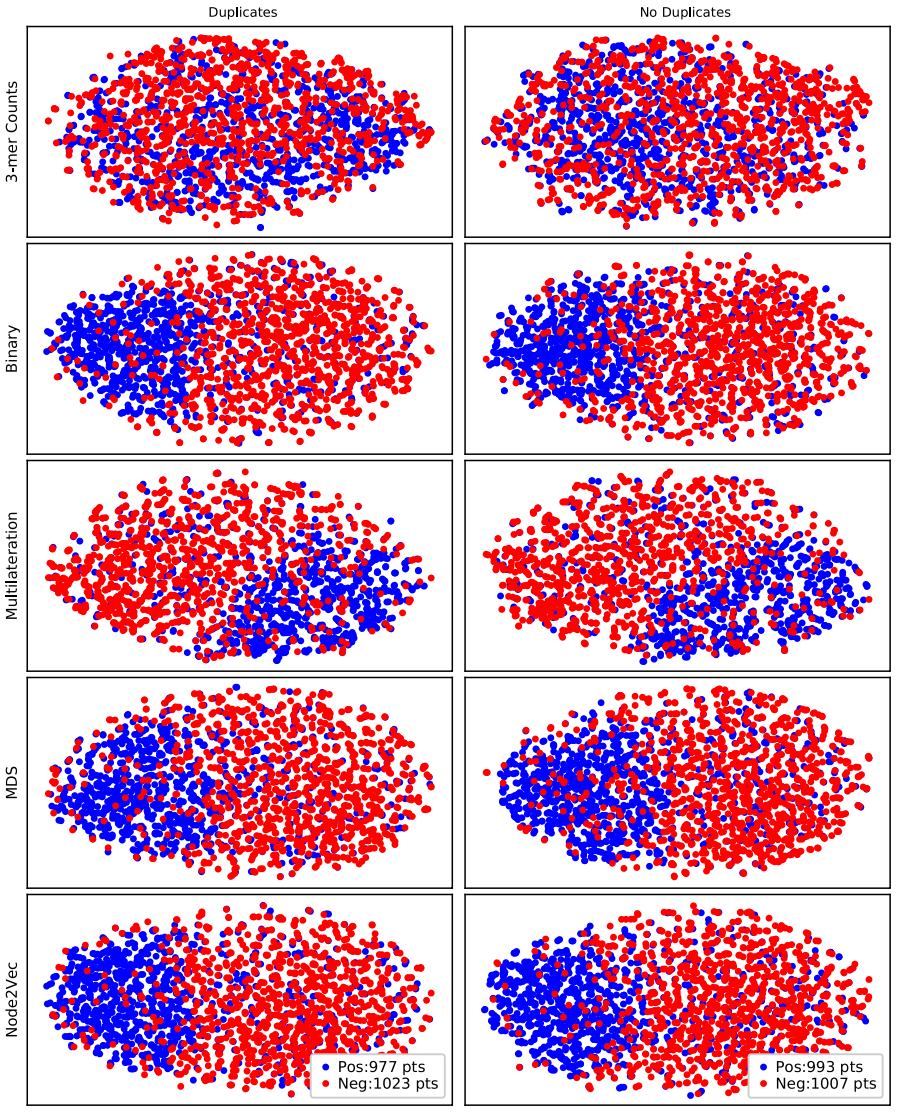


**Fig. 5** t-SNE projections of 2000 randomly chosen points from dataset 1 (left) and dataset 2 (right), using representations based on 3-mer count vectors, binary vectors, multilateration, MDS, and Node2Vec

allowed us to test the significance of differences in the accuracies, sensitivities, and specificities of each pair of features, over each dataset, and for each test set size.

As seen in Fig. 8, accuracies for 3-mer count vectors are substantially lower than those for other feature vectors. We attribute this and the poor performance of 3-mer count vectors with respect to precision-recall and ROC curves to the fact that the relative position of 3-mers within a sequence is ignored under this representation. In fact, in a context where each positive example is half intronic and half exonic DNA, location information should be relevant for classification. All other feature

