Opportunities and Limitations of Using Melting Curve Dissimilarity to Determine Protein Interactions

by

Joshua Teitz

A thesis submitted in partial fulfillment of the requirements for the degree of

Master of Science

Department of Computing Science

University of Alberta

Abstract

Proteins are large biomolecules that perform many functions within an organism. Proteins frequently form interactions with other biomolecules. A group of proteins held together by interactions is known as a protein complex.

Protein interactions are essential to many biological processes. Thus, determining interactions is a key objective in biology. Since experimental techniques that can directly infer interactions are currently expensive and time consuming, there is considerable interest in computational techniques that can determine protein interactions from biological data.

When a protein mixture is heated, the proportion of insoluble protein molecules forming precipitant increases for each kind of protein. A protein's melting curve records the fraction of the protein that is soluble over a series of increasing temperatures, relative to the amount of the protein that is soluble at the initial temperature. Thermal Proximity Co-Aggregation (TPCA) is the observation that interacting proteins tend to have similar melting curves. Although a previous work has provided empirical evidence of TPCA, no work has applied computational tools to melting curve datasets in an attempt to determine protein interactions. In this thesis, we apply computational tools to melting curve datasets, and describe the opportunities and limitations of such tools for determining interactions.

Clustering is the task of finding groups of related objects in a dataset. We first explore whether clusters found in melting curve datasets correspond to protein complexes. We then turn our attention to pull-down assays, a com-

monly used biochemical technique that generates a list of potential interaction partners for a protein-of-interest. To determine interactions from the results of a pull-down assay, subsequent validation experiments must be performed, each involving the protein-of-interest and a potential interaction partner. Since validation experiments are expensive and time-consuming, we explore whether performing validation experiments in order of increasing melting curve dissimilarity from the protein-of-interest can limit the number of experiments necessary to find interactions. Lastly, we describe how melting curve dissimilarity can be used to detect proteins that differentially interact in two conditions.

Acknowledgements

First and foremost, I would like to thank my supervisor Jörg Sander for his patience, support and insights as I worked on this thesis. I would also like to thank Carlos Fernandez-Patron for introducing Jörg and I to melting curves, sharing his expertise in biology, and serving on my thesis committee. A special thanks goes to Hassan Sarker for our numerous meetings and his editing of my biology-related writing. Lastly, I would like to thank Davood Rafiei and Anup Basu for serving on my thesis committee.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

Proteins are large biomolecules that perform many functions within an organism[1]. Antibodies, which bind and help neutralize foreign objects such as viruses, and enzymes, which catalyze biochemical reactions, are both examples of proteins.

Proteins frequently form interactions with other biomolecules. A group of proteins held together by various chemical interactions[2], such as hydrogen bonds, hydrophobic interactions and ionic bonds, is known as a protein complex. Figure 1.1 visualizes the DNA repair complex [15] as a graph. Each node represents a protein and each edge between a pair of proteins represents an interaction. Observe that the complex is formed from pairs of proteins that interact. Not every pair of proteins in the complex interacts, but every protein is connected to every other protein through a sequence of interactions. In this thesis, whenever we refer to an "interaction", we mean two proteins that interact with each other.

Protein interactions are essential to many biological processes. Thus, determining interactions is a key objective in biology [2]. Since experimental techniques that can directly infer interactions are expensive and time consuming, there is considerable interest in computational techniques that can determine interactions from biological data [43], [44]. In this thesis, we explore opportunities and limitations of using protein melting curve dissimilarity to determine interactions. Melting curves are introduced in Section 1.1. The

[1] `https://medlineplus.gov/genetics/understanding/howgeneswork/protein/`
[2] `https://en.wikipedia.org/wiki/Protein-protein_interaction`

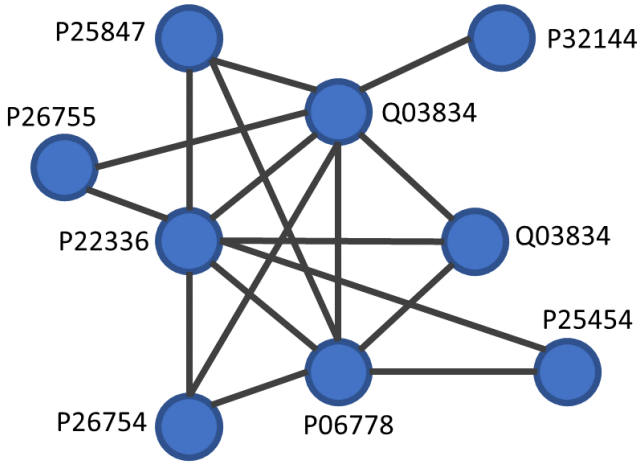

Figure 1.1: DNA repair complex graph. Each node is labelled with the UniProt ID [8] of the protein it represents.

relationship between melting curve dissimilarity and interactions is discussed in Section 1.2.

1.1 Melting Curves

Denaturation[3] is the process by which a protein's internal bonds are broken, causing it to loose its three-dimensional structure. Whereas a protein in its natural three-dimensional structure is soluble, a denatured protein is insoluble. A protein mixture consists of molecules of different kinds of proteins. Some of these molecules are soluble and others are insoluble, forming precipitant. As a protein mixture is heated, the proportion of insoluble molecules forming precipitant increases for each kind of protein, as heat causes protein molecules to denature.

A protein's melting curve records the fraction of the protein that is soluble over a series of increasing temperatures, relative to the amount of the protein that is soluble at the initial temperature. Thus, melting curves have a soluble fraction value of 1 at the initial temperature and are non-increasing. As Figure 1.2 shows, melting curves commonly have a sigmoidal shape, showing that there tends to be one temperature at which a protein's molecules begins to

[3] `https://www.britannica.com/science/denaturation`

denature and a second (higher) temperature above which all of the protein's molecules are denatured. Given a protein mixture and a series of m increasing temperatures $t_1, \ldots, t_m$, the following steps give an overview of how melting curves are generated.

1. Extract m aliquots[4] from the protein mixture, one for each temperature.

2. For each temperature t_i:

   Heat the corresponding aliquot to t_i degrees.

   Once the aliquot is at t_i degrees, use a protein detection method such as mass spectrometry to measure the amount of each protein that is soluble.

3. For each protein detected in the protein mixture:

   Divide the amounts measured at $t_1, \ldots, t_m$ by the amount measured at t_1. The resulting m soluble fraction values form the protein's melting curve.

If a protein mixture contains thousands of kinds of proteins and mass spectrometry is used as the protein detection method, then thousands of melting curves can be generated [41].

1.2 Thermal Proximity Co-Aggregation

Thermal Proximity Co-Aggregation (TPCA) is the observation of Tan et al. [41] that interacting proteins tend to have similar melting curves. The rationale for TPCA is that when a complex is heated, its proteins do not denature independently. Instead, the complex as a whole denatures, leading to the complex's proteins precipitating together. Figure 1.2 (B) shows the melting curves of two interacting proteins. Tan et al. provided evidence of TPCA by performing numerous analyses over thousands of melting curves.

[4] An aliquot is a sub-sample extracted from an original sample.

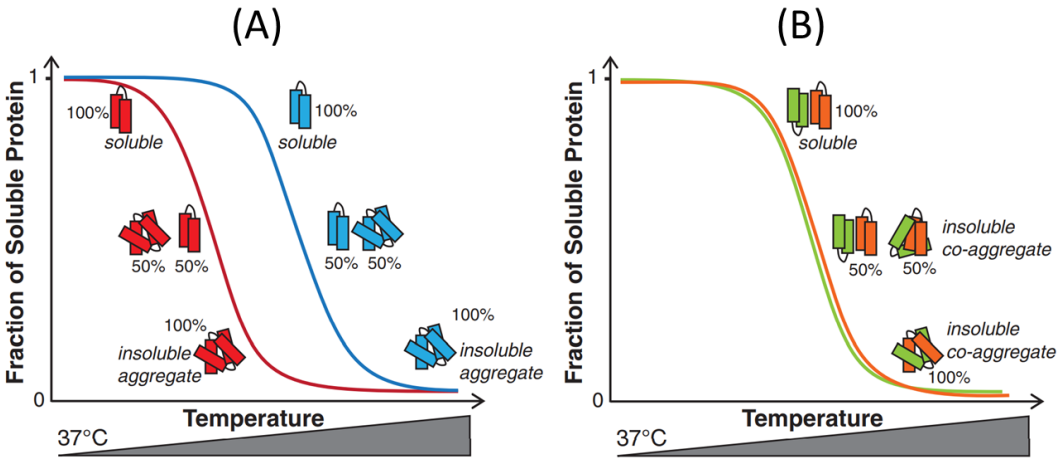

Figure 1.2: Example melting curves. (A) depicts the melting curves of two non-interacting proteins. (B) depicts the melting curves of two interacting proteins. Figure taken from [41].

Firstly, they obtained the melting curves of 7,693 human proteins from K562 lysate[5]. They assembled 111,776 protein interactions from multiple online databases and used Euclidean distance[6] to measure the melting curve dissimilarity of each interaction. Compared to randomly formed pairs of melting curves, they found that interactions tend to have statistically smaller Euclidean distances ($p <$ 2.2e-16, according to a one-tailed Mann-Whitney test[7]).

Secondly, Tan et al. considered 558 human protein complexes from the Comprehensive Resource of Mammalian Protein Complexes (CORUM) database [16]. For each complex, they computed the average Euclidean distance over all protein pairs that can be formed from the complex's proteins. The complexes varied in size from 3 to 131 proteins. For each size s, they computed an empirical distribution for average Euclidean distance according to the following procedure:

1. Randomly sample s melting curves.

2. Compute the average Euclidean distance over all pairs of the s melting

[5]K562 is a cell line derived from leukemia cells [24]. Lysis is the process by which cell membranes are destroyed. A lysate is a solution of lysed cells.

[6]Euclidean distance is defined in Section 2.2.2. The higher the Euclidean distance, the more dissimilar two melting curves are.

[7]The one-tailed Mann-Whitney test is described in Section 2.3.2

curves.

3. Repeat steps 1 and 2 10,000 times.

For each complex, they used the appropriate empirical distribution for the complex's size to compute an empirical p-value. The p-value is the fraction of values in the distribution that are smaller than the complex's average Euclidean distance. They found that 160 of the 558 complexes (28.7%) had $p < .05$ and describe such complexes as "exhibiting nonrandom TPCA signatures".

Thirdly, they considered existing algorithms that predict complexes from a protein interaction network (PIN). In a PIN, nodes correspond to proteins and an edge between two proteins means they are known to interact. The edges can be weighted or unweighted. Tan et al. built three weighted PINs over the same set of nodes and edges: a Euclidean distance PIN, where each edge is weighted as $\frac{1}{1+dist}$, with $dist$ being the Euclidean distance between the two proteins' melting curves; a publication number PIN, where each edge is weighted based on the number of publications that mention it; and an interaction reliability PIN, where each edge is weighted based on an introduction reliability score introduced in [7].

They then applied five complex prediction algorithms to the three PINs and evaluated how the PINs affected the predicted complexes. To summarize the results, for three of the five algorithms, the Euclidean distance PIN resulted in better complex predictions than the publication count PIN. For the other two algorithms, neither PIN consistently made better predictions than the other. Furthermore, for all five algorithms, neither the Euclidean distance PIN nor the interaction reliability score PIN consistently made better predictions than the other. The details of the results can be found in Figures S26 and S27 of the supplementary material of [41].

Based on the performance of the five complex prediction algorithms on the Euclidean distance PIN, Tan et al. came to the following conclusion: "TPCA profiles[8] could also serve to discover new interactions and protein complexes

[8] A TPCA profile refers to the melting curves of a group of proteins.

in combination with other approaches".

1.3 Thesis Outline

Although Tan et al. proposed TPCA and provided empirical evidence that it exists, they did not attempt to determine interactions by applying computational tools to melting curve datasets. In this thesis, we applied computational tools to melting curve datasets, and described the opportunities and limitations of such tools for determining protein interactions.

Chapter 2 provides an overview of clustering, a computational tool for finding groups of similar objects within a dataset. The chapter then investigates whether clustering melting curve datasets can provide useful information about protein complexes. Chapter 3 is motivated by pull-down assays, a commonly used biochemical technique that generates a list of potential interaction partners for a protein-of-interest. To determine interactions from the results of a pull-down assay, subsequent validation experiments must be performed, each involving the protein-of-interest and a potential interaction partner. Since validation experiments are expensive and time consuming, Chapter 3 explores whether performing validation experiments in order of increasing melting curve dissimilarity from the protein-of-interest can limit the number of experiments necessary to find interactions. Lastly, Chapter 4 describes how melting curve dissimilarity can be used to detect proteins that differentially interact.

Chapter 2

Can clustering melting curve datasets provide useful information about protein complexes?

Computational techniques that predict protein complexes are of considerable interest to scientists [43]. Although most of these techniques use an algorithm to analyze a PIN, PINs have a number of limitations. Scientists estimate that there are approximately 650,000 human protein interactions [40]. However, [41] was only able to assemble 111,776 interactions from multiple online databases. Thus a PIN is likely to contain only a small fraction of the interactions between its proteins. Furthermore, a PIN is likely to contain many false positives (i.e. edges between proteins that do not interact). In fact, [33] validated 33,000 purported interactions obtained from multiple online databases. Two thirds of these pairs were reported by a single publication relying on a single detection method. When the authors validated these low evidence pairs, they found their rate of interaction to be only slightly higher than randomly selected protein pairs. These limitations motivate a complex detection technique that does not depend on a PIN. So in this chapter we investigate whether clustering melting curve datasets can provide useful information about protein complexes.

Clustering is the task of summarizing a dataset in terms of a relatively small number of groups (clusters) such that objects in the same cluster are related to each other but not related to objects in other clusters [13]. Since

according to the TPCA observation the melting curves of a protein complex tend to be similar to each other but not necessarily dissimilar to the curves from other complexes, we investigate the extent to which protein complexes correspond to clusters in melting curve datasets.

Section 2.1 provides necessary background information on clustering. Section 2.2 describes the melting curve datasets on which our empirical analysis is based and our methods for clustering melting curve datasets. Section 2.3 answers the motivating question of this chapter through an empirical analysis.

2.1 Background

Clustering algorithms can be broadly divided into non-parametric and parametric algorithms. Sections 2.1.1 and 2.1.2 describe two popular non-parametric algorithms, and sections 2.1.3 and 2.1.4 describe two popular parametric algorithms. Section 2.1.5 discusses cluster validation, which aims to assess the quality of a partition.

2.1.1 Single Linkage Clustering

Single linkage clustering may be the most well-known non-parametric clustering algorithm. It requires a dataset and a dissimilarity measure. A dissimilarity measure quantifies dissimilarity between each pair of objects in a dataset. Euclidean distance (see Section 2.2.2 for details) is an example of a dissimilarity measure.

Single linkage clustering is an iterative algorithm with the following steps:

1. Place each object in its own cluster.
2. Compute the dissimilarity between every pair of clusters.
3. Merge the pair of clusters with the smallest dissimilarity.
4. Repeat steps (2) and (3) until all objects are in a single cluster.

The dissimilarity between a pair of clusters A and B is the minimum dissimilarity between a point in A and a point in B.

A run of single linkage clustering can be described by its dendrogram. A dendrogram specifies the dissimilarity at which each merge occurs. For example, Figure 2.1 shows the single linkage dendrogram when the dataset is the letters $\{\mathrm{A}, \mathrm{B}, \mathrm{F}, \mathrm{H}, \mathrm{K}\}$ and the dissimilarity measure d is "letters apart". At $d = 1$, clusters $\{\mathrm{A}\}$ and $\{\mathrm{B}\}$ are merged. At $d = 2$, $\{\mathrm{F}\}$ and $\{\mathrm{H}\}$ are merged. At $d = 3$, $\{\mathrm{F,H}\}$ and $\{\mathrm{K}\}$ are merged. And at $d = 4$, $\{\mathrm{A,B}\}$ and $\{\mathrm{F,H,K}\}$ are merged.

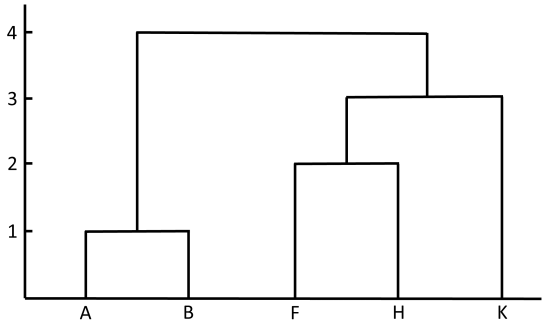

Figure 2.1: Dendrogram for single linkage clustering when the dataset is the letters $\{\mathrm{A}, \mathrm{B}, \mathrm{F}, \mathrm{H}, \mathrm{K}\}$ and the dissimilarity measure is letters apart.

A dendrogram can be viewed as a sequence of partitions. For example, the dendrogram of Figure 2.1 specifies the following sequence of partitions:

1. $d = 0$: $\{\{\mathrm{A}\}, \{\mathrm{B}\}, \{\mathrm{F}\}, \{\mathrm{H}\}, \{\mathrm{K}\}\}$
2. $d = 1$: $\{\{\mathrm{A,\ B}\}, \{\mathrm{F}\}, \{\mathrm{H}\}, \{\mathrm{K}\}\}$
3. $d = 2$: $\{\{\mathrm{A,\ B}\}, \{\mathrm{F,\ H}\}, \{\mathrm{K}\}\}$
4. $d = 3$: $\{\{\mathrm{A,\ B}\}, \{\mathrm{F,\ H,\ K}\}\}$
5. $d = 4$:$\{\{\mathrm{A,\ B,\ F,\ H,\ K}\}\}$

A dendrogram's sequence of partitions are called horizontal cuts. One strategy to select a single partition from a dendrogram is to choose the horizontal cut with the highest "clustering quality". Techniques to assess clustering quality are discussed in Section 2.1.5.

Figure 2.2 (A) shows a dataset containing 200 points, where each point is sampled from one of two uniform distributions. We applied single linkage

clustering to the dataset, and the two-cluster horizontal cut is shown. Notice that it perfectly separates the points by distribution.

