Frank-Wolfe Optimization for Dominant Set Clustering

Master's thesis in Computer Science and Engineering

CARL JOHNELL

Department of Computer Science and Engineering
CHALMERS UNIVERSITY OF TECHNOLOGY
UNIVERSITY OF GOTHENBURG
Gothenburg, Sweden 2020

MASTER'S THESIS 2020

Frank-Wolfe Optimization for Dominant Set Clustering

CARL JOHNELL

UNIVERSITY OF GOTHENBURG

CHALMERS
UNIVERSITY OF TECHNOLOGY

Department of Computer Science and Engineering
CHALMERS UNIVERSITY OF TECHNOLOGY
UNIVERSITY OF GOTHENBURG
Gothenburg, Sweden 2020

Frank-Wolfe Optimization for Dominant Set Clustering
CARL JOHNELL

Supervisor: Morteza Haghir Chehreghani, Department of Computer Science and Engineering
Examiner: Peter Damaschke, Department of Computer Science and Engineering

Master's Thesis 2020
Department of Computer Science and Engineering
Chalmers University of Technology and University of Gothenburg
SE-412 96 Gothenburg
Telephone +46 31 772 1000

Typeset in LaTeX
Gothenburg, Sweden 2020

Frank-Wolfe Optimization for Dominant Set Clustering
CARL JOHNELL
Department of Computer Science and Engineering
Chalmers University of Technology and University of Gothenburg

Abstract

It is convenient to represent a clustering problem as an edge-weighted graph, where the nodes and weights represent the objects and pairwise similarities of the problem. Dominant set is a graph-theoretical definition that extends the idea of maximal clique from unweighted to weighted graphs – intuitively it is a group of connected nodes with relatively large weights. Dominant Set Clustering is based on extracting groups of nodes (clusters) from the graph that satisfy the definition of dominant set. The clusters are computed by solving standard quadratic optimization problems (StQPs) and utilizing a correspondence between the solutions and graph-theoretical definition. In this thesis we study the StQP from the perspective of the Frank-Wolfe (FW) algorithm, and variants thereof, and relate them to replicator dynamics – a method commonly used for computing dominant sets. We consider standard FW, pairwise FW, and away-steps FW, and conclude that all variants perform similarly and are much more efficient compared to replicator dynamics. Explicit proofs of the convergence rate $\mathcal{O}(1/\sqrt{t})$, in terms of the so called Frank-Wolfe gap, are also included for the FW variants when applied to the StQP.

Keywords: clustering, convergence rate, dominant set, frank-wolfe, optimization, replicator dynamics, stqp.

Acknowledgements

I would like to thank my supervisor Morteza Haghir Chehreghani for the guidance and suggestions that helped shape this thesis. Thank you to Peter Damaschke for acting as the examiner. Finally, I want to acknowledge Chalmers University of Technology and the Department of Computer Science and Engineering for providing an excellent education during the course of my master's degree.

Carl Johnell, Gothenburg, June 2020

Contents

List of Figures

List of Tables

1

Introduction

The vast amount of data generated by, for example, the Internet, cameras, and other sensors, require algorithms that can automatically process and organize the data to make it more accessible [1]. Data clustering is one approach to these kind of problems – it is the task of organizing a set of objects into groups, based on similarity measures, such that the groups have high intra-cluster and low inter-cluster similarity. In other words, objects within a group should be highly similar relative to the objects in other groups.

Data clustering has been applied to numerous applications and domains. Some examples include image segmentation in computer vision, organizing documents into topics/categories for easier access, grouping customers into different types for marketing, and studying genome data in biology [1]. Generally speaking, clustering is used for a) exploratory data analysis (gaining insight into the data), b) classification (e.g. image segmentation or organizing documents), and c) summarizing data through cluster representatives [1].

1.1 Approaches to Clustering

The objects in a clustering problem can be represented by vectors in a feature space[1], where the similarity measure is commonly based on the Euclidean distance between a pair of vectors. Clustering is then based on finding dense regions in this feature space [1]. A probabilistic approach assumes the data was generated by a mixture distribution, and the clusters then correspond to the components in the mixture model [1]. Nonparametric methods directly search for dense connected regions, for some definition of connected depending on the specific algorithm [1].