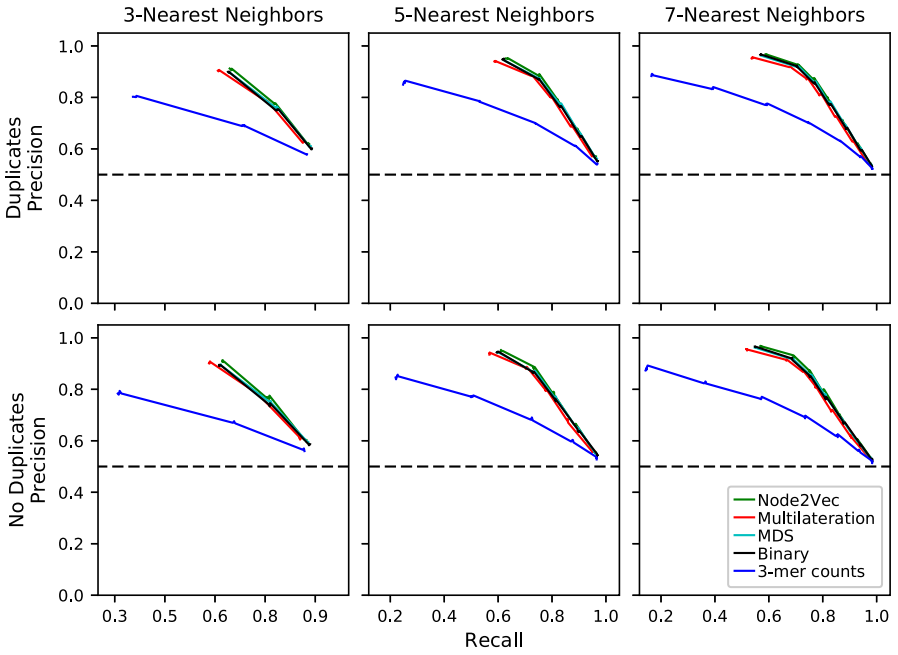


**Fig. 6** Precision-recall curves for 3-NN, 5-NN, and 7-NN classifiers under different feature types for dataset 1 (top) and dataset 2 (bottom). Five splits of the data were considered for each possible threshold

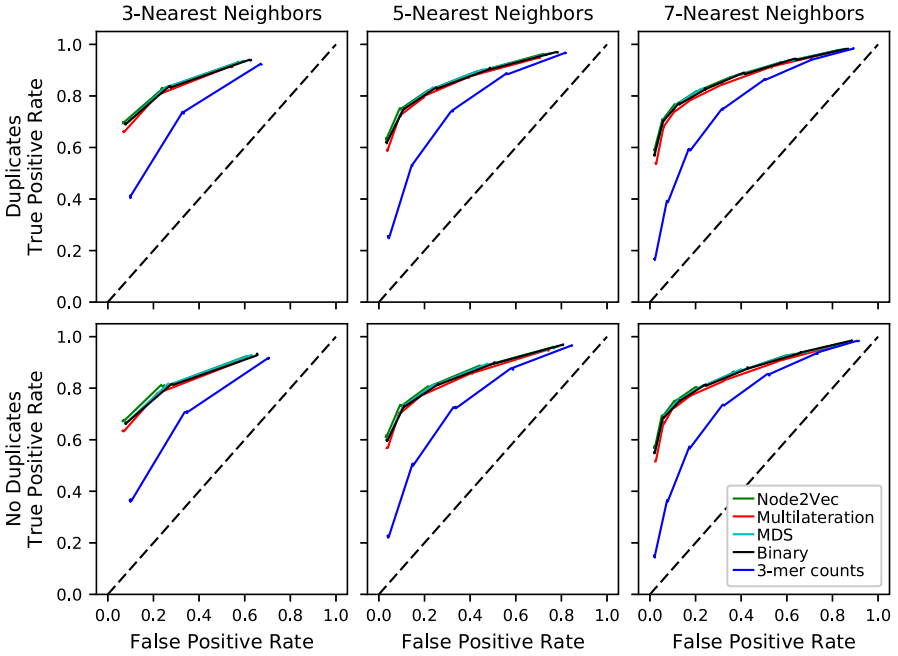


**Fig. 7** ROC curves for 3-NN, 5-NN, and 7-NN classifiers under different feature types for dataset 1 (top) and dataset 2 (bottom). Five splits of the data were considered for each possible threshold

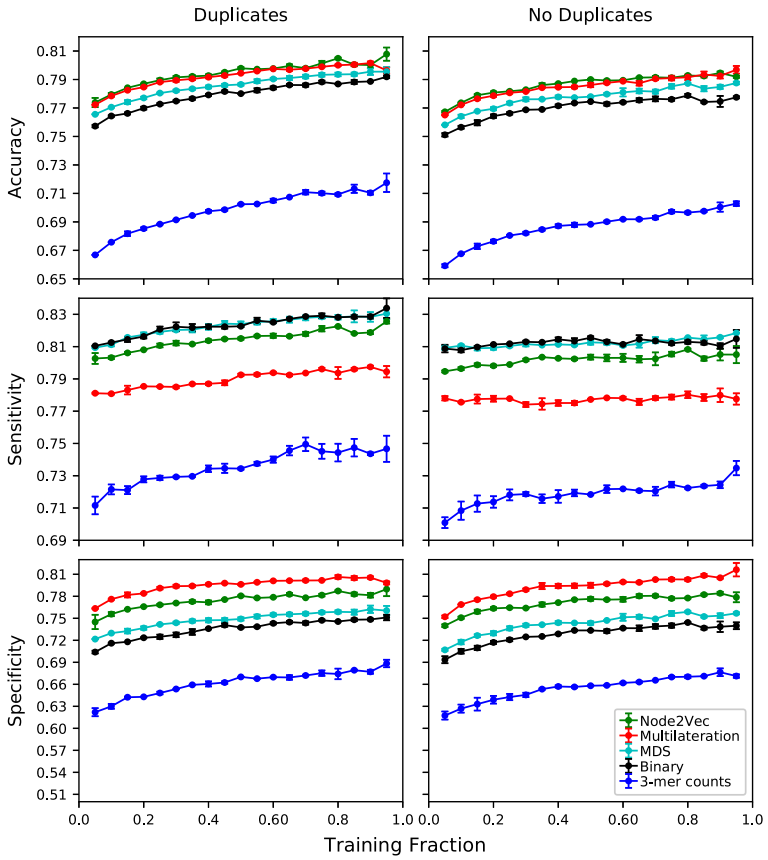


**Fig. 8** Mean and sample standard deviation of accuracy, sensitivity, and specificity for a 5-NN classifier under different feature types, and varying the fraction of data used for training. The left column displays results for dataset 1 and the right column results for dataset 2

vectors take the location of 3-mers within each example into account. Accuracies for these methods are all roughly comparable, falling within a tight 3% range (see Fig. 8). Though most differences are statistically significant based on the Binomial proportion test described before (only 5 out of 152 of the $p$-values are between 0.05 and 0.95), absolute differences in accuracy across training fraction sizes correspond to between 2000 and 4000 correctly labeled examples. This indicates features based on multilateration, MDS, Node2Vec, and a simple binary embedding are all highly competitive with one another in this context.