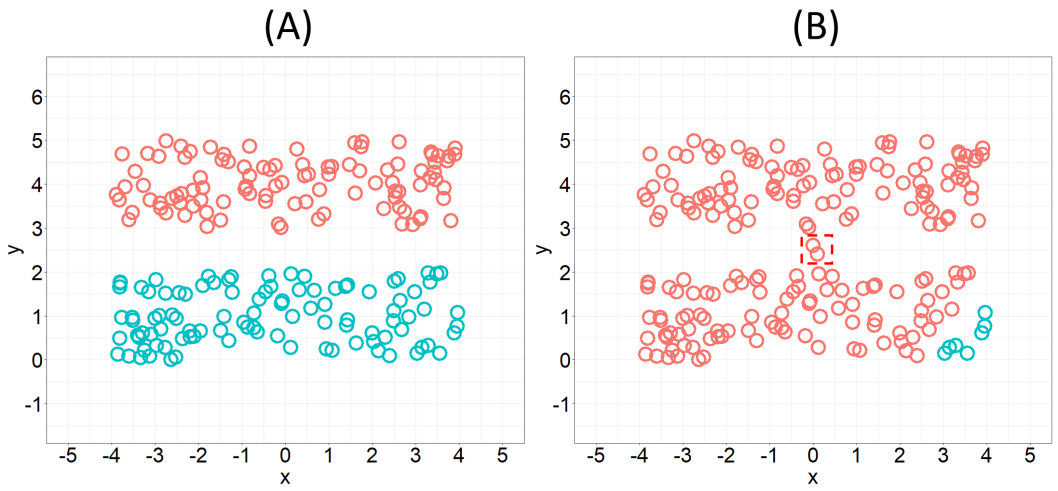

Figure 2.2: Two example datasets. (B) differs from (A) in that it contains two additional points. Single linkage clustering was run on both datasets. (A) and (B) show their respective two-cluster horizontal cuts

Figure 2.2 (B) shows the same dataset as (A), but with two points that do not belong to either distribution. These points can be viewed as outliers. We applied single linkage clustering to the dataset, and the two-cluster horizontal cut is shown. Notice that all but seven points are placed in a single cluster. The two-cluster horizontal cuts of (A) and (B) are very different.

Figure 2.2 shows that single linkage clustering is highly sensitive to outliers. Section 2.1.2 discusses HDBSCAN* [4], a density-based clustering algorithm that is not sensitive to outliers.

2.1.2 HDBSCAN*

HDBSCAN* is a hierarchical, density-based clustering algorithm that takes a dataset D, a dissimilarity measure d, and a positive integer called m_{pts} as input. HDBSCAN* defines an internal dissimilarity measure called mutual reachability distance (MRD) based on d and m_{pts}. MRD is then used to build a cluster hierarchy. As an optional step, HDBSCAN* can extract an optimal partition from its cluster hierarchy.

Mutual Reachability Distance

In order to define MRD, we must first define an object's core distance. The core distance of an object x, denoted $\text{core}_{m_{pts}}(x)$, is the distance to its m_{pts}-nearest neighbor. The 1-nearest neighbor of an object is considered to be itself, so $\text{core}_1(x) = 0$. Core distance can be though of as inverse measure of density for the region in which x is found. Figure 2.3 shows the core distances of three objects when $m_{pts} = 6$.

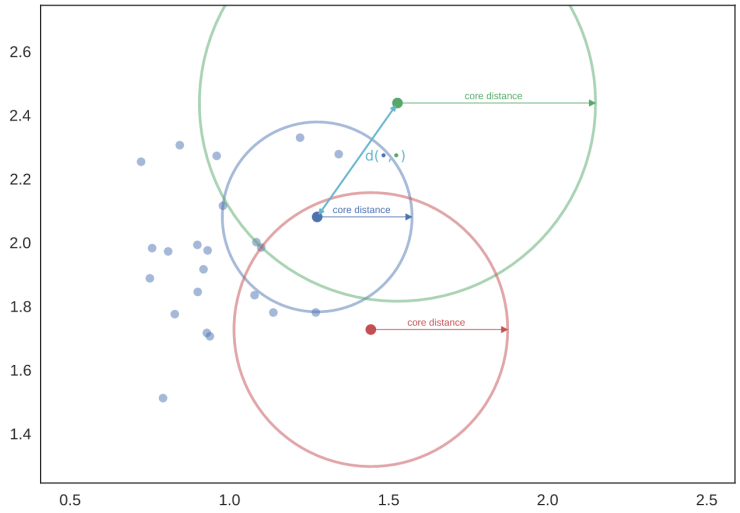

Figure 2.3: Synthetic dataset depicting core distances for three points. Figure taken from [27].

Given two objects x and y, their MRD is defined as:

$$\text{MRD}(x, y) = \max\{core_{m_{pts}}(x), core_{m_{pts}}(y), d(x, y)\}. \tag{2.1}$$

Observe that in Figure 2.3, the core distance of the green point is greater than the core distance of the blue point and the distance between the green and blue points. Thus, the MRD of the two points is the core distance of the green point.

Building the Cluster Hierarchy

The first step in building the cluster hierarchy is to use MRD to construct a minimum spanning tree (MST) M over the objects in D. Each node in the

hierarchy will correspond to a sub-MST of M, with the root node representing M in its entirety. The parameter m_{cl} specifies the minimum size of a cluster in the hierarchy. We follow the recommendation of [4] and set $m_{cl} = m_{pts}$ for all of our experiments.

The cluster hierarchy grows downward from the root by iteratively removing edges from M in order of decreasing edge weight. At each iteration, the focus is on the node furthest down the hierarchy that contains the edge being removed. If removing the edge results in the node's MST splitting into two sub-MSTs that both have at least m_{cl} objects, then two child nodes (corresponding to the two sub-MSTs) are added at the next level in the hierarchy. If one of the sub-MSTs has less than m_{cl} objects, then the node persists to the next level, although it is now corresponds to the larger sub-MST. If both child MSTs contain less than m_{cl} objects, then the node has no children and ceases to exist beyond the current level.

Figure 2.4 shows a dataset (A), its single linkage dendrogram (B), and how a cluster hierarchy is constructed for $m_{cl} = 2$. Observe that 13 edge removals are depicted on the dendrogram. Those in green (4 total) result in the formation of two child nodes (e.g. C_2 gives way to C_5 and C_4 when the edge connecting $\{6, 8, 7, 9, 5\}$ to $\{1, 3, 2, 4\}$ is removed; those in blue (4) result in a node decreasing in size (e.g. C_4 decreases in size when the edge connecting 4 to $\{1, 3, 2\}$ is removed); and those in gold (5) result in a node ceasing to exist after the split.

A cluster hierarchy can be far simpler than the dendrogram of the same dataset. However, each node must record the density level $\lambda = \frac{1}{\text{MRD}}$ at which it was formed, and the density level at which each object leaves the node (either from the node shrinking, disappearing, or splitting into two nodes). Let λ_{birth} be the λ value at which the node first forms, and for each object p, let λ_p be the λ value at which p leaves the node. The stability of a node is defined as:

$$S(\text{node}) = \sum_{p \in \text{node}} (\lambda_p - \lambda_{birth}). \tag{2.2}$$

Figure 2.5 gives the cluster hierarchy of the dataset in Figure 2.4, along with

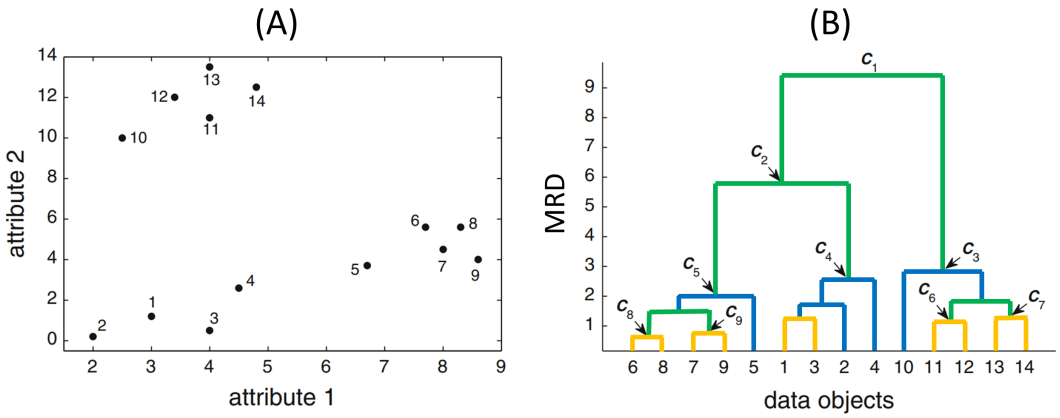

Figure 2.4: (A) Example dataset. (B) Dendrogram of dataset. Figure taken from [3]. Dendrogram colors are added and not in the original.

the stability of each node. As an example, we show how $S(C_4)$ is calculated. C_4 forms at $\lambda = \frac{1}{5.78}$, so $\lambda_{birth} = \frac{1}{5.78}$. Object 4 leaves the node at $\lambda = \frac{1}{2.56}$, object 2 at $\lambda = \frac{1}{1.72}$, and objects 3 and 1 leave when the node disappears, at $\lambda = \frac{1}{1.22}$. Thus $S(C_4) = (\frac{1}{2.56} - \frac{1}{5.78}) + (\frac{1}{1.72} - \frac{1}{5.78}) + 2 \cdot (\frac{1}{1.22} - \frac{1}{5.78}) = 1.92$.

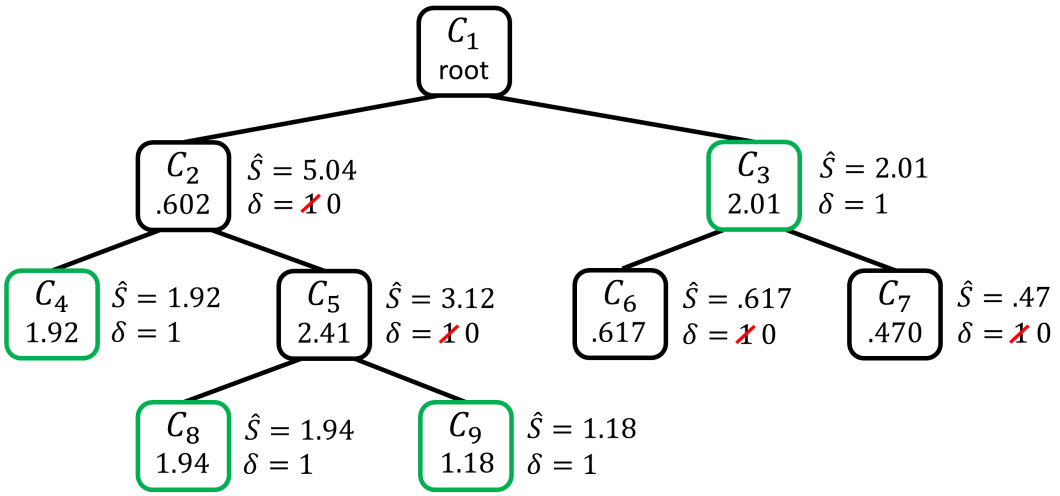

Figure 2.5: Cluster hierarchy for dataset in Figure 2.4. Inside each node is its stability. Green nodes form the optimal partition.

Extracting an Optimal Partition

One way to obtain a partition from an HDBSCAN* cluster hierarchy is through a horizontal cut, where any objects not contained in a node get classified as noise. There are two main drawbacks of a horizontal cut. First, it enforces

a single density threshold across each cluster. Second, the level at which to perform the cut is often unclear. To remedy these drawbacks, HDBSCAN* provides an algorithm that can automatically extract an optimal partition from a cluster hierarchy.

The cluster extraction algorithm requires each node's stability to be pre-computed. Each node C_i gets an indicator variable $\delta_i \in \{0, 1\}$, where $\delta_i = 1$ if C_i is selected and $\delta_i = 0$ if C_i is not selected. Initially, $\delta_i = 1$ for all i. By the end of the algorithm, the indicators that remain as 1 will choose the nodes that form the partition.

The algorithm works by iterating through the nodes, from the lowest level non-leaf node up to (but not including) the root node. At each iteration, certain indicators are set to 0 and the current node C_i's *total* stability $\hat{S}(C_i)$ is set. $\hat{S}(C_i)$ is the maximum sum of stabilities that can be obtained from a valid selection of nodes in the tree rooted at C_i. Accordingly, for each leaf node C_{leaf}, $\hat{S}(C_{leaf}) = S(C_{leaf})$.

At each iteration, the stability of the current node $S(C_{parent})$ is compared to the sum of the total stabilities of its children $\hat{S}(C_{child1}) + \hat{S}(C_{child2})$. If $S(C_{parent}) \geq \hat{S}(C_{child1}) + \hat{S}(C_{child2})$, then two things happen: each descendant of C_{parent} gets de-selected (i.e., $\delta_{descendant} = 0$), and $\hat{S}(C_{parent})$ gets assigned $S(C_{parent})$. If, on the other hand, $S(C_{parent}) < \hat{S}(C_{child1}) + \hat{S}(C_{child2})$, then the parent node gets de-selected (i.e., $\delta_{parent} = 0$), and $\hat{S}(C_{parent})$ gets assigned $\hat{S}(C_{child1}) + \hat{S}(C_{child2})$.

In Figure 2.5, C_5 gets de-selected because $S(C_5) < \hat{S}(C_8) + \hat{S}(C_9)$. Then C_6 and C_7 get de-selected because $S(C_3) > \hat{S}(C_6) + \hat{S}(C_7)$. Lastly, C_2 gets de-selected because $S(C_2) < \hat{S}(C_4) + \hat{S}(C_5)$.

We applied HDBSCAN* with $m_{pts} = 10$ to the datasets of Figure 2.2. The resulting partitions are shown in Figure 2.6. Apart from the two outliers in (B) but not (A), the only difference between the two partitions is that there are twelve more noise objects in (B) than in (A). These partitions demonstrate that HDBSCAN* is more robust to outliers than single linkage.

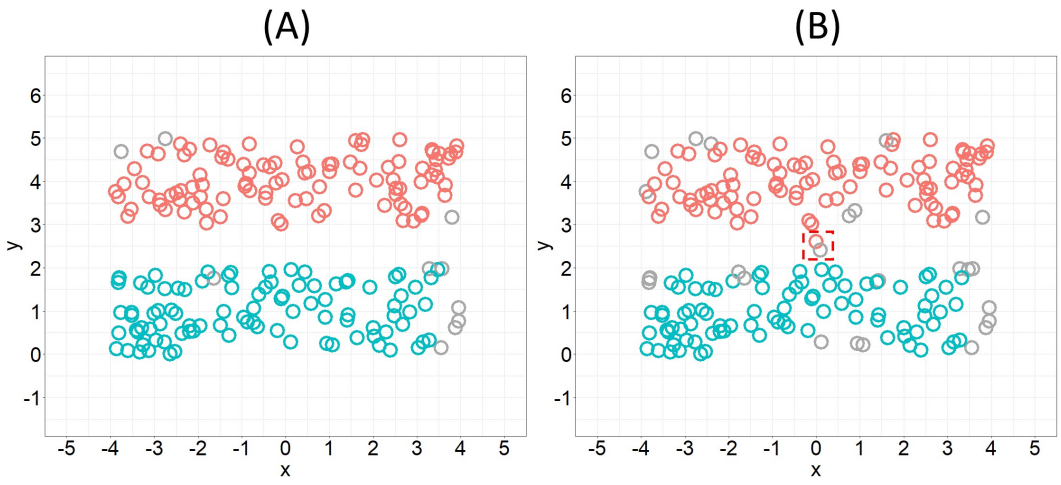

Figure 2.6: HDBSCAN* with $m_{pts} = 10$ applied to the dataset of Figure 2.2. Grey points are noise.

2.1.3 k-means

k-means may be the most well-known parametric clustering algorithm. It requires a dataset and an input parameter k. The goal of k-means is to partition the dataset into k clusters $C = \{C_1, ..., C_k\}$ such that within-cluster sum of squares (WCSS) is minimized. The objective function of WCSS is given below:

$$\text{WCSS}(C) = \sum_{i=1}^{k} \sum_{x \in C_i} dist(x, \mu_i)^2, \tag{2.3}$$

where μ_i is the mean of C_i and $dist(x, \mu_i)$ is the Euclidean distance between x and μ_i (see Section 2.2.2 for a description of Euclidean distance).

Minimizing Equation 2.3 is NP-hard [25], so there is not currently an efficient algorithm for this task and it's highly unlikely that there ever will be such an algorithm. In practice, the Expectation-maximization (EM) algorithm [12] is used to find a local minimum for Equation 2.3. The EM algorithm for k-means is quite simple and its steps are given below:

1. Initialize k centers.
2. *E-step*: Form k clusters $C = \{C_1, ..., C_k\}$ by assigning each object to its nearest center.
3. *M-step*: Recompute the k centers as the means of the clusters.

4. Compute WCSS(C).

5. Repeat steps 2-4 until the decrease in WCSS is less than some pre-specified amount.

The result of k-means can depend on how the k centers are initialized. In practice, K-MEANS can be run multiple times, each time with a random initialization of the k centers. A single partition can be selected by choosing the partition with the lowest WCSS.

It is sometimes unclear what value of k to use. In these cases, k-means should be run for multiple values of k. A cluster validation technique called silhouette analysis (Section 2.1.5) can then be used to choose the "highest quality" partition.

A shortcoming of k-means is that it is biased towards compact clusters. Section 2.1.4 discusses Gaussian Mixture Models, which can be viewed as a generalization of k-means that is not biased towards compact clusters.

2.1.4 Gaussian Mixture Models

Gaussian Mixture Models (GMMs) require a dataset and an input parameter k. GMMs assume that the dataset was created by a particular data-generating process. This process assumes that each object is sampled independently from one of k Gaussian distributions $\Theta = (\theta_1, ..., \theta_k)$, with the particular Gaussian determined by a set of k mixture weights $\Omega = (\omega_1, ..., \omega_k), \sum_{i=1}^{k} \omega_i = 1$. The goal of GMMs is to learn the parameters (Θ, Ω) that control this process. The GMM log-likelihood function for a dataset D is given below:

$$L(D \mid \Theta, \Omega) = \sum_{x \in D} \log \left[\sum_{i=1}^{k} \omega_i \cdot \mathcal{N}(x \mid \theta_i) \right], \tag{2.4}$$

where $\mathcal{N}(x \mid \theta_i)$ is the probabibility density function (PDF) value for x in the Gaussian distribution parameterized by θ_i.

Locally optimal parameters can be found through expectation-maximization [12]. The EM algorithm for GMMs is given below.

1. Initialize Θ and Ω. Note that each θ_i consists of a mean vector μ_i and a covariance matrix Σ_i.

2. *E-step*: For each object x, compute the probability that it comes from the Gaussian θ_i for $i = 1, ..., k$. I.e., compute:

$$\Pr[\theta_i \mid x] = \frac{\omega_i \cdot \mathcal{N}(x; \theta_i)}{\sum_j \omega_j \cdot \mathcal{N}(x; \theta_j)} \tag{2.5}$$

3. *M-step*: Recompute the parameters $\mu = (\mu_1, ..., \mu_k)$, $\Sigma = (\Sigma_1, ..., \Sigma_k)$ and $\Omega = (\omega_1, ..., \omega_k)$ according to the following equations:

$$\mu_i = \frac{1}{n_i} \sum_x \Pr[\theta_i \mid x] \cdot x \tag{2.6}$$

$$\Sigma_i = \frac{1}{n_i} \sum_x \Pr[\theta_i \mid x] \cdot (x - \mu_i)(x - \mu_i)^T \tag{2.7}$$

$$\omega_i = \frac{n_i}{n} \tag{2.8}$$

   where $n_i = \sum_x \Pr[\theta_i \mid x]$. I.e., n_i is the expected number of points assigned to θ_i.