An alternative to a global feature representation is to define object relations in terms of local pairwise similarities; in many real-world applications, this is often easier to obtain, or more efficient for learning, compared to the feature-based approach [2]. In this context, it is convenient to represent the clustering problem as an edge-weighted graph, where objects are nodes and edges the similarity measures. Algorithms based on the minimum cut[2] naturally applies to this type of clustering.

[1]The object features are embedded as vectors in a normed vector space.

[2]A minimum cut separates a graph into two disjoint subgraphs such that the total weight of the edges removed is minimized.

Furthermore, clustering algorithms can be categorized as *hierarchical* or *partitional*. The former generates a tree of nested clusters, while the latter partitions the objects into disjoint subsets [1].

1.2 Problem

The focus of this thesis is the Dominant Set Clustering (DSC) framework [2, 3] and variants of the Frank-Wolfe (FW) algorithm [4] applied to it. DSC is a general framework with links to graph, optimization, and game theory. It is based on pairwise similarities and can be used for both partitional [2] and hierarchical [5] clustering. The formulation of DSC most relevant to this thesis is that of the optimization of a quadratic form constrained to the simplex, called standard quadratic problem (StQP). Frank-Wolfe is an iterative method for solving nonlinear constrained optimization problems, which will be studied in the context of the StQP defined by DSC. We will consider three variants of Frank-Wolfe: standard, pairwise, and away-steps – denoted by FW, PFW, and AFW, respectively.

The StQP defined by DSC has traditionally been solved by the method of replicator dynamics (RD), a class of continuous- and discrete-time dynamical systems from evolutionary game theory [2]. However, RD has quadratic per-iteration time complexity, and as explained in [6], it is sensitive to a number of parameters and can therefore result in widely different solutions, depending on their configuration. The infection and immunization dynamics (InImDyn) was proposed in [7] as an alternative to RD for StQP, with linear per-iteration time complexity. It turns out InImDyn is equivalent to AFW for StQP, however, its derivation is based on evolutionary game theory as opposed to the Frank-Wolfe method. It is therefore interesting to investigate the same problem from the Frank-Wolfe perspective, and also evaluate RD, standard FW, PFW, and AFW/InImDyn experimentally.

1.3 Contributions

The contributions of the thesis are two-fold. Chapter 4 provides derivations and convergence rate analysis for the Frank-Wolfe variants when applied to the StQP. While the convergence rates are not novel results, their proofs have been adapted to the StQP and are therefore more easily accessible compared to their general counterparts. The second contribution is the experimental evaluation of the Frank-Wolfe variants for Dominant Set Clustering, relative to replicator dynamics.

2

Notation

The notation is mostly standard. Bold lowercase and uppercase denote vectors and matrices, respectively; normal lowercase are scalars and normal uppercase are scalars/sets. Component i of vector $\mathbf{x}$ is denoted x_i; element at row i and column j of matrix $\mathbf{A}$ is a_{ij}. The i-th row of $\mathbf{A}$ is $\mathbf{a}_{i*}$ and the j-th column of $\mathbf{A}$ is $\mathbf{a}_{*j}$. Vectors are in column order; the inner product of vectors $\mathbf{x}$ and $\mathbf{y}$ is $\mathbf{x}^T\mathbf{y}$, where T is the transpose. Vectors are n-dimensional unless otherwise stated. The t-th scalar or vector in a sequence is indexed by t as subscript, e.g. γ_t or $\mathbf{d}_t$. If a sequence also needs to be indexed by a component, then the sequence index is superscripted in parenthesis and the component index is subscripted, e.g. $x_i^{(t)}$. A named scalar, vector, or matrix has uppercase letters in the superscript, e.g. g^{FW}, $\mathbf{d}^{FW}$, or $\mathbf{D}^P$; note when T is in the superscript it is always the transpose. Sets have the name in the subscript, e.g. C_{GT}. Unless stated otherwise, $||\cdot||$ denotes the L2-norm.

Table 2.1: Frequently occurring notation.