Also seen in Fig. 8, the sensitivity of features derived using multilateration is generally somewhat lower than that achieved using Node2Vec, MDS, and binary embeddings. In contrast, the specificity of multilateration features tends to be higher than specificities achieved with other feature types. Again, most differences across feature types in these two metrics are statistically significant. Depending on the appli-

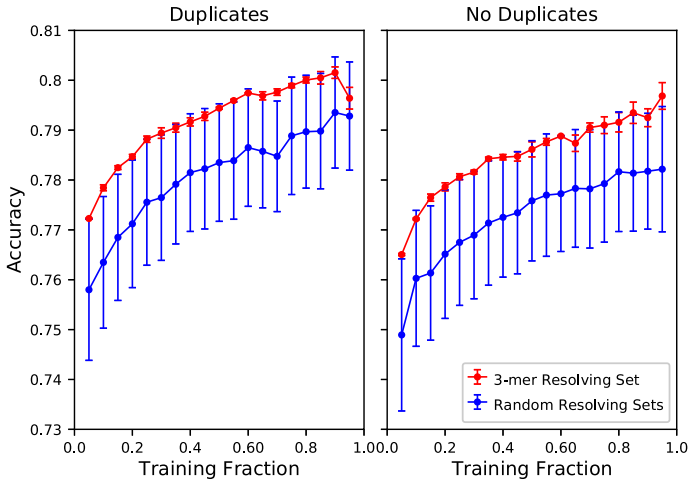


**Fig. 9** Accuracies using 5-NN with the 3-mer resolving set $R_3$, and with 50 other randomly chosen resolving sets

cation and the potential cost of different misclassifications, this may be a desirable property.

We note that there are over 18 million resolving sets of $H_{3,4}$. Due to the inherent symmetry of Hamming graph structures, many of these sets are isomorphic (see Sect. 3.1). To investigate the extent to which classification performance depends on the specific resolving set chosen, we collected a uniformly random sample of 50 resolving sets and repeated classification (see Online Resource 3). As seen in Fig. 9, $R_3$ in equation (5) ends up out-performing the average resolving set but nevertheless falls near one sample standard deviation away from the mean. In particular, the choice of resolving set does not appear to affect classification performance much in this setting. Nevertheless, this may not be the case in general and boosting over multiple classifiers, each based on a different resolving set, may be advisable to reduce performance dependence on a specific resolving set choice.

In summary, the performance of KNN classifiers for identifying intron–exon boundaries in the genome of *D. melanogaster* based on features derived from multilateration embeddings is comparable to those based on Node2Vec, MDS, and binary vector embeddings. Concerning precision-recall and ROC curves as well as sensitivity, multilateration underperforms slightly compared to these other methods. The accuracy and specificity of classifiers using multilateration based features are, however, generally higher than those obtained using other feature types. Interestingly, this proof-of-concept suggests that metric dimension serves as a guide to select the least dimension on which to embed $k$-mers into real-space using MDS and Node2Vec.

### 6.3 Biological insights via multilateration

Given the features generated by multilateration on codons, we next analyzed intron–exon boundaries to determine a subset of metric dimension based features that most

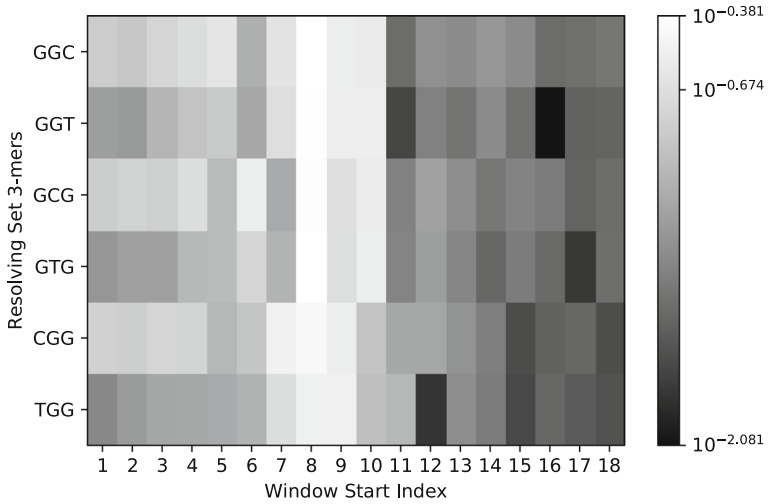


**Fig. 10** Heat-map of total variation distance between distribution of Hamming distances for positive and negative examples. The x-axis is associated with the starting index of a window of length three in a 20-mer, and the y-axis with 3-mers in the resolving set $R_3$

distinguished positive and negative examples. We focused our attention on the dataset containing no duplicate positive examples, though results are similar for the other dataset.