4. Repeat steps 2 and 3 until the increase in likelihood is less than some pre-specified amount.

The first step of the EM algorithm is to initialize Θ and Ω. In our experiments, we use model-based hierarchical clustering (MBHAC) [14] for this task. MBHAC can be viewed as single linkage clustering with a different merge criterion. At each iteration, the merge that results in the smallest decrease in GMM classification likelihood [5] occurs. Once an MBHAC dendrogram is built, a k-component GMM can be initialized with the dendrogram's k-cluster horizontal cut.

By modelling clusters as Gaussian distributions, GMMs can find clusters of varying shapes, volumes and orientations. This flexibility comes at the price of a high number of parameters, which can make GMMs over-parameterized for smaller datasets. However [6] showed that it's possible to learn a GMM

with significantly fewer parameters by placing constraints on the eigenvalue decomposition of each covariance matrix. These constraints affect cluster volume, shape and orientation. E.g., if the volume constraint is in place, then each cluster contains approximately the same number of points.

When applying GMMs, it is often unclear what k should be and whether the model's covariance matrices should be constrained. A common strategy is to use an information criterion such as the Bayesian Information Criterion (BIC) [35] to score each candidate model. The model with the highest score is then selected. The equation for BIC is given below:

$$\text{BIC} = L(D \mid \Theta, \Omega) - \frac{\nu}{2} \log |D|, \tag{2.9}$$

where ν is the number of free parameters in the model. Like all information criterions, BIC seeks to find an appropriate trade-off between how well the model fits the data and its complexity.

2.1.5 Cluster Validation

Regardless of the input dataset, all of the clustering algorithms discussed in Sections 2.1.1 through 2.1.4 can return a partition. Cluster validation is used to assess the quality of a partition. Internal cluster validation (ICV) assesses quality based on properties internal to the dataset. External cluster validation (ECV) assesses quality based on agreement with an external ground-truth partition. We begin by discussing two ICV techniques (silhouette analysis and DBCV) and end by discussing a frequently used ECV technique.

Silhouette Analysis

Silhouette analysis measures the extent to which a partition consists of compact and well-separated clusters. For each point $x_i \in D$, a silhouette coefficient s_i is computed as follows:

1. Define a_i as the average dissimilarity between x_i and points in the same cluster as x_i.

2. For each cluster that x_i is not in, compute the average dissimilarity between x_i and the points in that cluster. Define b_i as the minimum such value.

3. $s_i = (b_i - a_i)/\max(a_i, b_i)$

A silhouette coefficient will always be between -1 and 1. A negative s_i mean that, on average, x_i is more similar to points in some other cluster than to points in its own cluster. A positive s_i means that, on average, x_i is more similar to the points of its own cluster than to the points of any other cluster. A cluster's silhouette width is the average silhouette coefficient of the cluster. A partiton's average silhouette width refers to the average silhouette coefficient over the entire dataset.

According to [20], a partition's average silhouette width can be interpreted as follows:

- 0.71-1.0: A strong clustering structure has been found.
- 0.51-0.70: A reasonable clustering structure has been found.
- 0.26-0.50: The clustering structure is weak and could be artificial.
- < 0.25: No substantial clustering structure has been found.

Silhouette analysis is commonly used to evaluate partitions found by k-means. Euclidean distance is used as the dissimilarity measure.

Density-based Cluster Validation

Density-based Cluster Validation (DBCV) [28] measures the extent to which a partition consists of density-based clusters, i.e., clusters that are regions of high density surrounded by regions of low density. For each cluster, a DBCV score is computed. This score is based on density sparseness and density separation. Density sparseness measures the minimum density within the cluster and density separation is the maximum density between the cluster and some other cluster. Both of these notions are based on a version of mutual reachability distance (MRD).

The MRD used by DBCV is the same as that used by HDBSCAN* except it employs a different core distance. The core distance employed by DBCV is known as the all-points core distance. Given a point $x \in D$ that belongs to cluster C_i, the all-points core distance of x is defined as follows:

$$a_{pts}coredist(x) = \left(\frac{\sum_{i=2}^{n_i} \left(\frac{1}{KNN(x,i)} \right)^d}{n_i - 1} \right)^{-\frac{1}{d}}, \tag{2.10}$$

where n_i is the size of cluster C_i, d is the dimension of D, and $KNN(x, i)$ is the dissimilarity between x and the i-nearest neighbor of x.

Given a cluster C_i, the first step in computing its density sparseness $DSC(C_i)$ is to use MRD to build a minimum spanning tree of C_i. $DSC(C_i)$ is defined as the maximum edge weight of the minimum spanning tree, and can be viewed as an inverse measure of the minimum density within C_i.

The density separation between two clusters C_i and C_j is the minimum MRD between a point in C_i and a point in C_j. This MRD is viewed as the maximum density between C_i and C_j.

Let $C = \{C_1, ..., C_k\}$ be a partition. With density sparseness and density separation defined, we can now define the DBCV score of a cluster $C_i \in C$ as follows:

$$V(C_i) = \frac{\min\limits_{1 \leq j \leq l, j \neq i} \big(DSPC(C_i, C_j)\big) - DSC(C_i)}{\max\left(\min\limits_{1 \leq j \leq l, j \neq i} \big(DSPC(C_i, C_j)\big), DSC(C_i) \right)} \tag{2.11}$$

Observe that C_i is necessarily between -1 and 1. Greater values indicate higher quality density-based clusters. The DBCV score of an entire partition C is computed as follows:

$$DBCV(C) = \sum_{i=1}^{l} \frac{|C_i|}{|D|} V(C_i) \tag{2.12}$$

DBCV is commonly used to evaluate partitions found by single linkage clustering and HDBSCAN*.

Adjusted Rand Index

The Adjusted Rand Index (ARI) [18] is based on the Rand Index [31]. The Rand Index requires a dataset $D = \{o_1, ..., o_N\}$, a clustering partition $X = \{X_1, ..., X_R\}$, and a ground-truth partition $Y = \{Y_1, ..., Y_S\}$. Each pair of objects $o_i, o_j, i \neq j$ of D falls into one of these four categories:

- *True Positive*: $o_i, o_j \in X_r$ for some r and $o_i, o_j \in Y_s$ for some s.
- *True Negative*:: $o_i, o_j \notin X_r$ for all r and $o_i, o_j \notin Y_s$ for all s.
- *False Positive*: $o_i, o_j \in X_r$ for some r but $o_i, o_j \notin Y_s$ for all s.
- *False Negative*: $o_i, o_j \notin X_r$ for all r but $o_i, o_j \in Y_s$ for some s.

Let TP, TN, FP, FN be the number of true positives, true negatives, false positives, and false negatives, respectively. The Rand Index is computed as follows:

$$R = \frac{TP + TN}{TP + TN + FP + FN}. \tag{2.13}$$

The Rand Index is 1 if there are no false positves or false negatives, and 0 if there are no true positives or true negatives. Intuitively, the Rand Index is the fraction of "correct choices" made over the pairs of of D.

ARI can be viewed as the Rand Index adjusted for chance. This means that if X is a random partition, the expected ARI between X and Y is 0. ARI ranges from -1 to 1. It is 1 when the Rand Index is 1 and -1 when the Rand Index is 0.

2.2 Datasets and Methods

This section introduces the datasets and methods used for our empirical analysis. Section 2.2.1 discusses the melting curve datasets on which our empirical analysis is based. Section 2.2.2 discusses three dissimilarity measures that can be used for melting curves. Section 2.2.4 discusses the methods we use to cluster melting curve datasets.

2.2.1 Melting Curve Datasets

The authors of [41] generated the melting curves of 8260 proteins from K562 intact (not lysed) cells. Of these 8260 proteins, 1387 collectively form 529 CORUM [16] complexes. Each of these complexes contains between 3 and 131 proteins, with the median size being 5 proteins.

Let D be all 8260 melting curves and $C = \{C_1, ..., C_{529}\}$ be the 529 CORUM complexes, where each C_i contains the melting curves for a particular complex. Let $\mathcal{C} = C_1 \cup ... \cup C_{529}$ be the dataset containing all 1387 melting curves. We have not clustered $\mathcal{C}$ for two reasons. First, since most of the complexes intersect (share a common protein) with at least one other complex, we cannot use complex membership to construct a ground-truth partition. Thus ARI could not be used to evaluate a clustering of $\mathcal{C}$. Second, we want to evaluate clustering algorithms on a range of melting curve datasets, not just a single, large one. This motivates the following procedure, which, given a positive integer m and a seed complex $C_i \in \mathcal{C}$, builds a melting curve dataset consisting of the curves from m non-intersecting complexes, one of them being C_i:

1. Add the melting curves in C_i to the dataset D.
2. Randomly select a complex C_j from C such that $C_j \cap D = \emptyset$.
3. Add C_j to D.
4. Repeat steps 2 and 3 until D contains the melting curves of m complexes.

When a melting curve dataset is partitioned according to complex membership, we refer to the resulting partition as the dataset's complex partition.

By setting $m = 3$ and making each complex in C the seed of one run of the above procedure, we obtained 529 3-complex melting curve datasets. By construction, each complex occurs in at least one dataset. The average size of a dataset is 22.4; the minimum size is 9; and the maximum size is 169. The dataset of size 169 contains a complex of size 131 (CORUM ID: 351).

2.2.2 Dissimilarity Metrics for Melting Curves

In order to be applied to melting curve datasets, single linkage clustering and HDBSCAN* require a dissimilarity measure that is appropriate for melting curve datasets. In this section, we introduce three such dissimilarity measures.

Euclidean Distance

Throughout [41], Euclidean distance is used as a dissimilarity measure for melting curves. Intuitively, Euclidean distance measures the length of the straight line between two points in space. Given two d-dimensional points $x = (x_1, ..., x_d)$ and $y = (y_1, ..., y_d)$, Euclidean distance is computed as follows:

$$dist(x, y) = \sqrt{\sum_{i=1}^{d} (x_i - y_i)^2} \tag{2.14}$$

When applied to melting curves, each melting curve is treated as a d-dimensional point, where the solubility at the lowest temperature corresponds to the first dimension, the solubility at the second lowest temperature corresponds to the second dimension, and so forth.

In Figure 2.7 (A), the salmon and blue melting curves differ by a solubility of .05 at each temperature (apart from the reference temperature), leading to a Euclidean distance of .112. In contrast, the salmon and green curves differ by a solubility of .20 at each temperature, leading to a Euclidean distance of .447.

Pearson Dissimilarity

Pearson correlation is a commonly used statistic to measure the degree to which two variables X and Y lie on a straight line. The Pearson correlation is 1 if the (x, y) pairs lie on a straight line with positive slope and -1 if the pairs lie on a straight line with negative slope. Given two d-dimensional points $x = (x_1, ..., x_d)$ and $y = (y_1, ..., y_d)$ whose mean values are $\bar{x}$ and $\bar{y}$, Pearson correlation is computed as follows:

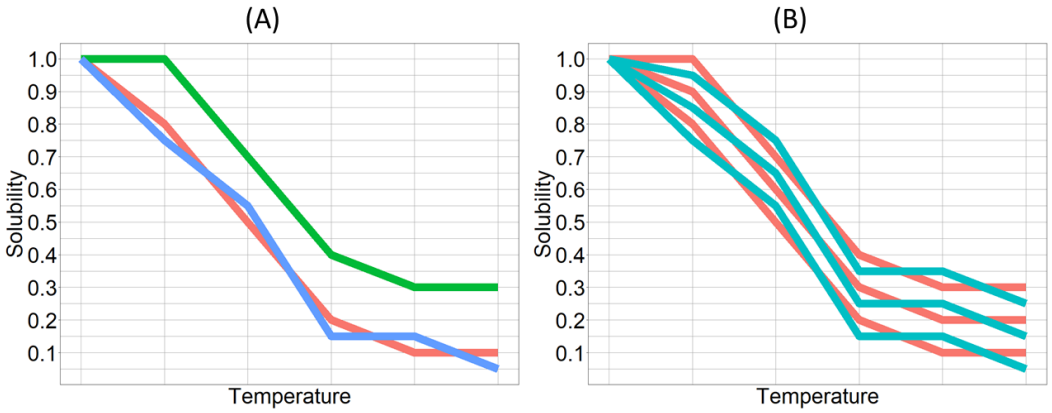

Figure 2.7: Two synthetic melting curve datasets. (A) Euclidean distance between red and blue: .112; red and green: .447. Pearson dissimilarity between red and blue: 8.10e-3; red and green: 0 (B) Melting curves from two overlapping complexes.

$$\phi'(x,y) = \frac{\sum_{i=1}^{d}(x_i - \bar{x})(y_i - \bar{y})}{\sqrt{\sum_{i=1}^{d}(x_i - \bar{x})^2 \sum_{i=1}^{d}(y_i - \bar{y})^2}} \tag{2.15}$$

Pearson correlation can be converted into a dissimilarity measure known as Pearson dissimilarity through the following transformation:

$$\phi(x,y) = \frac{1 - \phi'(x,y)}{2} \tag{2.16}$$

Pearson dissimilarity necessarily takes on a value between 0 and 1. The motivation for using Pearson dissimilarity is that it can detect dissimilarity in the shapes of melting curves.

In Figure 2.7 (A) there is some dissimilarity in the shapes of the red and blue curves, but the red and green curves have identical shapes (when the reference temperature is omitted). So the Pearson dissimilarity of the salmon and blue curves (8.10e-3) is greater than that of the red and green curves (0).

Figure 2.7 (B) contains the melting curves of two complexes (salmon and blue) that occupy the same region of melting curve space. We performed two runs of single linkage clustering on the dataset, once with Euclidean distance and once with Pearson dissimilarity. The two-cluster horizontal cut of the Euclidean distance dendrogram differed from the complex partition. This is unsurprising, given that each curve's nearest neighbor (according to Euclidean

distance) belongs to the other complex. However, when the same cut was performed on the Pearson dissimilarity hierarchy, the resulting partition matched the ground-truth partition. Thus, for melting curve datasets where distinct complexes occupy the same region of melting curve space, Pearson dissimilarity can be an effective dissimilarity measure.

2.2.3 Converting Melting Curve Datasets to Standardized Parametric Form

Euclidean distance and Pearson dissimilarity do not take advantage of the structured nature of melting curve datasets. According to [13], a dataset is structured if each observation consists of "repeated measures of the same outcome variable but under different conditions". Melting curve datasets are structured because each melting curve consists of repeated measures of solubility over a range of temperatures.

The authors of [41] found that melting curves have an underlying functional form that can be represented by a three-parameter log-logistic function with upper limit set to 1. Using t to represent temperature, the equation of this function is as follows:

$$f(t) = c + \frac{1 - c}{1 + \exp(b \log(\frac{x}{e}))} \tag{2.17}$$

where c, b and e are real-valued parameters and log refers to the natural logarithm. The parameter c is the lower limit of $f(t)$, meaning that $f(t)$ will approach but never cross the horizontal line $y = c.$; e is the temperature at which the inflection point occurs; b does not have a simple interpretation, but it appears to be related to the slope at the inflection point. We used the R package provided by [32] to fit three-parameter log-logistic functions to melting curves. Figure 2.8 plots the three-parameter log-logistic functions of two melting curves.

Suppose a three-parameter log-logistic function with upper limit set to 1 is fitted to a melting curve, and the function's resulting parameters are b, c and e. We refer to the vector (b, c, e) as the melting curve's parametric form. Given

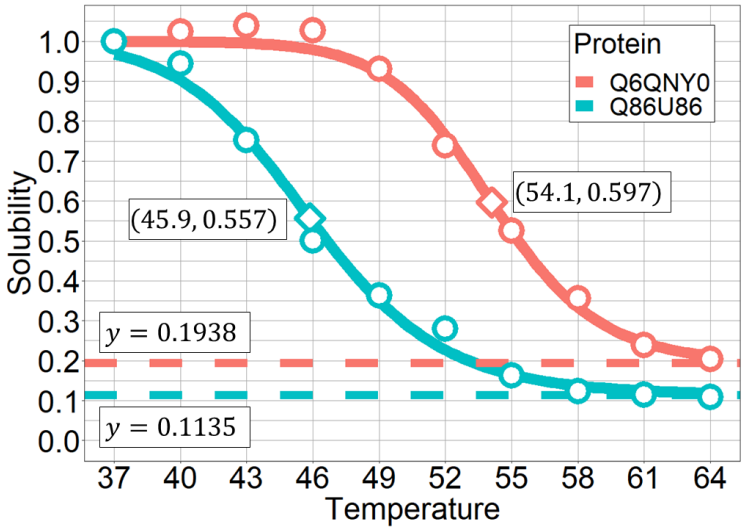

Figure 2.8: Three-parameter log-logistic functions fit to two melting curves. The diamond-shaped points are inflection points. The dotted lines are lower limits. The parameters of Q6QNY0 are $b = 22.3236$, $c = 0.1938$ and $e = 54.1$. The parameters of Q86U86 is $b = 15.2795$, $c = 0.1135$ and $e = 45.9$.

two melting curves' parametric forms $x_i = (b_i, c_i, e_i)$ and $x_j = (b_j, c_j, e_j)$, their Euclidean distance can be computed as follows:

$$dist(x_i, x_j) = \sqrt{(b_i - b_j)^2 + (c_i - c_j)^2 + (e_i - e_j)^2}. \tag{2.18}$$

For example, the parametric forms of Q6QNY0 and Q86U86 are $(22.3236, 0.1938, 54.1)$ and $(15.2795, 0.1135, 45.9)$, respectively, as shown in Figure 2.8. Their Euclidean distance is $\sqrt{(22.3 - 15.3)^2 + (.194 - .113)^2 + (54.1 - 45.9)^2} \approx 10.78$. However, if the Euclidean distance is computed only over the b and e parameters, then the resulting Euclidean distance is practically unchanged: $\sqrt{(22.3 - 15.3)^2 + (54.1 - 45.9)^2} \approx 10.78$, meaning that the values of c are not affecting Euclidean distance. This is due to c being on a very different scale than b and e. Whereas values of b and e routinely differ by more than 7 and 8, respectively, as shown in Figure 2.8, values of c are necessarily between 0 and 1, as c represents a log-logistic function's lower limit.

Given a dataset of melting curves in parametric form, each of b, c and e can be standardized. Standardization ensures that a parameter's values have mean 0 and standard deviation 1. We say that a melting curve dataset is in

standardized parametric form if each melting curve is in parametric form and each parameter is standardized. If the Euclidean distance of two vectors from such a dataset is computed, each parameter (including c) has the potential to affect the resulting Euclidean distance.

2.2.4 Clustering Procedures for Melting Curve Datasets

This section describes how the clustering algorithms of Sections 2.1.1 through 2.1.4 can be applied to the melting curve datasets of Section 2.2.1.

Single Linkage and HDBSCAN*

HDBSCAN* can be applied to a melting curve dataset using any one of the three dissimilarity measures discussed in Section 2.2.2. We consider four values for m_{pts}:

- $m_{pts} = 1$: Makes HDBSCAN* equivalent to single linkage clustering.
- $m_{pts} = 3$: Ensures that each node in the cluster hierarchy contains at least three melting curves. This is consistent with each CORUM complex containing at least three proteins.
- $m_{pts} = 5$: Makes HDSCAN* more robust to outlier curves.
- $m_{pts} = 9$: Maximum value of m_{pts} that can be applied to all of our datasets, as the smallest dataset contains 9 melting curves.