	General
$\mathbf{I}$	Identity matrix.
$\mathbf{e}_i$	i-th column of the identity matrix.
$\mathbf{e}$	Vector of ones.
	Dominant Set Clustering
$\mathcal{G} = (V, E, w)$	Edge-weighted graph, where V nodes, E edges, and $w : E \to \mathbb{R}_+$ positive weight function.
$V = \{1, ..., n\}$	Set of nodes.
$E \subseteq V \times V$	Set of edges.
$\mathbf{A}$	Similarity matrix. Definition 3.
Δ	Simplex. Definition 3.
$\sigma(\mathbf{x})$	Support; $\mathbf{x}$ is omitted when it is clear from context. Definition 4.
α	Regularization parameter. Section 3.1.3.
δ	Cutoff parameter. Section 5.1.
	Frank-Wolfe
g	Frank-Wolfe gap. Definition 6.
γ	Step size. Equation (3.12).
$\mathbf{d}$	Ascent direction. Equation (3.11).
$\mathbf{s}$	Frank-Wolfe vertex. Equation (3.9).
$\mathbf{v}$	Away vertex. Equation (3.10).

3

Background

This chapter presents an overview of the theory for Dominant Set Clustering, replicator dynamics, and Frank-Wolfe. The most closely related work is summarized at the end of the chapter.

3.1 Dominant Set Clustering

Generally speaking, clustering in this context is achieved as a byproduct from finding subsets of nodes that satisfy the definition of a *dominant set*. It is a graph-theoretical concept that formalizes the two key properties a cluster should satisfy: relatively high and low intra-cluster and inter-cluster similarity, respectively. The definition of dominant set is a generalization of maximal clique[1] from unweighted to weighted graphs, and for unweighted graphs, the two concepts are equivalent. This is some motivation of why dominant set is an appropriate cluster definition, since maximal clique can be considered as the strictest definition of a cluster in unweighted graphs [2].

DSC has been shown to achieve good performance. Pavan and Pelillo [2] performed numerical experiments, using image segmentation problems and point datasets, and found that DSC generally outperformed, or performed on par with, K-means [1], DBSCAN [8], and Normalized Cut (NCut) [9] – three popular clustering algorithms.

This section introduces the graph-theoretical notion of dominant set and its link to optimization theory, as outlined in [2, 5], as well as clustering strategies based on these ideas.

3.1.1 Graph-Theoretical Definition

Let $\mathcal{G} = (V, E, w)$ be an edge-weighted graph, where $V = \{1, ..., n\}$ the set of nodes, $E \subseteq V \times V$ the set of edges, and $w : E \to \mathbb{R}_+$ the positive weight function. The graph $\mathcal{G}$ is represented by an adjacency (similarity) matrix $\mathbf{A} = (a_{ij})$, where $a_{ij} = w(i, j)$ if $(i, j) \in E$ and $a_{ij} = 0$ otherwise.

Let $S \subseteq V$ be a nonempty subset and $i \in S$. The weighted degree of i with respect

[1]A clique is a subset of nodes such that any pair of nodes are connected, i.e. they form a complete subgraph. A maximal clique is not contained in any larger clique.

to S is defined as

$$\operatorname{awdeg}_S(i) = \frac{1}{|S|} \sum_{j \in S} a_{ij},$$

and for $j \notin S$, define $\phi_S(i,j) = a_{ij} - \operatorname{awdeg}_S(i)$. Intuitively, $\phi_S(i,j)$ measures the similarity between nodes i and j, relative to the average similarity between node i and the nodes in S.

Definition 1. Let $S \subseteq V$ be a nonempty subset, $i \in S$, and $S_i^c = S \setminus \{i\}$. The weight of i with respect to S is defined recursively as

$$w_S(i) = \begin{cases} 1, & \text{if } |S| = 1 \\ \sum\limits_{j \in S_i^c} \phi_{S_i^c}(j,i) w_{S_i^c}(j), & \text{otherwise.} \end{cases} \tag{3.1}$$

Furthermore, the total weight of S is defined to be $W(S) = \sum\limits_{i \in S} w_S(i)$.