For each of the 108 multilateration-based features, we considered the distribution of values for positive and negative examples. Computing the total variation distance between these two distributions, we compared features with respect to how well they differentiated sequences centered at intron–exon boundaries from those that are not (see Fig. 10). We found that the most important feature in classifying intron–exon boundaries is the distance of the 3-mer in the eighth window to GGC. Moreover, while the eighth window is the most discriminatory overall, values in the seventh through tenth windows show greater discrepancies between positive and negative examples than those in the remaining windows. Notice that none of these windows appears entirely within the exon. At first, this was somewhat puzzling considering that exons are under more evolutionary pressure than introns. In particular, we might expect certain patterns in exonic DNA to be highly conserved and thus strongly indicative of an exon as opposed to an intron. However, recall that the positive examples represent all known intron–exon boundaries in the fruit fly genome; in particular, the function of proteins being coded for by sequences in the positive example set is not preserved and there is no reason to believe that any specific codon (i.e. amino acid) will be over or under represented in these sequences.

We note that while 108 features are required to guarantee unique distinguishability of all 20-mers via multilateration, only a small fraction of these are necessary to distinguish intron–exon boundary sequences reliably. Using total variation distances to order the multilateration features, we explored how many features are needed to achieve reasonable accuracy. Adding one feature at a time in the designated order, we

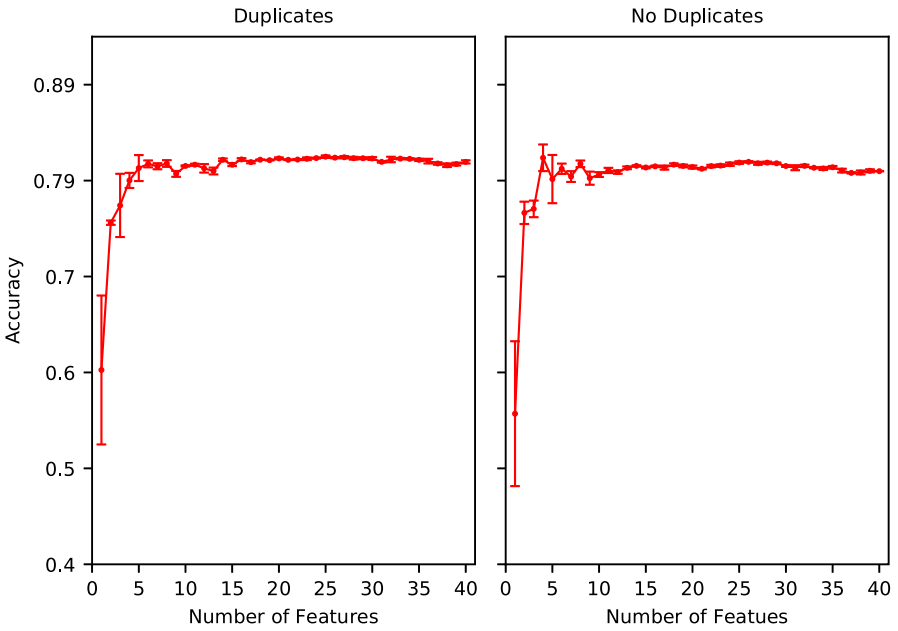


**Fig. 11** Accuracy of a KNN classifier as the top 40 metric dimension based features are added in the order determined by total variation distance. Each point is associated with 50 replicates. Plots are displayed following the same convention as Fig. 8

**Table 2** Distribution of distances of codon in the eighth window of positive examples to each 3-mer in resolving set $R_3$. The top row shows the fraction of distances expected over all possible 3-mers

| Resolving set 3-mers | 0 | 1 | 2 | 3 |
|---|---|---|---|---|
| Expected | 0.0156 | 0.1406 | 0.4219 | 0.4219 |
| GGC | 0.0047 | 0.0371 | 0.1225 | 0.8357 |
| GGT | 0.0035 | 0.0381 | 0.1322 | 0.8262 |
| GCG | 0.0046 | 0.0387 | 0.8064 | 0.1503 |
| GTG | 0.0037 | 0.0409 | 0.8306 | 0.1249 |
| CGG | 0.0038 | 0.4946 | 0.3494 | 0.1522 |
| TGG | 0.0044 | 0.2206 | 0.6387 | 0.1363 |

achieve a maximum accuracy of 0.8249 (sd = 0.0013) with the top 25 features (see Fig. 11). In fact, accuracies within 0.0071 of this maximum are reached with the top 8 features.

One of the hallmarks of intron–exon boundaries is the presence of a highly conserved AG sequence near the boundary at what is called the acceptor site (Breathnach et al. 1978). Even though our embedding does not make explicit use of this fact, features based on multilateration pick up on this characteristic as being indicative of a positive example. To see why, consider the relative frequencies in Table 2, and notice that there are far more positive examples at a distance three from GGC than we would expect by chance. This suggests that, for the second position of the eighth window, G is uncommon in positive examples. The abundance of positive examples at a distance