We now explain the details of how HDBSCAN* can be applied to a melting curve dataset and yield a single partition. Let HDB be the following procedure:

1. Run HDBSCAN* with $m_{pts} = 1, 3, 5$ and 9.
2. For each run, extract the optimal partition (according to the procedure outlined in Section 2.1.2).
3. Use DBCV to validate the four partitions from step 2.

4. Output the partition with the highest DBCV score.

There are three variants of HDB, depending on which dissimilarity measure is used and whether the melting curve dataset is in parametric form. We refer to these variants as HDB-EUCL (Euclidean distance), HDB-PEAR (Pearson dissimilarity), and HDB-PAR (Euclidean distance applied to a melting curve dataset in standardized parametric form).

Next, we explain how HDBSCAN* can be applied to a melting curve dataset without the procedure of Section 2.1.2. Let HDB-CUT be the following procedure:

1. Run HDBSCAN* with $m_{pts} = 1, 3, 5$ and 9.
2. For each of the four dendrograms, extract all horizontal cuts.
3. Use DBCV to validate each horizontal cut from step 2.
4. Output the partition with the highest DBCV score.

As with HDB, there are three variants of HDB-CUT: HDB-CUT-EUCL, HDB-CUT-PEAR and HDB-CUT-PAR.

We make use of the R package provided by [17] in our implementations of HDB and HDB-CUT.

k-means

In this section, we explain how k-means can be applied to a melting curve dataset. Even though we know each dataset contains three complexes, we let k range from 2 to 6. This reflects the practical situation where the number of complexes is unknown. Let K-MEANS be the following procedure:

1. For each k in $\{2, 3, 4, 5, 6\}$, run k-means 10 times, each time with a different random initialization of the k centers.
2. For each k, select the partition with the lowest WCSS.
3. For each k, compute the average silhouette width of the partition selected by WCSS.

4. Output the partition with the highest average silhouette width.

We refer to the procedure as K-MEANS-PAR when the melting curve dataset is in standardized parametric form. There is no K-MEANS-EUCL or K-MEANS-PEAR because k-means does not use a dissimilarity measure.

GMMs

In this section, we explain how GMMs can be applied to a melting curve dataset. As with k-means, we let the number of components range from 2 to 6. Let GMM be the following procedure:

1. Apply MBHAC to the dataset.
2. For each k in $\{2, 3, 4, 5, 6\}$, fit a GMM of each model type (14 total), using the MBHAC k cluster horizontal cut to initialize Θ and Ω.
3. Use BIC to score each partition (there are $5 \cdot 14$ of them).
4. Output the partition with the highest BIC score.

We refer to the procedure as GMM-PAR when the melting curve dataset is in standardized parametric form. There is no GMM-EUCL or GMM-PEAR because GMMs do not use dissimilarity measures. Our implementations of GMM and GMM-PAR make use of the R package provided by [36]

2.3 Results

This section describes our empirical analysis on whether clustering melting curve datasets can provide useful information about protein complexes. We began by applying the ten clustering procedures of Section 2.2.4 to the 529 melting curve datasets of Section 2.2.1. This resulted in 529 clustering partitions per clustering procedure. For each clustering partition, we used ARI to measure agreement with its corresponding complex partition.

Figure 2.2.4 shows the results in the form of boxplots. We see that for each clustering procedure, the ARI scores tend to be positive and skewed towards

higher values. For example, K-MEANS has a median of .103, a mean of .162, and 408 (77.1%) of its scores are positive.

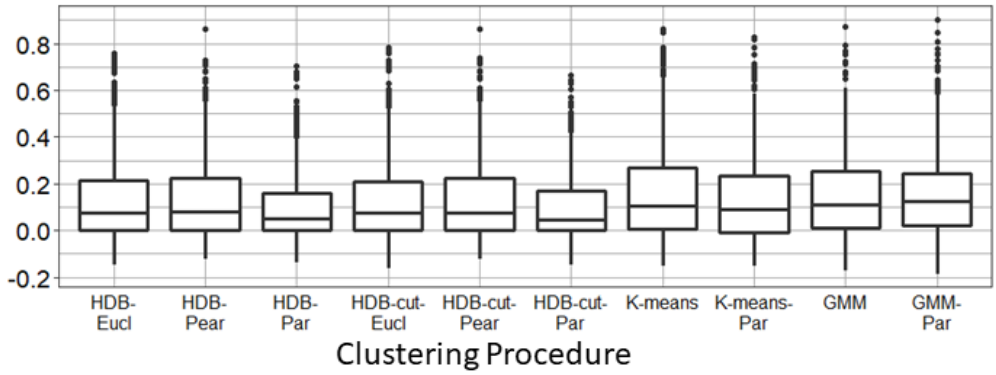

Figure 2.9: ARI scores for each clustering procedure.

For each clustering procedure, we performed a one-sided, one-sample t-test. The null and alternative hypothesis are given below:

- H_0: the mean of the ARI scores is 0.
- H_1: the mean of the ARI scores is positive.

Each test rejected the null hypothesis at the $p = 2.2\text{e-}16$ level. Thus clustering tends to provide information about a melting curve dataset's complex partition.

However, a partition with an ARI score of .103 (the median of K-MEANS) is unlikely to be of practical use to a scientist. This motivates the concept of "informative" partitions, which we define as those partitions with an ARI score of at least .5. Table 2.1 provides the count of informative partitions for each clustering procedure. K-MEANS has the highest count of informative partitions, at 49, but this still represents a rate of informative partitions below 10%.

2.3.1 Why are informative partitions relatively rare?

In this section, we use synthetic datasets to explain the lack of informative partitions. Figure 2.10 depicts four synthetic datasets, where the points represent

Clustering Procedure	# of Positive ARI Partitions	# of Informative Partitions
HDB-Eucl	361 (68.2%)	41 (7.75%)
HDB-Pear	346 (65.4%)	40 (7.56%)
HDB-Par	304 (57.5%)	37 (6.99%)
HDB-cut-Eucl	364 (68.8%)	46 (8.70%)
HDB-cut-Pear	365 (67.0%)	43 (8.13%)
HDB-cut-Par	338 (63.9%)	42 (7.94%)
K-means	408 (77.1%)	**49** (9.26%)
K-means-Par	379 (71.6%)	36 (6.81%)
GMM	409 (77.3%)	31 (5.86%)
GMM-Par	**422** (79.8%)	39 (7.37%)

Table 2.1: Counts of positive ARI partitions and informative partitions

melting curves and points of the same color belong to the same complex. We say that a complex is TPCA-consistent if its melting curves are similar to each other. We say that a group of complexes overlap if they occupy approximately the same region of melting curve space. Figure 2.10 (A) depicts four TPCA-consistent complexes that do not overlap; (B) depicts four TPCA-consistent complexes that do overlap; (C) and (D) depict the same complexes as (A) and (B), respectively, plus a large complex that is clearly TPCA-inconsistent.

Any clustering algorithm should find the complex partition of Figure 2.7 (A). However, recovering the complex partitions of (B), (C) and (D) is a more difficult matter, and it is unlikely that any clustering procedure could be completely successful. We applied four clustering procedures to the datasets of Figure 2.10, and computed the ARI of each clustering partition. The results are shown in Table 2.2

	A	B	C	D
K-means	1.00	**0.155**	0.229	0.0180
GMM	1.00	0.00835	0.229	**0.398**
HDB	1.00	0.00280	0.291	0.111
HDB-cut	1.00	0.00140	**0.355**	-0.0688

Table 2.2: ARI scores for clustering procedures applied to datasets of Fig 2.10.

The results for (B) show that if complexes overlap, clustering may be ineffective at placing the complexes in distinct clusters. The results for (C) show that the presence of a relatively large TPCA-inconsistent complex can lead

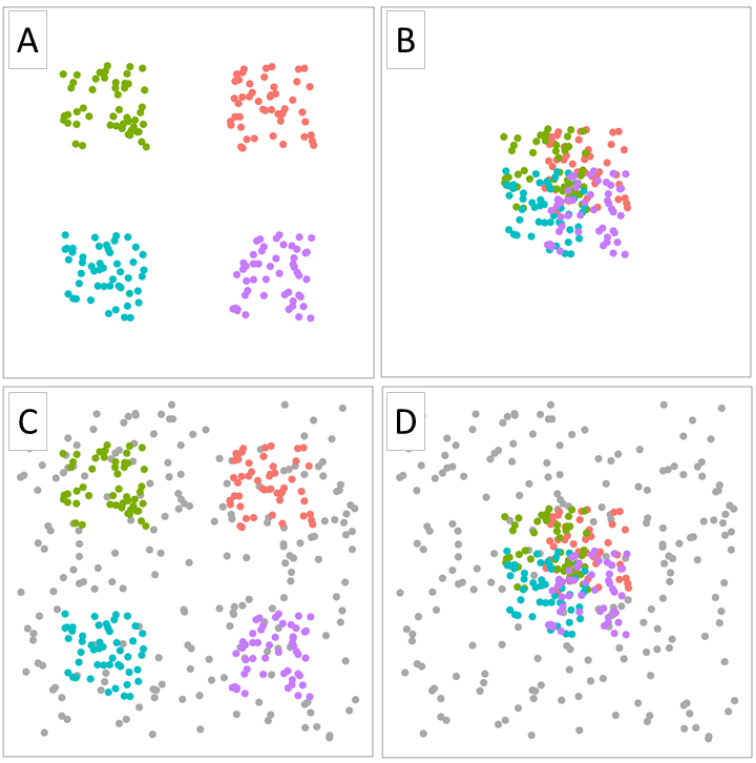

Figure 2.10: Synthetic datasets. Points should be viewed as melting curves and points of the same color as belonging to the same complex. The grey complex is of size 200. The other complexes are of size 50.

to each cluster containing some of its melting curves, resulting in the clusters being "impure". It is likely that informative partitions are rare because our melting curve datasets tend to have more in common with (B), (C) and (D) than they do with (A).

Figure 2.11 shows the melting curves of three complexes used in our melting curve datasets. Figure 2.11 (A) plots the melting curves of a TPCA-consistent complex. We make this designation because its p-value is .0018. See Section 1.2 for a description of how this p-value is computed. (B) plots the melting curves of another TPCA-consistent complex ($p = .0013$). (C) plots the melting curves from (A) and (B) together, demonstrating that TPCA-consistent complexes used in our melting curve datasets can overlap. (D) plots the melting curves of a large, TPCA-inconsistent complex. Although the median size of the complexes we considered is 5, this complex contains 131 proteins.

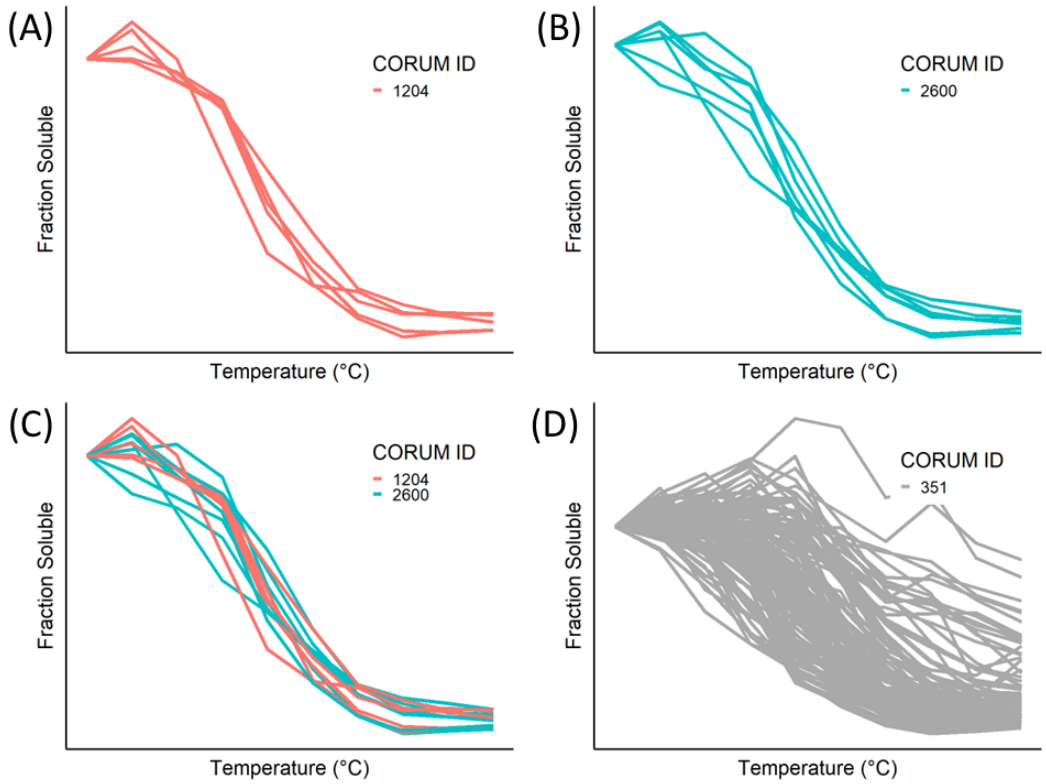

Figure 2.11: Melting curves of three complexes used in our melting curve datasets.

2.3.2 Can internal cluster validation scores identify informative partitions?

We have already established that informative partitions are relatively rare. However, if they are identifiable, then clustering melting curve datasets may still be of practical use for finding new protein complexes. In this section, we investigate whether informative partitions can be identified based on their ICV scores.

Recall that Single linkage/HDBSCAN* partitions can be validated by DBCV and that k-means/GMM partitions can be validated by average silhouette width. Figure 2.12 compares the ICV scores of informative partitions to those of non-informative partitions for each clustering procedure. It's clear that informative partitions tend to have higher ICV scores than non-informative partitions. To test the significance of these results, we performed a one-tailed Mann-Whitney test [26] for each clustering procedure. The null

and alternative hypothesis are given below:

- H_0: The informative scores are sampled from the same distribution as the non-informative scores.
- H_1: The informative scores are sampled from a distribution that is shifted to the the right of the distribution from which non-informative scores are sampled.

Each test rejected the null hypothesis at the $p = .05$ level.

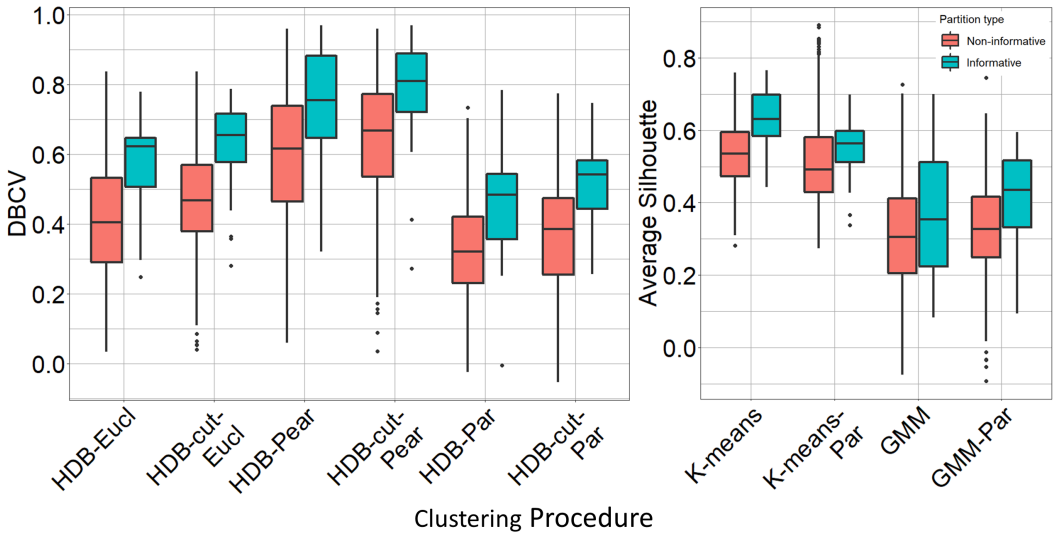

Figure 2.12: ICV scores of non-informative partitions vs. informative partitions. Salmom boxes are for non-informative scores and blue boxes are for informative scores.

Figure 2.12 shows that there is no clustering procedure for which *each* informative partition scores higher than *all* non-informative partitions. This means that ICV scores cannot perfectly identify informative partitions.

To better understand the predictability of ICV scores, we looked at the top 1, 5 and 10% of ICV scores for each clustering procedure, and calculated the fraction that correspond to informative partitions. We refer to these fractions as "precision at the top k%", or P@k%, for short.

Table 2.3 shows P@k% values for each clustering procedure. There is no clustering procedure with the highest P@k% for each k, but K-MEANS nearly satisfies this requirement. It is tied for highest P@1% and P@10%, and only

one informative partition away from being tied for highest P@5%. So for the task of predicting informative partitions from ICV scores, K-MEANS seems to be a good choice relative to the other clustering procedures.

Clustering Procedure	P@k%			Thresholds		
	$k = 1$	$k = 5$	$k = 10$	$k = 1$	$k = 5$	$k = 10$
HDB-EUCL	.333 (2)	.333 (9)	.302 (16)	.763	.670	.628
HDB-CUT-EUCL	.167 (1)	**.407 (11)**	**.378 (20)**	.784	.722	.673
HDB-PEAR	**.5 (3)**	. 333 (9)	.302 (16)	.946	.892	.843
HDB-CUT-PEAR	**.5 (3)**	.370 (10)	.283 (15)	.947	.895	.862
HDB-PAR	.333 (2)	.333 (9)	.283 (15)	.671	.546	.504
HDB-CUT-PAR	**.5 (3)**	.259 (7)	.283 (15)	.723	.603	.568
K-MEANS	**.5 (3)**	.370 (10)	**.378 (20)**	.745	.705	.669
K-MEANS-PAR	0 (0)	0 (0)	0 (0)	.848	.774	.707
GMM	.333 (2)	.185 (5)	.151 (8)	.660	.568	.504
GMM-PAR	0 (0)	.222 (6)	.245 (13)	.622	.541	.500

Table 2.3: P@k% values and corresponding thresholds for each clustering procedure. The value in parenthesis are the counts of informative partitions.

Table 2.3 also includes thresholds. A threshold is defined as the minimum validation score among the top k% of validation scores. These thresholds allow for P@k% values to be interpreted as probabilities. As an example, the $k = 5$ threshold for K-MEANS is .705. This means that if K-MEANS is applied to a melting curve dataset, and the resulting partition has an average silhouette width of at least .705, then we expect it to be informative with probability P@5% = .370. Thus, even if a K-MEANS partition has strong cluster structure, it is unlikely to be informative.

Based on the low P@k% values in Table 2.3, we conclude that internal validation scores cannot reliably identify informative partitions. Figure 2.13 provides an intuitive explanation for why this is the case. The blue points ($n = 100$) represent informative partitions while the salmon points ($n = 1000$) represent non-informative partitions. Points to the right of the dashed line are the top 10% of points ($n = 110$). The probability of a blue point being in the top 10% is $43/100 = .43$, but the probability of a salmon point being in the top 10% is only $67/1000 = .067$. However, due to salmon points outnumbering blue points 10:1, the fraction of blue points in the top 10% is only P@10% = $43/110 = .390$.