The weight $w_S(i)$ is a measure of the overall similarity between node i and the nodes $S \setminus \{i\}$, relative to the overall similarity among the nodes in $S \setminus \{i\}$. For example, we have $w_{\{1,2,3,4\}}(1) > 0$ and $w_{\{1,2,3,4\}}(1) < 0$ for Figures 3.1a and 3.1b, respectively. This matches our intuition that in Figure 3.1a the edge weights connected to node 1 are large compared to the edge weights between nodes $\{2,3,4\}$, while the opposite is the case in Figure 3.1b.

Definition 2. A nonempty subset $S \subseteq V$, such that $W(T) > 0$ for any nonempty subset $T \subseteq S$, is said to be dominant if

1. $w_S(i) > 0$ for all $i \in S$,
2. $w_{S \cup \{i\}}(i) < 0$ for all $i \notin S$.

The two conditions correspond to relatively high and low intra-cluster and inter-cluster similarity, respectively. Intuitively, one can think of condition 1 as being a clique, and condition 2 ensures it is maximal.

The nodes $\{1,2,3\}$ are a dominant set in Figure 3.1c, which can be understood by observing the intra-cluster edge weights are 60, 70 and 90, while the inter-cluster edges weights are in the range 5 - 25. That is, the main property of a dominant set is that the overall similarity among internal nodes in a cluster is higher than that between internal and external nodes. This is the motivation to consider dominant set as a cluster of nodes.

3.1.2 Link to Optimization Theory

Definition 3. Optimization problem (3.2) is referred to as the standard quadratic problem (StQP).

$$\begin{aligned} &\text{maximize } f(\mathbf{x}) = \mathbf{x}^T \mathbf{A} \mathbf{x} \\ &\text{subject to } \mathbf{x} \in \Delta = \left\{ \mathbf{x} \in \mathbb{R}^n : \mathbf{x} \geq \mathbf{0}^n \text{ and } \sum_{i=1}^{n} x_i = 1 \right\}. \end{aligned} \tag{3.2}$$

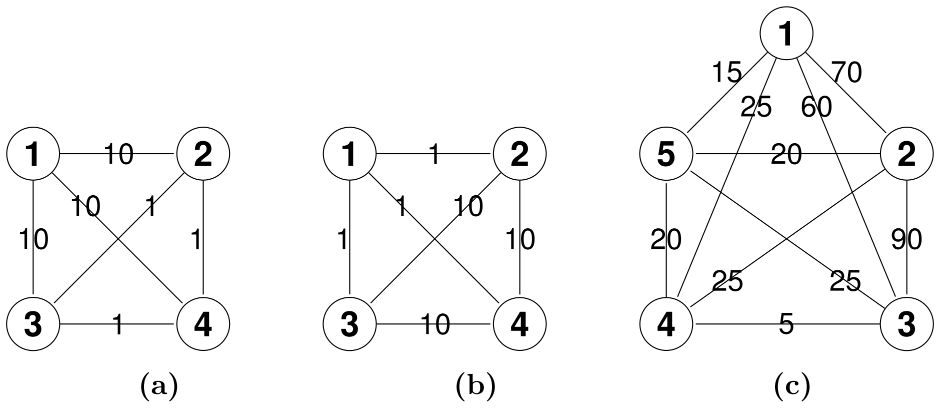


Figure 3.1: Edge-weighted graphs.

The constraint Δ is called the standard simplex. The matrix $\mathbf{A}$ is symmetric $\mathbf{A}^T = \mathbf{A}$, and we further restrict it to have nonnegative entries and zeros on the main diagonal. Note that $\mathbf{A}$ is generally indefinite and the problem is then nonconcave.

When $\mathbf{A}$ is the similarity matrix of an edge-weighted graph, a relationship between dominants sets and local solutions to the optimization problem is established in [2]. The key ideas are summarized here.

Let $V = \{1, ..., n\}$ be the set of nodes in the graph. The vector $\mathbf{x} \in \Delta$ is an n-dimensional vector that is associated to a cluster of nodes, where the components x_i, $i \in V$, represent the (weighted) participation in the cluster. Components with small and large values are weakly and strongly associated to the cluster, respectively. An intuitive interpretation of problem (3.2) then is that it maximizes the cohesiveness of the cluster – the intra-cluster and inter-cluster edges should have large and small edge weights, respectively.