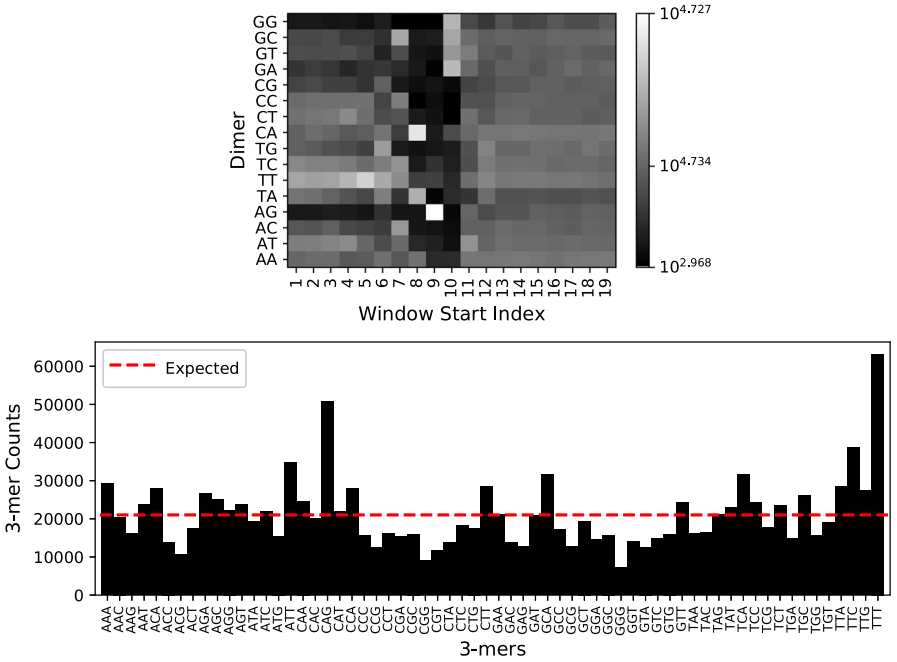


**Fig. 12** (Top) Number of DNA dimers observed in each window of all positive examples. (Bottom) 3-mer frequencies in positive examples

one from CGG then tells us that G is a rather common nucleotide in the third position. Moreover, the overabundance of positive examples at distance three from GGC and at distance two from GCG and GTG, suggest that neither G nor C nor T are common nucleotides in the second position. Consequently, the dimer AG must be a rather common nucleotide starting at position nine of the positive examples. This deduction is supported by raw counts of dimers across all positive examples (see Fig. 12 on the top). To end, we note that while 3-mer count features show an overabundance of CAG in positive examples (see Fig. 12 on the bottom), it is not immediately obvious that this 3-mer nearly always occurs immediately before the exon. Multilateration-based features provide this location information nevertheless.

## 7 Conclusion

In the Euclidean plane (i.e., two-dimensional real space) three non-colinear points suffice to represent any point by its distance to those points uniquely. In this manuscript, we have explored extensions of this idea, particularly in the context of finite metric spaces, to develop meaningful numeric features from certain types of symbolic data.

Specifically, we defined a multilateration set of a matrix as a minimal subset of columns which uniquely label all rows. As an embedding technique, multilateration focuses on producing low dimensional maps while guaranteeing unique distinguisha-

bility and ready embeddability of new examples. When applied to data that can be represented as nodes in a graph, it reduces to determining the metric dimension of the graph. This allows for the direct application of multilateration to data structured in this way. For example, multilateration may be used to embed vertices of networks like the Netflix movie ratings network (Bennett et al. 2007), arXiv citation networks (Gehrke et al. 2003), or airport networks (Opsahl 2011) in some real space. Such embeddings can then be used to encode movies, papers, and airports as features in machine learning algorithms.

While multilateration-based embeddings are agnostic of any specific domain and may be applied to any symbolic dataset, we examined how it may be used for the analysis of biological sequence data. Using features derived from multilateration, we classified DNA 20-mers in the *Drosophila melanogaster* genome as being centered or not around an intron–exon boundary. These features allowed us to achieve performances comparable to, and sometimes better than, those obtained by more traditional, or computationally intensive embedding strategies. They also made it possible to pick out the acceptor site motif as a distinguishing characteristic of positive examples.

Features based on two other embedding methods not commonly used in producing representations of biological sequences, MDS and Node2Vec, also performed well on the intron–exon boundary classification task. The success of features derived using multilateration, MDS, and Node2Vec in this proof-of-concept shows that these tools can be used to generate meaningful embeddings of biological sequence data.

Finally, the concept of multilateration is easily extended in a number of directions. Various weakenings of the unique identifiability constraint, including only requiring uniqueness between groups of rows or a set fraction of rows, could allow for more efficient discovery of a suitable set of columns. Extensions which allow for the application of multilateration on classes of matrices or graphs generated via particular underlying processes would significantly increase the scope of problems for which multilateration may prove useful.

## 8 Appendices

### 8.1 Proof of Lemma 1

Suppose first that $J$ is a resolving set for $\mathbf{M}$. Looking for a contradiction, suppose that $\cup_{j\in J} S_j \neq U$; in particular, there is $i \in U$ such that $i \notin \cup_{j\in J} S_j$. From the definition of $\mathbf{M}$, this implies that $\mathbf{M}(i, j) = a$ for all $j \in J$. But then $\mathbf{M}(i, j) = \mathbf{M}(a, j)$ for all $j \in J$, which contradicts the assumption that $J$ is a resolving set. As a result, $U = \cup_{j\in J} S_j$ as claimed.