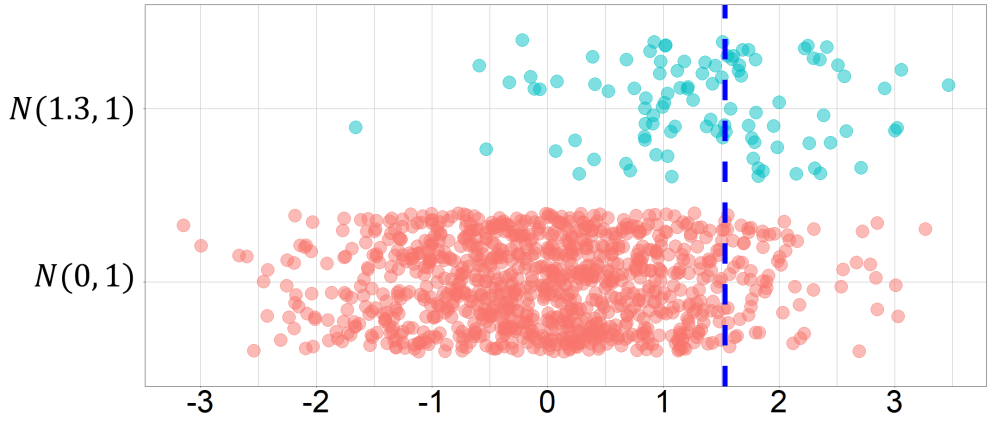

Figure 2.13: 100 points from Gaussian $\mathcal{N}(1.3, 1)$; 1000 points from Gaussian $\mathcal{N}(0, 1)$.

2.4 Conclusion

This chapter was motivated by the following question: can clustering melting curve datasets provide useful information about protein complexes? Section 2.1 provided the necessary background on clustering and Section 2.2 described the datasets and methods used in our empirical analysis.

We began the current section by applying the clustering procedures of Section 2.2 to each of the melting curve datasets. For each clustering partition, we used ARI to measure agreement with its corresponding complex partition. We found that clustering provides a statistically significant level of information about a melting curve dataset's complex partition. However, K-MEANS performed about as well as the other clustering techniques and its mean ARI score is only .162, which is not high high enough to be practically useful. So we turned our attention to "informative" partitions, which we defined as those with an ARI score of at least .5.

The motivation of Section 2.3.1 was to propose a plausible explanation for the relative rarity of informative partitions in our results. We used synthetic datasets to show that overlapping complexes and TPCA-inconsistent complexes can result in a melting curve dataset's clustering partition being non-informative. We then posited that most melting curve datasets contain one or both of these types of complexes.

Even though informative partitions are relatively rare, if they are identifiable, then clustering melting curve datasets may still be of practical use. Section 2.3.2 investigated whether ICV scores could be used to predict informative partitions. We found that informative partitions tend to have higher ICV scores than non-informative partitions, and that this difference is statistically significant.

To assess whether this difference is sufficient to identify informative partitions, we introduced a metric called P@k%, which computes the fraction of informative partitions among those with the highest k% of ICV scores. Table 2.3 shows P@k% values that are no larger than .5. This means that even if a partition has a very high ICV score (in the top k%), chances are that it is non-informative. We attributed this result to informative partitions being outnumbered by roughly 10:1.

Since informative partitions are relatively rare and they cannot be identified with a reasonable degree of accuracy, we conclude that clustering melting curve datasets does not provide useful information about protein complexes. However, clustering is not the only computational technique that can be applied to melting curves. In Chapter 3, we use information retrieval and metric learning to investigate whether melting curves can reduce the number of experiments necessary to find interactions.

Chapter 3

Are melting curves useful for limiting the number of validation experiments necessary to find protein interactions?

A pull-down assay is a biochemical technique used to detect potential interaction partners (prey) for a protein of interest (the bait) [23]. The assay relies on beads engineered to bind the bait. The first step in a pull-down assay is to incubate a purified solution of the bait with the beads, allowing the beads to bind to the bait. Next, a protein mixture is incubated with the bait-attached beads. During this incubation phase, interactors in the protein mixture will bind to the bait. After incubation, the beads are washed using a wash buffer to remove any unbound proteins. Finally, the bait-interactor complexes are detached from the beads and subjected to protein detection techniques, such as mass-spectrometry. We refer to the identified proteins as the bait's prey set. Unfortunately, the prey set will surely contain many proteins that do not interact with the bait (false positives).

There are several reasons for false positives in a prey set[1]. One is non-specific binding of proteins to the beads. This means that some non-baits in the protein mixture are likely to attach themselves to the beads. Moreover,

[1]See `https://www.sinobiological.com/category/ip-non-specific-binding` for more information.

when a non-bait attaches to the beads, it brings along its interactors, further increasing the number of non-interactors in the prey set. For reasons such as this, interactions can only be determined by follow-up validation experiments.

Given a pair of proteins, a validation experiment determines whether the pair interacts. A bait's prey set can be validated by performing a series of validation experiments, each involving the bait and a prey protein. It is therefore useful to order the prey set by some measure of "likelihood of interacting with the bait," and perform validation experiments in this order. This can be expected to reduce the number of experiments needed to find interactions, which is important given that validation experiments have associated costs of time and resources.

If both the bait and prey set have corresponding melting curves, then the prey set can be ranked in order of increasing melting curve dissimilarity from the bait (from smallest dissimilarity to largest). According to the TPCA observation, interacting proteins tend to have similar melting curves. Therefore, such a ranking can be expected to rank interactors highly, thereby limiting the number of validation experiments needed to determine interactions.

In this chapter we evaluate whether ranking a prey set in order of increasing melting curve dissimilarity from the bait is useful for limiting the number of validation experiments necessary to find interactions. Section 3.1 introduces information retrieval, which deals with the task of ranking documents in response to a query. The second part of Section 3.1 focuses on metric learning, which we consider for its potential to increase the performance of information retrieval systems. Section 3.2 describes how information retrieval can be used as a framework to evaluate the effectiveness of ranking prey sets by melting curve dissimilarity. And Section 3.3 contains our empirical evaluation.

3.1 Background

Section 3.1.1 describes information retrieval systems and how they can be evaluated. Section 3.1.2 introduces metric learning and overviews a class of metric learning algorithms.

3.1.1 Information Retrieval

Information retrieval (IR) systems rank objects in response to user queries. According to [34], "the typical interaction between a user and an IR system has the user submitting a query to the system, which returns a ranked list of objects that hopefully have some degree of relevance to the user's request with the most relevant at the top of the list."

Before a new IR system is deployed, it should be empirically evaluated and compared to existing systems. A typical evaluation requires a test collection and one or more evaluation measures. A test collection consists of three components:

- Q: a set of queries. Each query has a unique ID known as its *qid*.
- D: a set of objects known as "documents". Each document has a unique ID known as its *docid*.
- R: a set of relevance judgements in the form of $qid, docid$ pairs. R describes the documents in D that are relevant to each query in Q.

In response to each query in Q, an IR system ranks the documents in D, ideally so that relevant documents (as defined in R) are highly ranked. An IR system is evaluated by applying an evaluation measure to each ranking. In general, an evaluation measure evaluates the extent to which relevant documents are highly ranked.

In what follows, we describe three evaluation measures. To make their descriptions as clear as possible, we note that a binary sequence can represent a ranking and its relevance judgments. For example, Figure 3.1 shows a ranking of ten documents, where the first, second, fifth and seventh highest-ranked documents are relevant.

Precision/Recall at k

Precision at k and Recall at k, commonly referred to as P@k and R@k, respectively, are two evaluation measures used to evaluate IR system rankings.

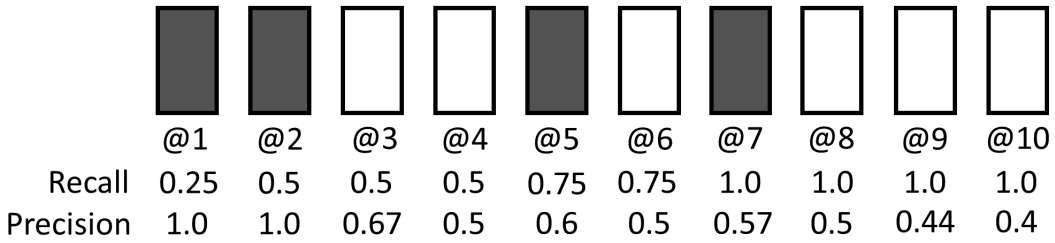

Figure 3.1: Example ranking with four relevant documents shown in grey.

For each run, they both compute a score based only on the k highest ranked documents. They are well-suited to evaluate IR systems for which the user is unlikely to examine more than k documents, such as internet search engines.

P@k computes the fraction of the k highest-ranked documents that are relevant. In Figure 3.1, P@5 = .6 because three of the top five documents are relevant. P@$k < 1$ if the total number of relevant documents is less than k.

R@k computes the fraction of relevant documents that are among the k highest ranked documents. The ranking in Figure 3.1 contains four relevant documents. R@5 = .75 because three of the four relevant documents are among the top five documents. R@k is non-decreasing as k increases and becomes 1 when k is the rank of the lowest-ranked relevant document.

P/R@k can vary considerably depending on the query. It is therefore standard to report the mean P/R@k value over all queries.

Average Precision

Average precision (AP) is different than P/R@k in that it evaluates a ranking in its entirety. AP is computed by calculating P@k at each k where there is a relevant document, and then taking the average of the P@k values. For example, in Figure 3.1 relevant documents occur at ranks 1, 2, 5 and 7, so AP is computed as follows:

$$\text{AP} = (\text{P@1} + \text{P@2} + \text{P@5} + \text{P@7})/4 = (1 + 1 + .6 + .57)/4 = .793 \quad (3.1)$$

The maximum value of AP is 1, which occurs when a ranking contains r relevant documents that occupy the top r positions in the ranking. AP can vary considerably depending on the query. It is therefore standard to report

the mean AP value over all queries, a measure known as mean average precision (mAP).

AP is not an absolute measure. This means that an AP value by itself is not enough to know whether the ranking is "good" or not. To see why, consider two rankings represented by the following binary sequences: 01 and 0100000000. Although both rankings have an AP score of .5, 01 is the worst of two possible rankings for a document set consisting of one relevant document and one non-relevant document, but 0100000000 is the second best of ten possible rankings for a document set consisting of one relevant document and nine non-relevant documents.

Instead of being an absolute measure, AP is a relative measure. This means that given two rankings on the same set of documents, AP can be used to assess which one is better. For example, consider two rankings represented by the following binary sequences: 01000 and 00010. The first has an AP score of .5 and the second an AP score of .25. In this way AP assesses that the first ranking is better relative to the second one.

It is common to apply two IR systems to a test collection and assess which one performs better in terms of mAP. However, in order to conclude that one system outperforms another, it is necessary to perform a statistical test and obtain a sufficiently small p-value. The authors of [38] show that a paired t-test is suitable for assessing whether two systems differ significantly in their respective mAP values. The test is conducted over the AP values of the two systems, which can be paired based on query. The null hypothesis is that the systems do not differ in their mAP values. The alternative hypothesis is that the systems do differ in their mAP values. In this chapter we conclude that one IR system outperforms another when $p < .05$.

3.1.2 Metric Learning

Metric learning is the task of learning a dissimilarity metric from the information provided in a set of training examples. This allows the resulting metric to incorporate domain-specific information. For d-dimensional vectors x and y, many metric learners learn a Mahalanobis metric $d_M(x, y)$, parameterized

by a $d \times d$ matrix M, where

$$d_M(x, y) = \sqrt{(x - y)^T M (x - y)}. \tag{3.2}$$

Observe that when M is the identify matrix, d_M corresponds to Euclidean distance. All the Mahalanobis learners discussed in [1] learn M such that $M = L^T L$ for some $k \times d$ matrix L. This allows the equation for d_M to be re-written as follows:

$$\begin{aligned} d_M(x, y) &= \sqrt{(x - y)^T M (x - y)} \\ &= \sqrt{(x - y)^T L^T L (x - y)} \\ &= \sqrt{(L(x - y))^T L (x - y)} \\ &= \sqrt{(Lx - Ly)^T (Lx - Ly)}. \end{aligned}$$

The final line of the derivation shows that a learned Mahalanobis metric can be viewed as Euclidean distance over a space that has undergone the linear transformation induced by L. This interpretation is one reason for the popularity of Mahalanobis learners.

Weakly Supervised Metric Learning

There are a number of different forms that training data for a metric learner can take. Here we consider metric learners that require weakly supervised training data.

Weakly supervised training data consists of a set of positive pairs of points $\mathcal{S}$ and a set of negative pairs of points $\mathcal{D}$. When a weakly supervised metric learner is applied to $\mathcal{S} \cup \mathcal{D}$, it attempts to construct a dissimilarity metric such that each positive pair gets a small dissimilarity score and each negative pair gets a large dissimilarity score.

A weakly supervised metric learner has been employed successfully if it results in a metric that generalizes to a new set of points. For example, suppose a metric learner is applied to a set of training pairs such that each positive pair corresponds to two pictures of the same person and each negative pair corresponds to two pictures of different people. Given a new pair of pictures,

the learned metric should assign a small dissimilarity score if the pictures are of the same person and a large dissimilarity score if they are of different people.

ITML

`metric-learn` [11] is a popular Python implementation of various metric learners. It implements three weakly supervised Mahalanobis learners: ITML [10], MMC [42], and SDML [30]. Of these three algorithms, ITML (Information Theoretic Metric Learning) is the only one we make use of in this chapter's empirical analysis. We therefore give an overview of ITML and explain why it is used over MMC and SDML.

ITML (Information Theoretic Metric Learning) assumes a $d \times d$ positive-definite[2] prior matrix M_0. Given a set of training pairs $\mathcal{S} \cup \mathcal{D}$, ITML seeks to learn M that approximates M_0 while also ensuring that $d_M(x, y)$ is small for $(x, y) \in \mathcal{S}$ and large for $(x, y) \in \mathcal{D}$. Since M must approximate M_0, it is desirable that d_{M_0} be a suitable metric for the problem at hand. Therefore, M_0 is often set to the identity matrix so that d_{M_0} is Euclidean distance.

Suppose p is a probability density function (PDF) and q is the PDF of a distribution that approximates p. The Kullback-Leibler (KL) divergence [21] of p and q, denoted $\mathrm{KL}(p \mid\mid q)$, measures the amount of information that is lost when q approximates p. If q is identitcal to p, $\mathrm{KL}(p \mid\mid q) = 0$, and the higher the KL divergence, the more information that is lost. $\mathrm{KL}(p \mid\mid q)$ is computed according to the following equation:

$$\mathrm{KL}(p \mid\mid q) = \int_x p(x) \log \frac{p(x)}{q(x)}. \tag{3.3}$$

Although M_0 and M are not probability distributions, they can be mapped to corresponding multivariate Gaussian distributions for a fixed μ:

$$p(x \mid A) = \frac{1}{(2\pi)^{d/2} \det(A)^{1/2}} \exp\left(-\frac{1}{2} d_A(x, \mu)\right), \tag{3.4}$$

where $A \in \{M_0, M\}$ and $det(A)$ denotes the determinant of A. ITML computes $\mathrm{KL}\left(p(x \mid M_0) \mid\mid p(x \mid M)\right)$ as a proxy measure for the extent to which M

[2] A $d \times d$ matrix M is positive-definite if it can be written as $M = L^T L$ for some $k \times d$ matrix L with linearly independent columns. $A \succ 0$ denotes that A is positive-definite.

approximates M_0. Thus, ITML learns M through the following optimization problem:

$$\begin{aligned} \min_{M \succ 0} \quad & \mathrm{KL}\left(p(x \mid M_0) \mid\mid p(x \mid M)\right) \\ \text{subject to} \quad & d_M(x_i, x_j) \leq u \quad \text{for all } (x_i, x_j) \in S, \\ & d_M(x_i, x_j) \geq l \quad \text{for all } (x_i, x_j) \in D. \end{aligned} \tag{3.5}$$

Note that u and l are user-provided threshold values, where u is the maximum dissimilarity a positive training pair can have and l is the minimum dissimilarity a negative training pair can have.

Given two $d \times d$ matrices A and B, LogDet divergence is defined as follows:

$$D_{ld}(A, B) = \mathrm{tr}(AB) - \log\det(AB) - d. \tag{3.6}$$

A result from [9] established a relationship between the KL divergence of two multivariate Gaussians and the LogDet divergence of their corresponding covariance matrices. The result implies that $\mathrm{KL}\left(p(x \mid M_0) \mid\mid p(x \mid M)\right) = \frac{1}{2} D_{ld}(M, M_0)$. Building on this result, the optimization problem of Problem 3.5 can be reformulated as follows:

$$\begin{aligned} \min_{M \succ 0} \quad & D_{ld}(M, M_0) + \gamma \sum_{i,j} \xi_{ij} \\ \text{subject to} \quad & d_M(x_i, x_j) \leq u + \xi_{ij} \quad \text{for all } (x_i, x_j) \in S, \\ & d_M(x_i, x_j) \geq l - \xi_{ij} \quad \text{for all } (x_i, x_j) \in D. \end{aligned} \tag{3.7}$$

The purpose of using LogDet divergence in the objective function is that unlike KL divergence, it is finite if and only if M is positive definite, thereby ensuring that M is positive definite without the need for additional constraints. Each training pair in Problem 3.7 has a corresponding slack variable. The slack variables guarantee that the constraints can be satisfied.

The objective function of Problem 3.7 consists of the sum of two terms. The first term $D_{ld}(M, M_0)$ measures the difference between M and M_0; the second term measures the amount of slack that is used to satisfy the constraints. In general, there is an inverse relationship between the amount of slack that is used and the difference between M and M_0. If γ is large, then using slack

is costly. This puts pressure on d_M to closely satisfy the constraints at the cost of increasing the difference between M and M_0. If γ is small, then using slack is less costly. This means that M can be learned such that M closely approximates M_0 at the cost of d_M only loosely satisfying the constraints.

ITML was the first algorithm to frame Mahalanobis learning as a LogDet optimization problem requiring a prior matrix. SDML (Sparse High-Dimensional Metric Learning) is similar to ITML in that it also relies on LogDet optimization and a prior matrix. The primary difference between SDML and ITML is that SDML explicitly seeks to learn a sparse matrix by minimizing the absolute sum of the learned matrix's off-diagonal elements. However, this aspect of SDML seems unnecessary for the application of metric learning in this chapter. Firstly, the melting curves to which we apply metric learners have only 9 dimensions, so there is not an exceedingly large number of parameters to learn. Secondly, we use ITML with its prior set to the identity, which implicitly ensures that ITML learns a sparse matrix. For these reasons, we do not expect SDML to outperform ITML in our application, and therefore do not use it.