Definition 4 (Support)**.** The support of $\mathbf{x} \in \Delta$ is defined as $\sigma(\mathbf{x}) = \{i \in V : x_i > 0\}$, the nonzero components of $\mathbf{x}$.

Definition 5 (Characteristic vector)**.** A nonempty subset $S \subseteq V$ admits weighted characteristic vector $\mathbf{x}^S$ if it has total weight $W(S) \neq 0$, in which case, we define

$$x_i^S = \begin{cases} \frac{w_S(i)}{W(S)}, & \text{if } i \in S \\ 0, & \text{otherwise.} \end{cases} \tag{3.3}$$

The nonzero components are the normalized node weights from Definition 1.

Theorem 1. *If $S \subseteq V$ is a dominant set, then its weighted characteristic vector $\mathbf{x}^S$ is a strict local solution of problem* (3.2)*. Conversely, if $\mathbf{x}^*$ is a strict local solution of problem* (3.2)*, then its support $\sigma = \sigma(\mathbf{x}^*)$ is a dominant set, provided that $w_{\sigma \cup \{i\}}(i) \neq 0$ for all $i \notin \sigma$.*

Theorem 1 thus establishes a correspondence between dominants sets and local solutions of StQP (3.2). In other words, by computing a local solution to the optimization problem we are also locating a dominant set of the graph corresponding to the similarity matrix.

3.1.3 Regularized Objective

Let $\mathbf{A}_\alpha = \mathbf{A} + \alpha(\mathbf{e}\mathbf{e}^T - \mathbf{I})$, where $\alpha \geq 0$ is called the regularization parameter. That is, α is added to all off-diagonal elements of $\mathbf{A}$. Let $f_\alpha(\mathbf{x}) = \mathbf{x}^T\mathbf{A}_\alpha\mathbf{x}$ and define optimization problem

$$\begin{aligned} &\text{maximize } f_\alpha(\mathbf{x}) \\ &\text{subject to } \mathbf{x} \in \Delta. \end{aligned} \tag{3.4}$$

For $\alpha = 0$, this is identical to (3.2).

In [5] it was shown that by increasing the value of α, local solutions to problem (3.2) with small support are no longer local solutions to problem (3.4). By virtue of Theorem 1, this means smaller dominant sets (clusters) are ignored. Specifically, for $\mathbf{A} = (a_{ij})$, $0 \leq a_{ij} \leq 1$ and $\alpha > m - 1$, clusters of size less than or equal to m are avoided. This is useful to avoid small, unwanted, clusters.

3.1.4 Clustering Strategies

A solution to problem (3.2) corresponds to a single cluster. If this is all that is needed, for example, separating a coherent region from noise, then it is sufficient to stop once a solution is obtained. For more complicated problems, different strategies are needed.

As briefly mentioned in the Chapter 1, clustering algorithms can generally be considered as *partitional* or *hiearchical* [1]. The former splits the data into disjoint subsets (clusters), while the latter creates a hiearchy of larger clusters at the top to smaller clusters at the bottom. Hierarchical algorithms either work bottom-up – called agglomerative mode – or top-down – called divisive mode. In bottom-up, each object starts as a cluster and then similar ones are successively merged to form the hierarchy. In top-down, they start with a single large cluster that is then successively divided into smaller clusters.

In [5] a hierarchical clustering algorithm for dominant sets is proposed using the regularized objective in the previous section. The method is top-down and starts with a large value for α and then iteratively decreases it. For each value of α, a partitioning is generated that forms a level in the hierarchy. It should be noted that the result is not necessarily a proper hierarchy, meaning a dominant set at one level might not be a subset of any dominant set extracted at a previous level.

A number of different methods for enumerating dominant sets are summarized in [3]. In this thesis we restrict our attention to the *peeling* and *multistart* strategies. The peeling strategy is outlined in Algorithm 1. It works by 1) finding a dominant set, 2) removing corresponding nodes from the graph, and 3) repeating until the graph is empty or the desired number of clusters have been found. The multistart strategy initializes an iterative algorithm for a number of different starting points, with the expectation that they to converge to different local optimas. However, it is difficult to guarantee that the algorithm converges to different local optimas, and a post-processing step is therefore required to handle overlapping clusters.