Conversely, suppose that $U = \cup_{j\in J} S_j$. Again looking for a contradiction, suppose that $J$ is not a resolving set for $\mathbf{M}$. Then there must be at least two rows in $\mathbf{M}$ that are identical when restricted to the columns in $J$. But note that row $i$ can only contain two elements, $i$ and $a$. This means that two rows can be identical only if they contain $a$'s in the specified columns. As a result, there is $i \in U$ such that $\mathbf{M}(i, j) = a$, for all $j \in J$. But then $i \notin \cup_{j\in J} S_j$ i.e. $i \notin U$. This contradicts that $i \in U$, therefore $J$ must be a resolving set for $\mathbf{M}$. This completes the proof of the lemma.

### 8.2 Proof of Lemma 2

To show part (a), suppose without loss of generality that $x \in H_{k,a} \setminus V_0$ and, let $i \neq 0$ be such that $x \in V_i$. In particular, there is $u \in \{0, \ldots, a-1\}^{k-1}$ such that $x = (i, u)$. Define $y := (0, u)$. Since for each $z \in V_0$, $\mathbf{S}_0(z, x) - \mathbf{S}_0(z, y) = 1$, it follows that $x$ and $y$ are equivalent. Moreover, because $y \in V_0$, this shows that any element in $H_{k,a}$ is equivalent to at least one element in $V_0$. To show that said element is unique, it suffices to show that no two different elements in $V_0$ are equivalent.

To show the latter claim, consider $x = (0, u)$ and $y = (0, v)$ with $u, v \in \{0, \ldots, a-1\}^{k-1}$ and $u \neq v$; in particular, $\mathbf{D}(x, y) > 0$. But note that $\mathbf{S}_0(x, x) - \mathbf{S}_0(x, y) = -\mathbf{D}(x, y)$ and $\mathbf{S}_0(y, x) - \mathbf{S}_0(y, y) = \mathbf{D}(x, y)$. Hence $x$ and $y$ cannot be equivalent, which shows part (a).

To show part (b), consider a resolving set $R_k$ of $\mathbf{S}_0$ and $x, y \in H_{k,a}$ equivalent. Without any loss of generality assume that $x \in R_k$. Otherwise, because $R_k$ is resolving, so is $R_k \cup \{z\}$ for any $z \in H_{k,a}$. Because $x$ and $y$ are equivalent, for all $z_1, z_2 \in V_0$, $\mathbf{S}_0(z_1, x) = \mathbf{S}_0(z_2, x)$ if and only if $\mathbf{S}_0(z_1, y) = \mathbf{S}_0(z_2, y)$. In particular, replacing $x$ by $y$ cannot alter the rows of $\mathbf{S}_0$ resolved by $R_k$. Since $R_k$ is a resolving set for $\mathbf{S}_0$, this completes the proof of the lemma.

### 8.3 Proof of Lemma 3

To show part (a), let $R_{k-1}$ be a resolving set for $H_{k-1,a}$; in particular, $R_{k-1}$ resolves the rows of the matrix $\mathbf{d}$. In addition, let $\mathbf{S_{i,0}}$ be the sub-matrix of $\mathbf{S_i}$ associated with columns labeled by $V_0$. From Eq. (2), we see that $\mathbf{S}_{0,0} = \mathbf{d}$. From this, it is immediate that $\{0\} \times R_{k-1}$ resolves $\mathbf{S}_{0,0}$ and hence also $\mathbf{S}_0$. Instead, for $i > 0$, $\mathbf{S}_{i,0} = (\mathbf{d} + \mathbf{1})$ and again it is immediate that $\{0\} \times R_{k-1}$ resolves $\mathbf{S}_{i,0}$ and thus $\mathbf{S}_i$ too. This shows part (a).

To show part (b), consider a resolving set $R_k$ that resolves the rows of each $\mathbf{S}_i$. Furthermore, let $x, y \in H_{k,a}$, with $x \in R_k$, be equivalent (with the respect to the second equivalence relation defined). It follows that for all $i$ and $z_1, z_2 \in V_i$, $\mathbf{S}_i(z_1, x) = \mathbf{S}_i(z_2, x)$ if and only if $\mathbf{S}_i(z_1, y) = \mathbf{S}_i(z_2, y)$. Following an argument analogous to the one used to show Lemma 2(b), this means that replacing $x$ with $y$ cannot alter the rows of $\mathbf{S}_i$ resolved by $R_k$. Since $R_k$ resolves each $\mathbf{S}_i$, the lemma follows.

### 8.4 Proof of Lemma 4

For (a), let $0 \leq i < j \leq (a-1)$ and $u, v, x \in H_{k-1,a}$. By contradiction, suppose that $\mathbf{D}((i, u), (i, x)) = \mathbf{D}((j, v), (i, x))$, and $\mathbf{D}((j, v), (j, x)) = \mathbf{D}((i, u), (j, x))$. Equivalently, said:

$$d(u, x) = 1 + d(v, x);$$
$$d(v, x) = 1 + d(u, x).$$

The first identity implies that $d(u, x) > d(v, x)$ whereas the second one implies that $d(v, x) > d(u, x)$. Since the latter two inequalities are not simultaneously possible, part (a) follows.

To show part (b), consider $0 \leq m \leq (a-1)$ that is different from $i$ and $j$. Again proceeding by contradiction, suppose that $\mathbf{D}((i,u),(i,x)) = \mathbf{D}((j,v),(i,x))$ and $\mathbf{D}((i,u),(m,x)) = \mathbf{D}((j,v),(m,x))$. Then, because $i$, $j$, and $m$ are different, this is equivalent to saying that

$$d(u,x) = 1 + d(v,x);$$
$$d(u,x) = d(v,x).$$

Hence, we must have $d(v,x) = 1 + d(v,x)$. Since this is not possible, part (b) in the lemma follows and completes its proof.