MMC (Metric Learning with Application to Clustering with Side Information) was the first Mahalanobis distance learner. However, as its name suggests, it is designed to learn a metric for clustering applications. Since we do not perform clustering in this chapter, we do not use MMC.

3.2 Datasets and Methods

This section introduces the test collections and IR systems used to evaluate whether it is useful to rank a prey set based on increasing melting curve dissimilarity from the bait. Section 3.2.1 describes how a bait and prey set with corresponding melting curves can be modelled as a test collection, when combined with a set of ground-truth interactions to form the relevance judgments. Section 3.2.2 introduces some IR systems that can be applied to the test collections of Section 3.2.1.

3.2.1 Melting Curve Test Collections

In this section, we describe a procedure that constructs a test collection from a bait, its prey set, a set of melting curves, and a set of ground-truth interactions. Some notation is necessary to describe the details of this procedure. For a bait b, let prey(b) be the prey set of b, and for a protein $prot$, let loc($prot$) be the subcellular location of $prot$ according to the Human Protein Atlas[3] [37].

Given a bait b, a set of melting curves M, and a set of ground-truth interactions $ints$, the following procedure constructs a melting curve test collection if possible:

1. If $b \notin M$, exit the procedure.
2. Let $somePreys$ be all $prey \in$ prey(b) such that $prey \in M$ and loc($prey$) = loc(b). If $somePreys$ is empty, exit the procedure.
3. Let $someInts$ be all interactions in $ints$ such that one of the interactors is b and the other is in $somePreys$. If $someInts$ is empty, exit the procedure.
4. Construct a melting curve test collection by setting Q as the melting curve of b and D as the melting curves of $somePreys$. For each interaction $(b, prey) \in someInts$, form a relevance judgement with $qid = b$ and $docid = prey$.

Step 2 removes a protein from the prey set if it has a different subcellular location than the bait or does not have a melting curve in M. This is done because we are unaware of any interaction that involves proteins from different subcellular locations. Step 3 identifies the interactions that can serve as relevance judgments. If there are no such interactions, then a test collection is not formed. Step 4 identifies the test collection's query, documents and relevance judgments. We refer to the above procedure as CTC (Construction of Test Collections).

[3] https://www.proteinatlas.org/humanproteome/subcellular

Sources of data for CTC

Here we describe the data used as input to CTC. BioPlex 3.0 [19] is an online repository[4] of pull-down assay data for 10,128 human bait proteins. Each experiment was performed with a protein mixture derived from 293T cells. Tan et al. [41] published 7,945 melting curves[5] derived from 293T intact cells. They also compiled a set of 111,776 interactions[6] from multiple online databases, along with the number of scientific publications that support each interaction.

As discussed in the introduction of Chapter 2, [33] validated 33,000 purported interactions obtained from multiple online databases. They found that interactions reported by multiple publications had a significantly lower false positive rate than interactions reported by a single publication relying on a single detection method. Of the 111,776 interactions provided by [41], 13,601 are supported by multiple publications.

Due to the higher reliability of the 13,601 interactions supported by multiple publications, we use these as our ground-truth interactions. Unfortunately this set of ground-truth interactions is highly incomplete, as described below.

There are an estimated 650,000 interactions among human proteins [40]. Assuming that each gene codes for one protein, there are roughly 20,000 human proteins [29]. The interactions in our ground-truth set are collectively formed from 3852 proteins. Assuming that each pair of the 20,000 human proteins is equally likely to be an interaction, then in expectation there should be 24,107 interactions formed from the ground-truth proteins. We derived this by computing the expected value of the hypergeometric[7] distribution with its parameters set as $n = 650000$, $N = \binom{20000}{2}$, and $K = \binom{3852}{2}$. Thus, we estimate that our ground-truth is missing 10,506 interactions, 43.6% percent of its potential number of interactions.

[4]The BioPlex website is https://bioplex.hms.harvard.edu/. We specifically made use of the BioPlex 3.0 Unfiltered Interaction List dataset downloadable from the website.

[5]See Table S21 of the supplementary material of [41]

[6]See Table S2 of the supplementary material of [41]

[7]The hypergeometric distribution describes the probability of drawing k marked objects from a population of N objects with K total marked objects, when the draws occur without replacement. The expectation of the distribution is $n\frac{K}{N}$

Running CTC

We ran CTC for each of the 10,128 BioPlex baits, with M set to the 7,945 melting curves derived from 293T intact cell and *ints* set to our 13,601 ground-truth interactions. This resulted in 1414 melting curve test collections. As the number of test collections relative to the number of baits shows, most baits don't result in a test collection. This is due to CTC exiting in steps 1 through 3 for most baits, often because only 4039 of the 10,128 baits have corresponding melting curves.

Table 3.1 shows that there is considerable variation among the test collections, both in terms of the number of documents $|D|$ (size of the prey set) and the number of relevance judgements $|R|$ (number of interactors in the prey set). In general, only a very small percentage of a query's documents are relevant. For instance, the average number of documents in a test collection with 5 relevance judgements is about 278, meaning that less than 2% of documents are relevant to the query. Moreover, the test collection with the highest rate of relevant documents contains 33 relevant documents among its 89 total documents.

# of test collections	Mean $\|D\|$	Max. $\|D\|$	Min. $\|D\|$	Mean $\|R\|$	Max. $\|R\|$	Min. $\|R\|$
1414	276.3	1025	16	5.62	56	1

Table 3.1: Some statistics of the test collections

3.2.2 Information Retrieval Systems for Melting Curve Test Collections

A standard way to create an IR system is by way of a dissimilarity measure that can be applied to each possible query-document pair. For a given query, the documents can then be ranked in order of increasing dissimilarity from the query. Since the query and documents of each of our test collections are melting curves, we used the dissimilarity measures of Section 2.2.2 to create three IR systems: IR-Eucl (Euclidean distance), IR-Pear (Pearson dissimilarity), and IR-Par (Euclidean distance applied after the melting curves of the query

and documents have been converted to standardized parametric form).

In addition to IR systems based on existing metrics, we also used ITML to create two categories of learned-metric IR systems. IR-ITML-ALL systems are based on ITML metrics learned on both positive and negative training pairs, while IR-ITML systems are based on ITML metrics learned only on positive training pairs. Each positive pair corresponds to the melting curves of an interaction, and each negative pair corresponds to a random pair of melting curves, which can be viewed as a non-interaction. Although it is possible to select a random pair of melting curves that corresponds to an interaction, it is highly unlikely. Indeed, out the $\binom{20000}{2}$ pairs of proteins, only an estimated 650,000 (.325%) correspond to interactions.

3.3 Results

We applied IR-EUCL to each of the test collections from Section 3.2.1. This resulted in 1414 rankings, one for each test collection. The mAP of these rankings is .148. As discussed in Section 3.1.1, mAP is a relative measure, so more context is necessary to interpret the mAP of IR-EUCL.

To see if IR-EUCL generates better-than-random rankings, we constructed an empirical distribution of mAP for a random ranking system applied to our test collections. Below is the procedure we followed:

1. Generate a random ranking of each test collection's documents.
2. Compute the mAP of the 1414 random rankings
3. Repeat steps 1 and 2 10,000 times.

Performing the above procedure resulted in an empirical distribution of 10,000 mAP values. Since the mAP of IR-EUCL is higher than all 10,000 mAP values in the distribution, we conclude that IR-EUCL outperforms a random ranking system. This means that compared to performing validation experiments in random order, IR-EUCL on average reduces the number of validation experiments needed to find interactions between a bait and prey

set, although at this point we do not know the magnitude of this reduction or how often it occurs.

Next, we applied IR-Pear and IR-Par to the test collections and obtained mAP values of .153 and .118, respectively. With the mAP values of IR-Eucl and IR-Pear being so close, we performed a paired t-test over their respective AP values to see how they compare. We obtained $p = .0877 > .05$ and therefore do not have enough evidence to conclude that IR-Pear outperforms IR-Eucl.

3.3.1 How often do information retrieval systems reduce the need for validation experiments?

So far we have seen that IR-Eucl on average reduces the number of validation experiments needed to find interactions between a bait and prey set. However, this does not imply that IR-Eucl always or even often reduces the need for validation experiments. Indeed, it is possible that IR-Eucl rarely accomplishes this, but when it does, the reduction is large. Therefore, we now investigate how often IR-Eucl reduces the need for validation experiments.

One way to assess the quality of a ranking is to find the probability that its AP is higher than that of a random ranking. For each of our test collections, we constructed an empirical AP distribution for a random ranking system by generating 10,000 random rankings of the test collection's documents, and computing the AP of each ranking. Given a ranking of one of our test collections, we define the ranking's "improvement probability" (IP) as the fraction of values in the corresponding empirical AP distribution that are smaller than the ranking's AP. For each test collection, we computed the IP of the rankings generated by IR-Eucl, IR-Pear, and IR-Par. The results are summarized in Table 3.2.

Table 3.2 shows that 75.5% of IR-Eucl rankings have an improvement probability exceeding .5 and 31.3% have an improvement probability exceeding .95. A practical interpretation of these results is that for 75.5% of pull-down assays, IR-Eucl has at least a .5 chance of reducing the need for validation experiments. Moreover, for 31.3% of pull-down assays, IR-Eucl has at least a

IR system	mAP	% of rankings with IP > .95	% of rankings with IP > .5
IR-EUCL	.148	31.3	**75.5**
IR-PEAR	**.153**	**32.4**	73.7
IR-PAR	.118	23.8	72.6

Table 3.2: Percentage of rankings with improvement probabilities greater than .95 and .5

.95 chance of reducing the need for validation experiments.

3.3.2 How many validation experiments are needed to find a single interactor?

In this section, we quantify the number of validation experiments necessary to find a single interaction between a bait and prey set. Of course, the size of a prey set and the number of interactors it contains affects how many experiments are required. To control for this, we standardized each of our test collections to contain exactly 50 documents (prey proteins), with at most five of them being relevant (interactors). Given a test collection (Q, D, R), our procedure for standardizing it is described below.

1. Create a new set of relevance judgments R_{new}. If $|R| > 5$, let R_{new} be a random sample of 5 relevance judgments from R. If $|R| \leq 5$, let $R_{new} = R$.

2. Create two subsets of D, one made up of relevant documents (D_{rel}) and the other made up of non-relevant documents (D_{nrel}). Let D_{rel} be all documents in R_{new} and D_{nrel} be all documents in D that are not in R.

3. If $|D_{rel}| + |D_{nrel}| < 50$, exit the procedure.

4. Create a new set of documents D_{new}. Let D_{new} consist of D_{rel} plus a random sample of $50 - D_{rel}$ documents from D_{nrel},

5. Modify the test collection by making $D = D_{new}$ and $R = R_{new}$.

Step 1 creates a new set of relevance judgements R_{new}. Step 2 ensures that the new document set will not contain any interactors without a corresponding

relevance judgment in R_{new}. Steps 3 and 4 create the new document set D_{new} (if possible). We applied the above procedure to our test collections, which resulted in 1,383 standardized test collections. Test collections with one relevant document represent prey sets where $\frac{1}{50} \times 100 = 2\%$ of proteins interact with the bait. Similarly, test collections with two, three, four, or five relevant documents represent prey sets where 4%, 6%, 8%, or 10% of proteins interact with the bait.

We applied IR-EUCL to each of the standardized test collections, recording the rank of the highest-ranked relevant document. Figure 3.2 shows that there is considerable variation in the number of experiments necessary to find one interaction. In fact, among test collections containing only a single relevant document, there are 50 (of 460) where the lone relevant document gets the highest rank and five where it gets the lowest rank. Still, for each category of test collection, the highest density region occurs between ranks one and five, indicating that IR-EUCL tends to reduce the number of experiments to find one interaction.

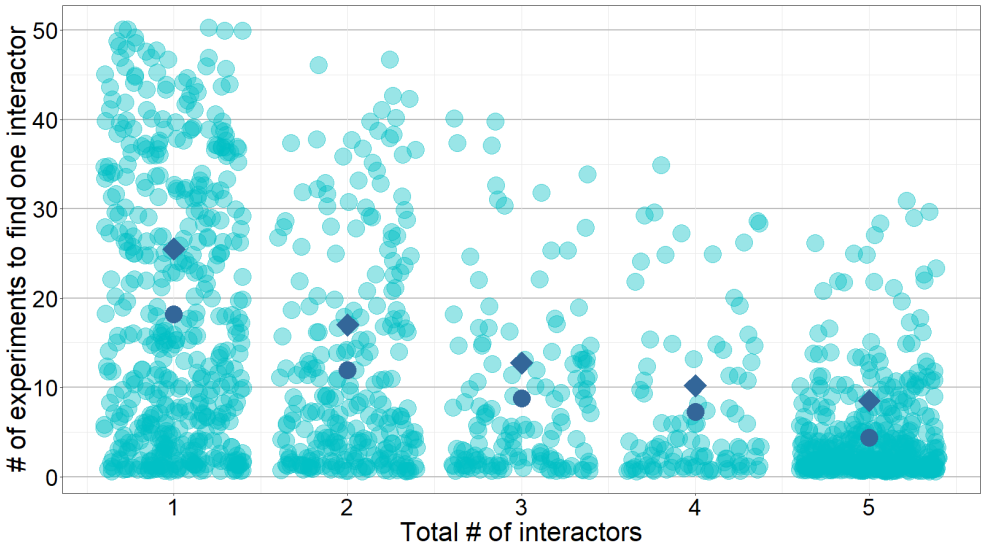

Figure 3.2: Number of experiments to find a single interactor when IR-EUCL is used to rank prey sets of size 50 with between 1 and 5 interactors. A category's navy point is the mean number of experiments. A category's navy diamond is the theoretical expected number of experiments if experiments are performed in random order according to the negative hypergeometric distribution.

Observe that each category in Figure 3.2 contains both a navy point and a navy diamond. The navy point corresponds to the mean number of experiments needed to find a single interaction when IR-EUCL defines the order of experiments, while the navy diamond is the theoretical expected number of experiments needed to find a single interaction, assuming prey proteins are processed in random order. This expectation is derived from the negative hypergeometric distribution [39], explained below.

Consider a population of N objects, M of which are marked. Suppose objects are randomly drawn without replacement from the population until $m \leq M$ marked objects are obtained. Then the number of objects X in the sample follows the negative hypergeometric distribution. The expected value of X is given by the following equation:

$$E(X) = m\frac{N+1}{M+1} \tag{3.8}$$

In our context N corresponds to the size of the prey set, M to the number of interactors in the prey set, and m to the desired number of interactions. This means that for each test collection, $N = 50$, $M \in \{1, 2, 3, 4, 5\}$, depending on how many relevant documents it contains, and $m = 1$. Thus, the expected number of experiments needed to find a single interaction in a prey set of size 50 when prey are processed in random order is $\frac{51}{M+1}$, as shown by the blue diamonds in Figure 3.2.

Table 3.3 reports some summary statistics for the results in Figure 3.2. We say a ranking represents a "single experiment success" if the highest ranked document is relevant. Although most rankings do not correspond to single experiment successes, some do. Among test collections containing a total of five relevant documents (modelling prey sets in which 10% of proteins are interactors), nearly 40% of rankings are single experiment successes.

Table 3.3 also shows that, regardless of the total number of interactors in a prey set, IR-EUCL on average results in fewer experiments to find one interaction than would be expected if the experiments were performed in random order. The mean reduction varies from 28.6% for test collections that con-

Total # of interactors	# of rankings	# of 1-expt successes	Expected # of expts	Mean # of expts	Mean reduction in # of expts	# of expt-reducing rankings
1	460	50 (10.9%)	25.5	18.2	7.3 (28.6%)	306 (66.5%)
2	246	44 (17.9%)	17	11.9	5.1 (30%)	176 (71.5%)
3	135	32 (23.7%)	12.75	8.75	4 (31.4%)	99 (73.3%)
4	98	29 (29.6%)	10.2	7.28	2.92 (28.6%)	72 (73.5%)
5	444	176 (39.6%)	8.5	4.40	4.1 (48.2%)	373 (84%)

Table 3.3: Summary statistics for IR-Eucl applied to standardized test collections. Column 3 counts the number of single experiment successes. Column 4 is the expected number of experiments to find a single interactor, according to the negative hypergeometric distribution. Column 7 is the number of prey sets for which an interactor is found in fewer experiments than would be expected if the experiments were performed in random order.

tain one or four relevant documents, to 48.2% for test collections that contain five relevant documents. The mean percentage reduction irrespective of the number of relevant documents is 35.4%.

Lastly, Table 3.3 shows that for most prey sets, IR-Eucl can be expected to reduce the number of experiments necessary to find one interaction. The percent probability of reduction varies from 66.5% for test collections with one relevant document up to 84% for test collections with five relevant documents

3.3.3 Comparing Information Retrieval Systems Based on Learned Metrics to IR-Eucl

As in previous sections, we consider a scientist who has the melting curves of a pull-down assay, and who would like to determine interactions between the bait and proteins in the prey set. New to this section is the additional assumption that the scientist has the melting curves of some proteins that are known to interact with the bait. Although the scientist could use IR-Eucl to define an order in which to perform validation experiments on the prey set, such an approach ignores the melting curves of the known interactors, potentially leaving useful information on the table. In this section, we describe how the melting curve of a bait and the melting curves of its known interactors can be combined to form a training set suitable for weakly supervised learning. We then compare IR systems based on learned metrics to IR-Eucl.

Devising a training set suitable for weakly supervised metric learning

Let b be the melting curve of a bait and $c_1, ..., c_n$ be the melting curves of n proteins known to interact with the bait. Pairing b with each of $c_1, ..., c_n$ forms a set of positive pairs: $\mathcal{S} = \{(b, c_1), ..., (b, c_n)\}$. Similarly, pairing b with m randomly selected melting curves $c'_1, ..., c'_m$ forms a set of negative pairs $\mathcal{D} = \{(b, c'_1), ..., (b, c'_m)\}$. Whereas each positive pair corresponds to a bait and an interactor, each negative pair can be viewed as a bait and a non-interactor.

If a weakly supervised metric learner is applied to $\mathcal{S} \cup \mathcal{D}$, the resulting metric will place interactors in $\mathcal{S}$ close to the bait and likely non-interactors in $\mathcal{D}$ far from the bait. It is reasonable to expect the interactors in $\mathcal{S}$ to bear some degree of melting curve similarity to the interactors in b's prey set, as the TPCA observation suggests that all interactors of b should have similar melting curves to b (and thus to each other). It's also reasonable to expect the melting curves of non-interactors in $\mathcal{D}$ to represent the distribution of non-interactor melting curves in b's prey set. If both these expectations hold, the learned metric should generalize to the prey set and place interactors close to the bait and non-interactors far from the bait. Thus, ranking a prey set according to a learned metric could outperform IR-Eucl.

Evaluating learned-metric information retrieval systems

We began by modifying our test collections to make them suitable for the task of comparing learned metric IR systems to IR-Eucl. Once a test collection is modified, we refer to it as an ML-compatable test collection, where ML is short for "Metric Learning". For each test collection in which it was possible, we extracted ten documents to form a training set suitable for a weakly supervised metric learner. Given a test collection (Q, D, R), our procedure works as follows:

1. If the test collection contains 5 or fewer relevant documents or 5 or fewer non-relevant documents, exit the procedure.