Algorithm 1 Peeling strategy for partitioning

```
 1: procedure SPLIT($\mathcal{G}$, $K$)    ▷ Graph $\mathcal{G} = (V, E, w)$, maximum number of clusters $K$.
 2:     $P := \emptyset$                   ▷ Partitions
 3:     $k := 0$                           ▷ Current number of clusters
 4:     while $V \neq \emptyset$ and $k < K$ do
 5:         $S \leftarrow$ DOMINANT-SET($G$)
 6:         $P \leftarrow P \cup \{S\}$
 7:         $V \leftarrow V \setminus S$
 8:         $k \leftarrow k + 1$
 9:     end while
10:     return $P$
11: end procedure
```

The procedure `DOMINANT-SET` computes a local solution to problem (3.2) (or (3.4)) and returns its support.

3.2 Replicator Dynamics

Replicator dynamics (RD) is a common method for solving problem (3.2) [10, 2]. The specific form of replicator dynamics used by DSC to compute it is a discrete-time dynamical system defined as

$$x_i(t+1) = x_i(t)\frac{\mathbf{a}_{i*}^T\mathbf{x}}{\mathbf{x}(t)^T\mathbf{A}\mathbf{x}(t)}, \quad i = 1, ..., n. \tag{3.5}$$

The numerator is the inner product of the i-th row of $\mathbf{A}$ and $\mathbf{x}$. We say that a point $\mathbf{x}$ is *stationary* if $x_i(t+1) = x_i(t)$, $i = 1, ..., n$, and it is also *asymptotically stable* if every trajectory that starts close to $\mathbf{x}$ will converge to $\mathbf{x}$ as $t \to \infty$. It can be shown that any trajectory of (3.5) with initial value $\mathbf{x}(0) \in \Delta$, will converge to a stationary point in Δ. Furthermore, stationary $\mathbf{x} \in \Delta$ is asymptotically stable in (3.5) if and only if it is a strict local solution of (3.2) [10]. This correspondence is why replicator dynamics can be used to find dominant sets – the asymptotically stable stationary points are local solutions to problem (3.2).

Some issues with replicator dynamics for computing dominant sets are explained in [6, 3]. The per-iteration time complexity is $\mathcal{O}(n^2)$ and the convergence can require many iterations. Since the convergence can be slow, it is usually necessary to upperbound the number of iterations. However, this leads to a second problem – if the dynamics is stopped before converge, the current iterate $\mathbf{x}(t)$ is not an exact solution. A second parameter is therefore required, called cutoff, which defines a threshold for the support in Definition 4. That is, only the component values of $\mathbf{x}(t)$ that are greater than or equal to the cutoff are considered part of the support, as opposed to including all nonzero components. Setting an upperbound on the number of iterations and a threshold for the cutoff can be difficult, and bad values might result in splitting a good cluster in the middle.

3.3 Frank-Wolfe

Let $\mathcal{P} \subset \mathbb{R}^n$ be a finite set of vertices (points) and $\mathcal{D} = \text{convex}(\mathcal{P})$ its convex hull (convex polytope). The Frank-Wolfe algorithm, first introduced in [11], is an iterative method for constrained optimization

$$\max_{\mathbf{x} \in \mathcal{D}} f(\mathbf{x}), \tag{3.6}$$

where f is nonlinear and differentiable. In the original formulation there was the additional assumption that f was also concave, but more recently this has been relaxed to only assume it has L-Lipschitz ("well-behaved") gradient [12].

Algorithm 2 outlines the high-level steps of Frank-Wolfe.