## 9 Resolving set of $H_{8,20}$

The resolving set in Eq. (4) was found via repeated applications of Algorithm 3, starting with the set:

$$\left\{\begin{array}{lllllllll} \text{AAA,} & \text{RRR,} & \text{NNN,} & \text{DDD,} & \text{CCC,} & \text{QQQ,} & \text{EEE,} & \text{GGG,} & \text{HHH,} \\ \text{III,} & \text{LLL,} & \text{KKK,} & \text{MMM,} & \text{FFF,} & \text{PPP,} & \text{SIS,} & \text{NST,} & \text{TTC,} \\ \text{ARW,} & \text{WWD,} & \text{MKY,} & \text{QYE,} & \text{YGL,} & \text{HPV,} & \text{VFR,} & \text{EAG,} & \text{KLQ,} \\ \text{DHF,} & \text{WMP,} & \text{CNS,} & \text{SVT,} & \text{QPK} & & & & \end{array}\right\},$$

which is resolving for $H_{3,20}$. The latter was found using the ICH algorithm. For both algorithms, amino acids were ordered lexicographically by full name, not abbreviations. Namely, the following order was assumed: A < R < N < D < C < Q < E < G < H < I < L < K < M < F < P < S < T < W < Y < V.

**Acknowledgements** The authors thank the reviewers for their very insightful comments on the original version of this manuscript. This research was partially funded by the NSF IGERT Grant 1144807, and NSF IIS Grant 1836914. The authors acknowledge the BioFrontiers Computing Core at the University of Colorado–Boulder for providing High-Performance Computing resources (funded by National Institutes of Health 1S10OD012300), supported by BioFrontiers IT group.

### Compliance with ethical standards

**Conflict of interest** The authors declare that they have no conflict of interest.

## References

Aguirre S, Maestre AM, Pagni S, Patel JR, Savage T, Gutman D, Maringer K, Bernal-Rubio D, Shabman RS, Simon V, Rodriguez-Madoz JR, Mulder LC, Barber GN, Fernandez-Sesma A (2012) DENV inhibits type I IFN production in infected cells by cleaving human STING. PLoS Pathog 8(10):e1002–934

Asgari E, Mofrad MR (2015) Continuous distributed representation of biological sequences for deep proteomics and genomics. PloS One 10(11):e0141–287

Baum LE, Petrie T (1966) Statistical inference for probabilistic functions of finite state Markov chains. Ann Math Stat 37(6):1554–1563

Bennett J, Lanning S et al (2007) The Netflix prize. In: Proceedings of KDD cup and workshop, New York, vol 2007, p 35

Berman P, DasGupta B, Kao MY (2005) Tight approximability results for test set problems in bioinformatics. J Comput Syst Sci 71(2):145–162
Blumenthal LM (1953) Theory and applications of distance geometry. Clarendon Press, Oxford
Bock JR, Gough DA (2001) Predicting protein–protein interactions from primary structure. Bioinformatics 17(5):455–460
Breathnach R, Benoist C, O'hare K, Gannon F, Chambon P (1978) Ovalbumin gene: evidence for a leader sequence in mRNA and DNA sequences at the exon–intron boundaries. Proc Natl Acad Sci 75(10):4853–4857
Cáceres J, Hernando C, Mora M, Pelayo IM, Puertas ML, Seara C, Wood DR (2007) On the metric dimension of cartesian products of graphs. SIAM J Discrete Math 21(2):423–441
Cai C, Han L, Ji ZL, Chen X, Chen YZ (2003a) SVM-Prot: web-based support vector machine software for functional classification of a protein from its primary sequence. Nucleic Acids Res 31(13):3692–3697
Cai YD, Feng KY, Li YX, Chou KC (2003b) Support vector machine for predicting $\alpha$-turn types. Peptides 24(4):629–630
Chaouche FA, Berrachedi A (2006) Automorphisms group of generalized Hamming graphs. Electron Notes Discrete Math 24:9–15
Chartrand G, Eroh L, Johnson MA, Oellermann OR (2000) Resolvability in graphs and the metric dimension of a graph. Discrete Appl Math 105(1):99–113
Chvátal V (1983) Mastermind. Combinatorica 3(3–4):325–329
Cook SA (1971) The complexity of theorem-proving procedures. In: Proceedings of the third annual ACM symposium on theory of computing. ACM, pp 151–158
Cortes C, Vapnik V (1995) Support-vector networks. Mach Learn 20(3):273–297
Davis J, Goadrich M (2006) The relationship between precision-recall and roc curves. In: Proceedings of the 23rd international conference on machine learning. ACM, pp 233–240
Fix E, Hodges JL Jr (1951) Discriminatory analysis-nonparametric discrimination: consistency properties. Tech. rep, DTIC Document
Gary MR, Johnson DS (1979) Computers and intractability: a guide to the theory of NP-completeness. WH Freeman and Company, New York
Gehrke J, Ginsparg P, Kleinberg J (2003) Overview of the 2003 KDD cup. ACM SIGKDD Explor Newsl 5(2):149–151
Grover A, Leskovec J (2016) node2vec: scalable feature learning for networks. In: Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining. ACM, pp 855–864
Hamming RW (1950) Error detecting and error correcting codes. Bell Labs Techn J 29(2):147–160
Harary F, Melter R (1976) On the metric dimension of a graph. Ars Comb 2:191–195
Harrison MA (1963) The number of transitivity sets of Boolean functions. J Soc Ind Appl Math 11(3):806–828
Hauptmann M, Schmied R, Viehmann C (2012) Approximation complexity of metric dimension problem. J Discrete Algorithms 14:214–222
Hayes WS, Borodovsky M (1998) How to interpret an anonymous bacterial genome: machine learning approach to gene identification. Genome Res 8(11):1154–1171
Hoff KJ, Tech M, Lingner T, Daniel R, Morgenstern B, Meinicke P (2008) Gene prediction in metagenomic fragments: a large scale machine learning approach. BMC Bioinf 9(1):217
Jaakkola TS, Diekhans M, Haussler D (1999) Using the Fisher kernel method to detect remote protein homologies. ISMB 99:149–158
Karp RM (1972) Reducibility among combinatorial problems. In: Complexity of computer computations. Springer, pp 85–103
Khuller S, Raghavachari B, Rosenfeld A (1996) Landmarks in graphs. Discrete Appl Math 70(3):217–229
Kratica J, Kovačević-Vujčić V, Čangalović M (2009) Computing the metric dimension of graphs by genetic algorithms. Comput Optim Appl 44(2):343–361
Krzanowski WJ (2000) Principles of multivariate analysis: a user's perspective. OUP, Oxford
Leslie CS, Eskin E, Noble WS (2002) The spectrum kernel: a string kernel for SVM protein classification. Pac Symp Biocomput 7:566–575
Li J, Lim SP, Beer D, Patel V, Wen D, Tumanut C, Tully DC, Williams JA, Jiricek J, Priestle JP, Harris JL, Vasudevan SG (2005) Functional profiling of recombinant NS3 proteases from all four serotypes of dengue virus using tetrapeptide and octapeptide substrate libraries. J Biol Chem 280(31):28,766–28,774