2. Extract a random sample $D_1 = \{d_1, d_2, d_3, d_4, d_5\}$ of 5 relevant docu-

ments from D. Remove their corresponding relevance judgments from R.

3. Extract a random sample $D_2 = \{d_6, d_7, d_8, d_9, d_{10}\}$ of 5 non-relevant documents from D.

4. Form a set of positive pairs $\mathcal{S}$ by pairing the query q with each document in D_1 so that $\mathcal{S} = \{(q, d_1), ..., (q, d_5)\}$, and form a set of negative pairs $\mathcal{D}$ by pairing the query q with each document in D_2 so that $\mathcal{D} = \{(q, d_6), ..., (q, d_{10})\}$.

Step 1 ensures that the test collection contains sufficient documents to still be meaningful after the extractions of steps 2 and step 3. We note that 1052 of 1414 test collections are eleminated due to having five or fewer relevant documents, but no test collections are eleminated due to having five or fewer non-relevant documents. D_1 serves to model the melting curves of known interactors and D_2 contains a random sample of melting cures that can be viewed as non-interactors. Step 4 creates a set of training pairs suitable for a weakly supervised metric learner

We applied IR-ITML, IR-ITML-ALL and IR-EUCL to each of the 362 ML-compatible test collections, and computed the mAP of each system. Table 3.4 summarizes the results and shows that IR-ITML has the highest mAP. We performed a paired t-test over the AP values of IR-ITML and IR-EUCL and obtained a p-value of 2.01e-4, showing that IR-ITML has a significantly higher mAP than IR-EUCL. This means that IR-ITML on average generates better rankings than IR-EUCL, although how much better is uncertain at this point. It additionally shows that ITML is capabale of learning a metric that not only places the known interactors in a training set close to the bait, but also generalizes to place prey set interactors close to the bait.

We then performed a paired t-test over the AP values of IR-ITML and IR-ITML-ALL and obtained a p-value of 6.63e-9, showing that IR-ITML significantly outperforms IR-ITML-ALL. All things being equal, machine learning algorithms perform better as the amount of training data increases.

IR system	mAP	# of rankings with IP > .95	# of rankings with IP > .5
IR-ITML	**.257**	**182** (50.3%)	**314** (86.7%)
IR-ITML-ALL	.241	175 (48.3%)	**314** (86.7%)
IR-EUCL	.224	156 (43.1%)	291 (80.4%)

Table 3.4: Performance of IR systems on the 362 ML-compatible test collections in terms of mAP and improvement probability

For this reason, we need to explain why IR-ITML, whose metrics are based on smaller training sets than IR-ITML-ALL, is the better performing of the two.

One possible explanation relates to the negative pairs on which IR-ITML-ALL was trained. Its negative pairs are formed from randomly selected melting curves, making them highly variable. Thus, by chance some negative pairs could be very close to some positive pairs, which could block some positive pairs from affecting the resulting ITML metric. In such a case, we would expect IR-ITML, which does not use negative pairs, to learn a more effective metric than IR-ITML-ALL

We also computed the improvement probability of each ranking according to the empirical procedure outlined in Section 3.3.1. Table 3.4 shows the number and percentage of rankings with IP > .95 and IP > .5 for each of the three IR systems. Compared to IR-EUCL, IR-ITML generates more rankings with an improvement probability of at least .5 (314 vs. 291) and more rankings with an improvement probability of at least .95 (182 vs. 156).

Can IR-ITML reduce the number of validation experiments to find one interaction compared to IR-Eucl?

We saw previously that IR-ITML outperforms IR-EUCL by a statistically signifcant margin. This motivates us to quantify the extent to which IR-ITML can reduce the number of validation experiments necessary to find one interaction compared to IR-EUCL.

In order to evaluate IR-ITML for this task, we first standardized our ML-compatible test collections according to the standardization procedure of Section 3.3.2. This resulted in 359 (a decrease of 3) standardized ML-compatible

test collections. We then applied IR-ITML and IR-EUCL to each test collection. As in Section 3.3.2, we evaluated each ranking according to the rank of the highest ranked relevant document. The results for IR-EUCL are shown in Table 3.5 and the results of IR-ITML are shown in Table 3.6.

Total # of interactors	# of rankings	# of 1-expt successes	Expected # of expts	Mean # of expts	Mean reduction in # of expts	# of expt-reducing rankings
1	49	5 (10.2%)	25.5	17.5	8 (31.4%)	**33** (67.4%)
2	35	5 (14.3%)	17	14.9	2.1 (12.4%)	22 (62.9%)
3	35	**11** (31.4%)	12.75	**6.97**	5.78 (45.3%)	**29** (82.85%)
4	24	9 (37.5%)	10.2	5.63	4.57 (44.8%)	20 (83.3%)
5	216	98 (45.4%)	8.5	4.75	3.75 (44.1%)	177 (81.9%)

Table 3.5: IR-EUCL applied to standardized ML-compatible test collections. If a cell's value is in bold, then IR-EUCL performs better than IR-ITML. See Table 3.3 for definitions of the columns.

Total # of interactors	# of rankings	# of 1-expt successes	Expected # of expts	Mean # of expts	Mean reduction in # of expts	# of expt-reducing rankings
1	49	**6** (12.2%)	25.5	**16.1**	9.4 (36.9%)	32 (65.3%)
2	35	**6** (17.1%)	17	**9.71**	7.29 (42.9%)	**28** (80%)
3	35	8 (22.9%)	12.75	7.77	4.98 (39.1%)	**29** (82.9%)
4	24	**11** (45.8%)	10.2	**4.13**	6.07 (59.5%)	**22** (91.7%)
5	216	**107** (49.5%)	8.5	**3.77**	4.73 (55.6%)	**190** (88.0%)

Table 3.6: IR-ITML applied to ML-compatible, standardized test collections. If a cell's value is in bold, then IR-ITML performs better than IR-EUCL. See Table 3.3 for definitions of the columns.

The preponderance of bold cells in Table 3.6 compared to Table 3.5 provides evidence that IR-ITML outperforms IR-EUCL according to "number of single experiment successes", "mean number of experiments to find one interaction", and "number of experiment-reducing rankings".To see if this difference in performance is significant, we used a paired t-test to compare IR-EUCL to IR-ITML in terms of the mean number of validation experiments to find one interaction. This test was conducted over 359 pairs, where each pair consisted of the rank of the highest-ranked relevant document in the IR-EUCL ranking and the IR-ITML ranking of a test collection. On average IR-ITML results in 1.30 fewer experiments than IR-EUCL and this difference is statistically significant ($p = 9.02\text{e-}3$).

To summarize the extent to which IR-ITML outperforms IR-EUCL, we computed the average percent reduction in number of experiments for both

systems. We found that on average, IR-ITML reduces the number of validation experiments to find one interaction by 50.4%, compared to 39.4% for IR-Eucl. In other words, when performing validation experiments to find one interactions, IR-ITML can be expected to save 11% more experiments than IR-Eucl.

Since IR-ITML significantly outperforms IR-Eucl in terms of mAP and the mean number of experiments to find one interaction, we conclude that IR-ITML should be used over IR-Eucl whenever a scientist has access to the melting curves of five or more known interactors.

3.3.4 Some Comments on Interpreting our Results

In this section, we make some comments on interpreting our results. Firstly, ranking a prey set in order of increasing melting curve dissimilarity from the bait does not necessarily require a scientist to invest the resources to produce an original set of melting curves. In fact, each of our test collections is based on publicly available melting curves not derived from the corresponding pull-down assay's protein mixture. Thus, if a prey set has publicly available melting curves, the ranking produced by our technique can be viewed as "free" information.

However, Tan et al. [41] show that a protein's melting curve can depend on the protein mixture from which it was generated. For instance, they showed that the melting curves of the Origin Recognition Complex vary considerably depending on whether they are obtained from intact cells or lysate. Furthermore, [19] found that of 118,162 purported interactions in 293T cells, only 35,704 of them could be detected in HCT116 cells, indicating that many interactions only occur in certain protein mixtures. In light of these results, we expect our technique to perform best when the melting curves used are derived from the pull-down assay's protein mixutre. Thus, in cases where this is possible, the results presented in this chapter likely underestimate the capability of our technique.

Our last comment is that relevant documents being mislabelled as non-relevant leads to rankings getting lower-than-deserved evaluation scores. For

example, the ranking 1111100000 is perfect and achieves an AP of 1. However, if the highest and fourth-highest ranked documents are incorrectly considered non-relevant, then the ranking appears to be 011010000 and achieves an AP of .589.

Recall that our set of ground-truth interactions is incomplete. Since each relevance judgment in our test collections corresponds to a ground-truth interaction, many of our test collections are surely missing relevance judgments, leading to some relevant documents being mislabelled as non-relevant. This is therefore another reason why the results of this chapter likely underestimate the performance of our technique.

3.4 Conclusion

This chapter investigated whether ranking a prey set in order of increasing melting curve dissimilarity from the bait is useful for limiting the number of validation experiments necessary to find interactions. Section 3.1 introduced information retrieval and metric learning. Section 3.2 described how IR techniques and metric learning were applied to perform our empirical analysis.

The first result of Section 3.3 was that IR-Eucl significantly outperforms random ranking systems. We then concluded that, compared to performing validation experiments in random order, IR-Eucl reduces the number of validation experiments necessary to find interactions.

Section 3.3.1 investigated how often IR-Eucl can reduce the need for validation experiments. We found that for 75.5% of pull-down assays, the ranking generated by IR-Eucl has at least a 50% chance of reducing the need for validation experiments. Moreover, for 31.3% of pull-down assays, the ranking has at least a 95% chance of reducing the need for validation experiments.

Section 3.3.2 sought to quantify the number of validation experiments needed to find one interaction. We found that for prey sets of size 50 with five interactors, an IR-Eucl ranking results in a "single experiment successes" nearly 40% of the time. Additionally, for prey sets of size 50 with a single

interactor, on average 18.2 experiments would be required to find a single interaction, which is 7.3 fewer than would be expected if experiments were performed in random order. We summarized this section by noting that on average IR-Eucl reduces the number of validation experiments to find one interaction by 35.4%.

Section 3.3.3 considered the situation where a scientist has access to the melting curves of some proteins known to interact with the bait. To take advantage of this situation, we proposed IR-ITML and IR-ITML-all, two categories of learned-metric IR systems capable of learning a metric from training pairs formed from the melting curves of known interactors. When evaluated in terms of mAP, we found that IR-ITML significantly outperforms both IR-ITML-all and IR-Eucl. We also found that on average IR-ITML reduces the number of validation experiments necessary to find one interaction by 11% more than IR-Eucl.

In Section 3.3.4 we made some comments on interpreting our results. Firstly, since many melting curves are publicly available, our technique can potentially be used without having to generate an original set of melting curves. However, we expect our technique to perform best when the melting curves used are obtained from the pull-down assay's protein mixture. Since our empirical analysis used publicly available melting curves and our relevance judgments are based on an incomplete set of ground-truth interactions, we concluded that our empirical analysis likely underestimates the capability of our technique.

Based on the empirical analysis performed in Section 3.3, we conclude that our technique, namely ranking a prey set in order of increasing melting curve dissimilarity from the bait, is useful for limiting the number of validation experiments necessary to find interactions.

Chapter 4

How Melting Curve Dissimilarity can be Used to Detect Proteins that Differentially Interact

So far we have seen a number of limitations of using melting curve dissimilarity to determine protein interactions. For example, we saw that melting curves of distinct complexes may occupy the same region of melting curve space, as shown in Figure 2.10 (B) and Figure 2.11 (C). This prevents clustering algorithms from finding clusters that correspond to protein complexes. We also saw that despite TPCA, some interactions do not have similar melting curves. This can be see in Figure 3.2 and Table 3.3, which show that for 33.5% of prey sets containing a single interactor, the interactor is more dissimilar to the bait than most of the non-interactors. Below we give a biological reason for an interaction's melting curves being dissimilar.

It is not unusual for a protein to be involved in numerous interactions in multiple complexes. In fact, UniProt ID: Q13547 is found in 41 of the 558 CORUM complexes considered by [41]. Figure 4.1 (A) depicts a protein mixture consisting of equal amounts of a 4-protein complex and a 5-protein complex. Observe that the purple protein occurs in both complexes. When heated, half of the purple proteins will denature as part of the 4-protein complex and the other half will denature as part of the 5-protein complex. This leads to the purple protein having a melting curve that is between the TPCA profiles of

the two complexes, as shown in Figure 4.1 (B). Therefore, any interaction involving the purple protein and a protein from one of the two complexes will consist of two dissimilar melting curves.

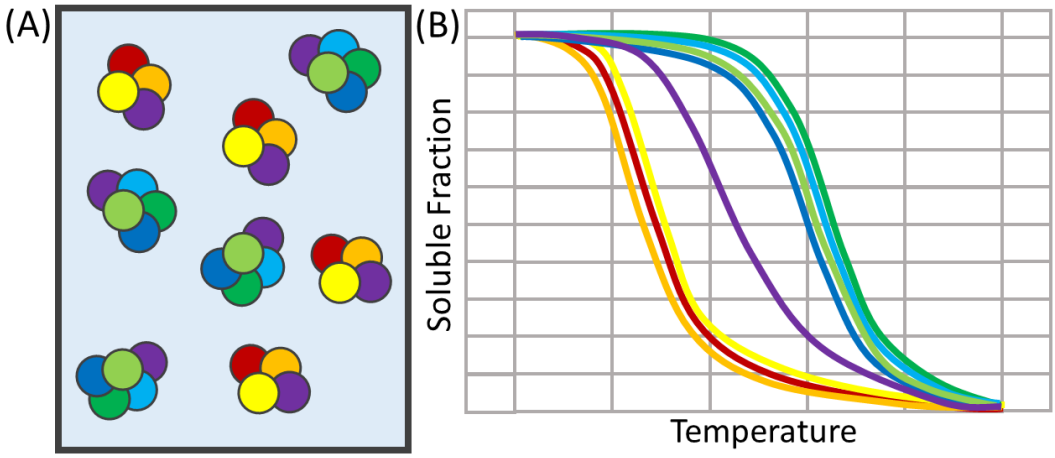

Figure 4.1: (A) A protein mixture consisting of equal amounts of a 4-protein complex and a 5-protein complex. The purple protein occurs in both complexes. (B) The melting curves of the protein mixture.

Due to the limitations of using melting curve dissimilarity to determine protein interactions, it is unlikely that the empirical results of Chapters 2 and 3 can be meaningfully improved by computational tools that we have not considered.

In this chapter, we describe how melting curve dissimilarity can be used to detect proteins that differentially interact. A differential interaction occurs between two proteins if they interact in one condition but do not interact in another condition. According to TPCA, there is evidence of a differential interaction if the melting curves of two proteins are similar in one condition but dissimilar in another condition. Therefore, when melting curves are obtained in two conditions, they can potentially be used to detect proteins that differentially interact.

Section 4.1 describes a previously published differential interaction measure F, and how it can be used to detect proteins that differentially interact. Section 4.2 identifies a problem with F through an illustrative example of melting curves obtained in two conditions. Motivated by the problem identified in Section 4.2, Section 4.3 proposes an improved differential interaction

measure.

4.1 Detecting Proteins that Differentially Interact

Given two proteins a and b with corresponding melting curves obtained in two conditions (denoted 0 and 1), Kurzawa et al. [22] proposed the following differential interaction measure:

$$F(a,b) = \frac{|d_0(a,b) - d_1(a,b)|}{\min\left(d_0(a,b), d_1(a,b)\right)}, \tag{4.1}$$

where $d_0(a,b)$ and $d_1(a,b)$ are the melting curve dissimilarities of a and b in conditions 0 and 1. We use the variable F because that is what is used in [22]. Suppose that $d_0(a,b) < d_1(a,b)$. Then $F(a,b)$ increases as $d_0(a,b)$ decreases and $d_1(a,b)$ increases. Therefore, $F(a,b)$ increases as the melting curve evidence of a differential interaction between a and b grows stronger.

To detect proteins that differentially interact, Kurzawa et al. proposed forming an empirical null distribution for F. This can be done by computing F for a large number (e.g. 10,000) of randomly selected protein pairs. An empirical p-value for two proteins a and b can be calculated as the fraction of values in the null distribution that are at least as high as $F(a,b)$. Two proteins demonstrate melting curve evidence of a differential interaction if their p-value is below a certain threshold (e.g. .05).

Forming an empirical null distribution for F requires a large number of melting curves. In fact, a distribution of size 10,000 requires the melting curves of at least 142 proteins in two conditions, as $\binom{141}{2} = 9870 < 10,000$. Although mass spectrometry can be used to generate thousands of melting curves [41], less expensive protein detection methods may only produce melting curves for a targeted class of proteins. If there are not enough melting curves to generate a sufficiently large null distribution for F, we propose ranking protein pairs in order of decreasing differential interaction scores (from highest F score to lowest F score). The higher a protein pair's rank, the more melting curve evidence there is that the pair differentially interacts. Ranking is particularly

useful if a scientist is interested in obtaining the k protein pairs that are most likely to differentially interact.

4.2 A Problem with F

Suppose the melting curve of a protein a has been obtained in two conditions (0 and 1). If a does not differentially interact, then a is part of the same set of interactions in both conditions. Therefore, a's condition 0 and 1 melting curves will be approximately the same, with any dissimilarity attributable to measurement error. However, if a differentially interacts with a protein b, then in one condition, a will denature as part of a complex formed with b, but in the other condition, a will denature independently of b. Therefore, a's condition 0 and 1 melting curves may be quite dissimilar. Thus, the greater the dissimilarity between a protein's condition 0 and 1 melting curves, the more likely the protein is to be involved in a differential interaction.

Figure 4.2 shows an illustrative example of three proteins' melting curves obtained in two conditions. Observe that the purple protein's condition 0 and 1 melting curves are approximately the same. In contrast, the blue protein's melting curves are quite dissimilar and the green protein's melting curves are even more dissimilar than those of the blue protein. Table 4.1 provides the soluble fraction values of each melting curve. It also contains the dissimilarity of each protein's condition 0 and 1 melting curves as measured by Euclidean distance.

When Euclidean distance is used as the dissimilarity measure, $F(\text{green}, \text{purple}) = 10.8 > F(\text{green}, \text{blue}) = 5.07$, implying that the green and purple proteins are more likely to differentially interact than the green and blue proteins. This is because the difference in Euclidean distance of the green and purple melting curves is greater than the difference in Euclidean distance of the green and blue melting curves.

However, the purple protein's condition 0 and 1 melting curves are approximately the same, as their Euclidean distance is only .0609 according to Table 4.1. Therefore, the purple protein's melting curves provide no evidence

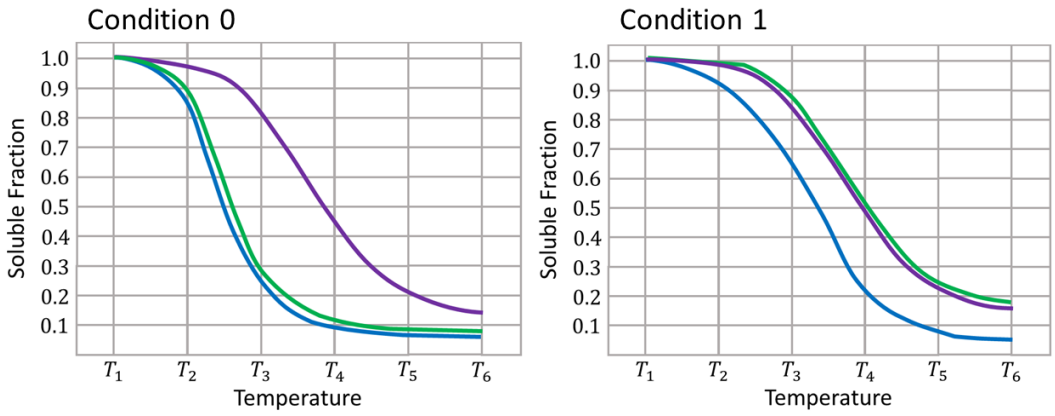

Figure 4.2: Illustrative example of three melting curves obtained in two conditions.

that the purple protein is involved in a differential interaction. In contrast, the green and blue proteins' condition 0 and 1 melting curves have high dissimilarities of .739 and .421, respectively, as measured by Euclidean distance. This is evidence that the green and blue proteins are involved in differential interactions. Therefore, contrary to the differential interaction scores produced by F, we contend that the green and blue proteins are more likely to differentially interact than the green and purple proteins. In the next section, we produce a differential interaction score that is consistent with this assertion.

Protein	Condition 0						Condition 1						Euclidean
	T_1	T_2	T_3	T_4	T_5	T_6	T_1	T_2	T_3	T_4	T_5	T_6	Distance
Green	1.00	.90	.29	.12	.09	.08	1.00	1.00	.88	.51	.25	.18	.739
Blue	1.00	.85	.25	.10	.07	.06	1.00	.92	.65	.21	.08	.05	.421
Purple	1.00	.97	.81	.45	.21	.14	1.00	.99	.84	.49	.23	.16	.0609

Table 4.1: Soluble fraction values for the melting curves of Figure 4.2. The right-most column is the Euclidean distance between a protein's condition 0 and 1 melting curves.

4.3 An Improved Differential Interaction Measure

In this section, we propose an improved differential interaction measure that takes into account whether both proteins' condition 0 and 1 melting curves show evidence of a differential interaction. Given two proteins a and b with

corresponding condition 0 and 1 melting curves, the improved differential interaction measure is defined as follows:

$$\Delta(a, b) = F(a, b) \cdot \min(d_{01}(a), d_{01}(b)), \tag{4.2}$$

where $d_{01}(a)$ and $d_{01}(b)$ are the dissimilarities of the condition 0 and 1 melting curves for a and b.

Table 4.2 shows $\Delta(\text{green}, \text{purple})$ and $\Delta(\text{green}, \text{blue})$. $\Delta(\text{green}, \text{purple})$ is computed by multiplying $F(\text{green}, \text{purple}) = 10.7$ by $d_{01}(\text{purple}) = .0609$, where $d_{01}(\text{purple})$ is obtained from Table 4.1. Likewise, $\Delta(\text{green}, \text{blue})$ is computed by multiplying $F(\text{green}, \text{blue}) = 5.07$ by $d_{01}(\text{blue}) = .421$, where $d_{01}(\text{blue})$ is obtained from Table 4.1. Observe that $\Delta(\text{green}, \text{blue}) > \Delta(\text{green}, \text{purple})$, as the purple protein's condition 0 and 1 melting curves are approximately the same in both conditions, causing $d_{01}(\text{purple})$ to be nearly 0, but the blue protein's condition 0 and 1 melting curves are quite dissimilar in the two conditions, causing $d_{01}(\text{blue})$ to be nearly seven times higher than $d_{01}(\text{purple})$. Unlike F, Δ is consistent with our assertion that the green and blue proteins are more likely to differentially interact than the green and purple proteins.

Protein a	Protein b	$F(a, b)$	$\Delta(a, b)$
Green	Purple	**10.8**	.658
Green	Blue	5.07	**2.13**

Table 4.2: Differential interaction scores according to F and Δ.

4.4 Conclusion

In this chapter, we described how melting curve dissimilarity can be used to detect proteins that differentially interact. Section 4.1 introduced the differential interaction score of Kurzawa et al., known as F, and two ways that F can be used to detect proteins that differentially interact. The first way is to form an empirical null distribution for F, and use it to compute p-values for protein pairs. If a protein pair has a p-value below a certain threshold, then it demonstrates melting curve evidence of a differential interaction. The second way is to rank protein pairs in order of decreasing differential interaction scores. The

higher a protein pair's rank, the more melting curve evidence there is that the pair differentially interacts.

Section 4.2 began by noting that the greater the dissimilarity between a protein's condition 0 and 1 melting curves, the more likely the protein is to be involved in a differential interaction. We then computed F for two protein pairs, one pair containing a protein with condition 0 and 1 melting curves that are approximately the same, and another pair made up of two proteins that both have highly dissimilar condition 0 and 1 melting curves. Despite our observation, we found the F score of the former protein pair to be greater than that of the latter pair, implying that the former pair is the most likely to differentially interact. Therefore, Section 4.3 proposed Δ, an improved differential interaction measure that takes into account whether both proteins' condition 0 and 1 melting curves show evidence of being involved in a differential interaction. In contrast to F, Δ gave the latter pair the highest differential score, implying that the latter pair is the most likely to differentially interact.

Chapter 5

Conclusion

Although Tan et al. proposed TPCA and provided empirical evidence that it exists, they did not attempt to determine interactions by applying computational tools to melting curve datasets. In this thesis, we applied computational tools to melting curve datasets, and described the opportunities and limitations of such tools for determining protein interactions.

Chapter 2 asked the following question: can melting curve datasets provide useful information about protein complexes? To answer this question, we generated 529 melting curve datasets, each containing the melting curves of three complexes. Although we found that clustering a melting curve dataset provides a statistically significant level of information about the dataset's complex partition, in general the clustering partition is not close enough to the complex partition to be practically useful.

We then turned our attention to "informative" partitions, which we defined as clustering partitions that provide a meaningful level of information about their respective complex partitions. Although informative partitions are rarer than non-informative partitions, we noted that clustering melting curve datasets could still be useful if informative partitions are identifiable by internal cluster validation (ICV) measures. However, we found this not to be the case, as even among clustering partitions with the highest ICV scores, non-informative partitions are still more common than informative partitions. Since informative partitions are relatively rare and cannot be identified by ICV scores with any reasonable degree of accuracy, we concluded that cluster-

ing melting curve datasets does not provide useful information about protein complexes.

Chapter 3 asked the following question: are melting curves useful for limiting the number of validation experiments necessary to find protein interactions? We answered this question in the context of pull-down assays. Recall that given a bait protein, a pull-down assay generates a prey set consisting of proteins that potentially interact with the bait. To determine interactions from the results of a pull-down assay, validation experiments must be performed, each involving the bait and a prey protein. In Chapter 3, we investigated whether ranking a prey set in order of increasing melting curve dissimilarity from the bait is useful for limiting the number of validation experiments necessary to find interactions.

We first determined that ranking prey proteins in order of increasing Euclidean distance from the bait (i.e. using IR-Eucl) results in a statistically significant reduction in the number of validation experiments necessary to find interactions. Since this result does not imply that IR-Eucl always limits the number of required validation experiments, or even frequently does so, we computed the probability that IR-Eucl reduces the need for validation experiments for 1414 prey sets. We found that for 75.5% of prey sets, IR-Eucl has at least a .5 probability of reducing the need for validation experiments, and for 31.3% of prey sets, this probability is at least .95. We then quantified the number of validation experiments needed to find one interaction if IR-Eucl is used to rank the prey set, and found that IR-Eucl reduces the need for validation experiments by 35.4%.

Next, we considered the the situation where a scientist has access to the melting curves of some proteins that are known to interact with the bait, and described how metric learning can be used to learn a metric that takes advantage of the information in these melting curves. We established that ranking a prey set based on a learned metric outperforms IR-Eucl by a statistically significant margin. Furthermore, it reduces the number of validation experiments necessary to find one interaction by an additional 11%.

Based on these empirical results, we concluded that melting curves are

indeed useful for limiting the number of validation experiments necessary to find protein interactions.

Lastly, Chapter 4 described how melting curve dissimilarity can be used to detect proteins that differentially interact. We began by introducing the differential interaction measure of Kurzawa et al., known as F, and two ways that F can be used to detect proteins that differentially interact. We then noted that the greater the dissimilarity between a protein's melting curves in two conditions, the more likely the protein is to be involved in a differential interaction. Since F does not consider the dissimilarity of a protein's melting curves in two conditions, we introduced a new differential interaction measure Δ that does take this into account. On an illustrative example of melting curves obtained in two conditions, we showed that Δ is an improvement over F.

References

[1] A. Bellet, A. Habrard, and M. Sebban, "Metric learning," *Synthesis lectures on artificial intelligence and machine learning*, vol. 9, no. 1, pp. 1–151, 2015.

[2] T. Berggård, S. Linse, and P. James, "Methods for the detection and analysis of protein–protein interactions," *Proteomics*, vol. 7, no. 16, pp. 2833–2842, 2007.

[3] R. J. Campello, D. Moulavi, A. Zimek, and J. Sander, "A framework for semi-supervised and unsupervised optimal extraction of clusters from hierarchies," *Data Mining and Knowledge Discovery*, vol. 27, no. 3, pp. 344–371, 2013.

[4] ——, "Hierarchical density estimates for data clustering, visualization, and outlier detection," *ACM Transactions on Knowledge Discovery from Data (TKDD)*, vol. 10, no. 1, pp. 1–51, 2015.

[5] G. Celeux and G. Govaert, "A classification em algorithm for clustering and two stochastic versions," *Computational statistics & Data analysis*, vol. 14, no. 3, pp. 315–332, 1992.

[6] G. Celeux and G. Govaert, "Gaussian parsimonious clustering models," *Pattern recognition*, vol. 28, no. 5, pp. 781–793, 1995.

[7] H. N. Chua, W.-K. Sung, and L. Wong, "Exploiting indirect neighbours and topological weight to predict protein function from protein–protein interactions," *Bioinformatics*, vol. 22, no. 13, pp. 1623–1630, 2006.

[8] U. Consortium, "Uniprot: A worldwide hub of protein knowledge," *Nucleic acids research*, vol. 47, no. D1, pp. D506–D515, 2019.

[9] J. Davis and I. Dhillon, "Differential entropic clustering of multivariate gaussians," *Advances in Neural Information Processing Systems*, vol. 19, 2006.

[10] J. V. Davis, B. Kulis, P. Jain, S. Sra, and I. S. Dhillon, "Information-theoretic metric learning," in *Proceedings of the 24th international conference on Machine learning*, 2007, pp. 209–216.

[11] W. de Vazelhes, C. Carey, Y. Tang, N. Vauquier, and A. Bellet, "Metric-learn: Metric Learning Algorithms in Python," *Journal of Machine Learning Research*, vol. 21, no. 138, pp. 1–6, 2020.

[12] A. P. Dempster, N. M. Laird, and D. B. Rubin, "Maximum likelihood from incomplete data via the em algorithm," *Journal of the Royal Statistical Society: Series B (Methodological)*, vol. 39, no. 1, pp. 1–22, 1977.

[13] B. Everitt, S. Landau, M. Leese, and D. Stahl, *Cluster Analysis*, ser. Wiley Series in Probability and Statistics. Wiley, 2011.

[14] C. Fraley, "Algorithms for model-based gaussian hierarchical clustering," *SIAM Journal on Scientific Computing*, vol. 20, no. 1, pp. 270–281, 1998.

[15] A.-C. Gavin, P. Aloy, P. Grandi, R. Krause, M. Boesche, M. Marzioch, C. Rau, L. J. Jensen, S. Bastuck, B. Dümpelfeld, *et al.*, "Proteome survey reveals modularity of the yeast cell machinery," *Nature*, vol. 440, no. 7084, pp. 631–636, 2006.

[16] M. Giurgiu, J. Reinhard, B. Brauner, I. Dunger-Kaltenbach, G. Fobo, G. Frishman, C. Montrone, and A. Ruepp, "Corum: The comprehensive resource of mammalian protein complexes—2019," *Nucleic acids research*, vol. 47, no. D1, pp. D559–D563, 2019.

[17] M. Hahsler, M. Piekenbrock, and D. Doran, "Dbscan: Fast density-based clustering with r," *Journal of Statistical Software*, vol. 91, no. 1, pp. 1–30, 2019.

[18] L. Hubert and P. Arabie, "Comparing partitions," *Journal of classification*, vol. 2, no. 1, pp. 193–218, 1985.

[19] E. L. Huttlin, R. J. Bruckner, J. Navarrete-Perea, J. R. Cannon, K. Baltier, F. Gebreab, M. P. Gygi, A. Thornock, G. Zarraga, S. Tam, *et al.*, "Dual proteome-scale networks reveal cell-specific remodeling of the human interactome," *Cell*, vol. 184, no. 11, pp. 3022–3040, 2021.

[20] L. Kaufman and P. J. Rousseeuw, *Finding groups in data: an introduction to cluster analysis*. John Wiley & Sons, 2009, vol. 344.

[21] S. Kullback and R. A. Leibler, "On information and sufficiency," *The annals of mathematical statistics*, vol. 22, no. 1, pp. 79–86, 1951.

[22] N. Kurzawa, A. Mateus, and M. M. Savitski, "Rtpca: An r package for differential thermal proximity coaggregation analysis," *Bioinformatics*, vol. 37, no. 3, pp. 431–433, 2021.

[23] A. Louche, S. P. Salcedo, and S. Bigot, "Protein–protein interactions: Pull-down assays," in *Bacterial Protein Secretion Systems*, Springer, 2017, pp. 247–255.

[24] C. B. Lozzio and B. B. Lozzio, "Human chronic myelogenous leukemia cell-line with positive philadelphia chromosome," 1975.

[25] M. Mahajan, P. Nimbhorkar, and K. Varadarajan, "The planar k-means problem is np-hard," in *International Workshop on Algorithms and Computation*, Springer, 2009, pp. 274–285.

[26] H. B. Mann and D. R. Whitney, "On a test of whether one of two random variables is stochastically larger than the other," *The annals of mathematical statistics*, pp. 50–60, 1947.

[27] L. McInnes, J. Healy, and S. Astels. (). "How hdbscan works," [Online]. Available: `https://hdbscan.readthedocs.io/en/latest/how_hdbscan_works.html`. (accessed: 09.15.2021).

[28] D. Moulavi, P. A. Jaskowiak, R. J. Campello, A. Zimek, and J. Sander, "Density-based clustering validation," in *Proceedings of the 2014 SIAM international conference on data mining*, SIAM, 2014, pp. 839–847.

[29] E. A. Ponomarenko, E. V. Poverennaya, E. V. Ilgisonis, M. A. Pyatnitskiy, A. T. Kopylov, V. G. Zgoda, A. V. Lisitsa, and A. I. Archakov, "The size of the human proteome: The width and depth," *International journal of analytical chemistry*, vol. 2016, 2016.

[30] G.-J. Qi, J. Tang, Z.-J. Zha, T.-S. Chua, and H.-J. Zhang, "An efficient sparse metric learning in high-dimensional space via l 1-penalized log-determinant regularization," in *Proceedings of the 26th Annual International Conference on Machine Learning*, 2009, pp. 841–848.

[31] W. M. Rand, "Objective criteria for the evaluation of clustering methods," *Journal of the American Statistical association*, vol. 66, no. 336, pp. 846–850, 1971.

[32] C. Ritz, F. Baty, J. C. Streibig, and D. Gerhard, "Dose-response analysis using r," *PloS one*, vol. 10, no. 12, e0146021, 2015.

[33] T. Rolland, M. Taşan, B. Charloteaux, S. J. Pevzner, Q. Zhong, N. Sahni, S. Yi, I. Lemmens, C. Fontanillo, R. Mosca, *et al.*, "A proteome-scale map of the human interactome network," *Cell*, vol. 159, no. 5, pp. 1212–1226, 2014.

[34] M. Sanderson, *Test collection based evaluation of information retrieval systems*. Now Publishers Inc, 2010.

[35] G. Schwarz, "Estimating the dimension of a model," *The annals of statistics*, pp. 461–464, 1978.

[36] L. Scrucca, M. Fop, T. B. Murphy, and A. E. Raftery, "mclust 5: Clustering, classification and density estimation using Gaussian finite mixture models," *The R Journal*, vol. 8, no. 1, pp. 289–317, 2016. [Online]. Available: `https://doi.org/10.32614/RJ-2016-021`.

[37] E. Sjöstedt, W. Zhong, L. Fagerberg, M. Karlsson, N. Mitsios, C. Adori, P. Oksvold, F. Edfors, A. Limiszewska, F. Hikmet, *et al.*, "An atlas of the protein-coding genes in the human, pig, and mouse brain," *Science*, vol. 367, no. 6482, eaay5947, 2020.

[38] M. D. Smucker, J. Allan, and B. Carterette, "A comparison of statistical significance tests for information retrieval evaluation," in *Proceedings of the sixteenth ACM conference on Conference on information and knowledge management*, 2007, pp. 623–632.

[39] Springer Verlag GmbH, European Mathematical Society, *Negative hypergeometric distribution*, Website, URL: `https://encyclopediaofmath.org/wiki/Negative_hypergeometric_distribution`. Accessed on 2022-04-06.

[40] M. P. Stumpf, T. Thorne, E. De Silva, R. Stewart, H. J. An, M. Lappe, and C. Wiuf, "Estimating the size of the human interactome," *Proceedings of the National Academy of Sciences*, vol. 105, no. 19, pp. 6959–6964, 2008.

[41] C. S. H. Tan, K. D. Go, X. Bisteau, L. Dai, C. H. Yong, N. Prabhu, M. B. Ozturk, Y. T. Lim, L. Sreekumar, J. Lengqvist, *et al.*, "Thermal proximity coaggregation for system-wide profiling of protein complex dynamics in cells," *Science*, vol. 359, no. 6380, pp. 1170–1177, 2018.

[42] E. Xing, M. Jordan, S. J. Russell, and A. Ng, "Distance metric learning with application to clustering with side-information," *Advances in neural information processing systems*, vol. 15, 2002.

[43] J. Zahiri, A. Emamjomeh, S. Bagheri, A. Ivazeh, G. Mahdevar, H. S. Tehrani, M. Mirzaie, B. A. Fakheri, and M. Mohammad-Noori, "Protein complex prediction: A survey," *Genomics*, vol. 112, no. 1, pp. 174–183, 2020.

[44] J. Zahiri, J. Hannon Bozorgmehr, and A. Masoudi-Nejad, "Computational prediction of protein–protein interaction networks: Algorithms and resources," *Current genomics*, vol. 14, no. 6, pp. 397–414, 2013.