Lozupone C, Knight R (2005) UniFrac: a new phylogenetic method for comparing microbial communities. Appl Environ Microbiol 71(12):8228–8235
Lozupone C, Lladser ME, Knights D, Stombaugh J, Knight R (2011) UniFrac: an effective distance metric for microbial community comparison. ISME J 5(2):169–172
Maaten Lvd, Hinton G (2008) Visualizing data using t-SNE. J Mach Learn Res 9((Nov)):2579–2605
Mikolov T, Chen K, Corrado G, Dean J (2013) Efficient estimation of word representations in vector space. ArXiv e-prints 1301.3781
Mikolov T, Sutskever I, Chen K, Corrado GS, Dean J (2013) Distributed representations of words and phrases and their compositionality. In: Advances in neural information processing systems, pp 3111–3119
Mladenović N, Kratica J, Kovačević-Vujčić V, Čangalović M (2012) Variable neighborhood search for metric dimension and minimal doubly resolving set problems. Eur J Oper Res 220(2):328–337
Ng P (2017) dna2vec: consistent vector representations of variable-length k-mers. ArXiv e-prints 1701.06279
Opsahl T (2011) Why Anchorage is not (that) important: binary ties and sample selection. http://toreopsahl.com/2011/08/12/why-anchorage-is-not-that-important-binary-tiesand-sample-selection. Accessed September 2013
Pedregosa F, Varoquaux G, Gramfort A, Michel V, Thirion B, Grisel O, Blondel M, Prettenhofer P, Weiss R, Dubourg V, Vanderplas J, Passos A, Cournapeau D, Brucher M, Perrot M, Duchesnay E (2011) Scikit-learn: machine learning in Python. J Mach Learn Res 12:2825–2830
Perozzi B, Al-Rfou R, Skiena S (2014) Deepwalk: online learning of social representations. In: Proceedings of the 20th ACM SIGKDD international conference on knowledge discovery and data mining. ACM, pp 701–710
Reese MG, Kulp D, Tammana H, Haussler D (2000) Genie—gene finding in Drosophila melanogaster. Genome Res 10(4):529–538
Sarda D, Chua GH, Li KB, Krishnan A (2005) pSLIP: SVM based protein subcellular localization prediction using multiple physicochemical properties. BMC Bioinform 6(1):152
Sciabola S, Cao Q, Orozco M, Faustino I, Stanton RV (2012) Improved nucleic acid descriptors for siRNA efficacy prediction. Nucleic Acids Res 41(3):1383–1394
Slater PJ (1975) Leaves of trees. Congressus Numerantium 14(549–559):37
Stanke M, Steinkamp R, Waack S, Morgenstern B (2004) AUGUSTUS: a web server for gene finding in eukaryotes. Nucleic Acids Res 32(suppl 2):W309–W312
Yang KK, Wu Z, Bedbrook CN, Arnold FH (2018) Learned protein embeddings for machine learning. Bioinformatics 34(15):2642–2648
Yates A, Akanni W, Amode MR, Barrell D, Billis K, Carvalho-Silva D, Cummins C, Clapham P, Fitzgerald S, Gil L et al (2016) Ensembl 2016. Nucleic Acids Res 44(D1):D710–D716
Yu CY, Chang TH, Liang JJ, Chiang RL, Lee YL, Liao CL, Lin YL (2012) Dengue virus targets the adaptor protein MITA to subvert host innate immunity. PLoS Pathog 8(6):e1002–780

**Publisher's Note** Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.