Algorithm 2 Frank-Wolfe pseudocode

```
 1: procedure PSEUDO-FW($f$, $\mathcal{D}$, $T$)    ▷ Function $f$, convex polytope $\mathcal{D}$, and iterations $T$.
 2:     Select $\mathbf{x}_0 \in \mathcal{D}$
 3:     for $t = 0, ..., T-1$ do
 4:         if $\mathbf{x}_t$ is stationary then break
 5:         Compute feasible ascent direction $\mathbf{d}_t$ at $\mathbf{x}_t$
 6:         Compute step size $\gamma_t \in [0,1]$ such that $f(\mathbf{x}_t + \gamma_t \mathbf{d}_t) > f(\mathbf{x}_t)$
 7:         $\mathbf{x}_{t+1} := \mathbf{x}_t + \gamma_t \mathbf{d}_t$
 8:     end for
 9:     return $\mathbf{x}_t$
10: end procedure
```

From the definition of $\mathcal{D}$, any point $\mathbf{x}_t \in \mathcal{D}$ can be written as a convex combination of the vertices in $\mathcal{P}$

$$\mathbf{x}_t = \sum_{\mathbf{v} \in \mathcal{P}} \lambda_{\mathbf{v}}^{(t)} \mathbf{v}, \tag{3.7}$$

where the coefficients $\lambda_{\mathbf{v}}^{(t)} \in [0,1]$ and $\sum_{\mathbf{v} \in \mathcal{P}} \lambda_{\mathbf{v}}^{(t)} = 1$. Define

$$S_t = \{\mathbf{v} \in \mathcal{P} : \lambda_{\mathbf{v}}^{(t)} > 0\} \tag{3.8}$$

as the set of vertices with nonzero coefficients at iterate $\mathbf{x}_t$.

Three variants of the Frank-Wolfe algorithm are considered in this thesis: standard FW, pairwise FW (PFW), and away-steps FW (AFW) – they differ in how the ascent direction $\mathbf{d}_t$ is computed.

Let

$$\mathbf{s}_t \in \arg\max_{\mathbf{s} \in \mathcal{D}} \nabla f(\mathbf{x}_t)^T \mathbf{s}, \tag{3.9}$$

$$\mathbf{v}_t \in \arg\min_{\mathbf{v} \in S_t} \nabla f(\mathbf{x}_t)^T \mathbf{v}. \tag{3.10}$$

Since $\mathcal{D}$ is a convex polytope, (3.9) is equivalent to maximizing over the vertices $\mathcal{P}$. That is, (3.9) is the vertex that maximizes the linearization, and (3.10) is the vertex with nonzero coefficient that minimizes it.

Let $\mathbf{x}_t$ be the current iterate and define

$$\begin{aligned} \mathbf{d}_t^{A} &= \mathbf{x}_t - \mathbf{v}_t, \\ \mathbf{d}_t^{FW} &= \mathbf{s}_t - \mathbf{x}_t, \\ \mathbf{d}_t^{PFW} &= \mathbf{s}_t - \mathbf{v}_t, \\ \mathbf{d}_t^{AFW} &= \begin{cases} \mathbf{d}_t^{FW}, & \text{if } \nabla f(\mathbf{x}_t)^T \mathbf{d}_t^{FW} \geq f(\mathbf{x}_t)^T \mathbf{d}_t^{A} \\ \frac{\lambda_{\mathbf{v}_t}^{(t)}}{1-\lambda_{\mathbf{v}_t}^{(t)}} \mathbf{d}_t^{A}, & \text{otherwise} \end{cases} \end{aligned} \tag{3.11}$$

as the away, FW, pairwise, and away/FW direction, respectively. The FW direction moves towards a "good" vertex, and the away direction moves away from a "bad" vertex. The pairwise direction shifts weight from a "bad" vertex to a "good" vertex [4]. The coefficient for $\mathbf{d}_t^{A}$ in $\mathbf{d}_t^{AFW}$ ensures the next iterate remains feasible.

An issue with standard FW, which PFW and AFW attempt to remedy, is the zig-zagging phenomenon. This occurs when the optimal solution to (3.6) lies on the boundary of the domain. The name comes from the fact that the iterates starts to zig-zag between the vertices defining the face the solution lies on, which negatively impacts the convergence. By adding the possibility of an away step in AFW, or alternatively, using the pairwise direction, the zig-zagging is attenuated and its impact is therefore reduced. See figure 1 in [4] for an illustration.

When feasible, the step size γ_t is computed by line-search

$$\gamma_t \in \arg\max_{\gamma \in [0,1]} f(\mathbf{x}_t + \gamma \mathbf{d}_t). \tag{3.12}$$

Finally, the Frank-Wolfe gap is used to check if an iterate is (close enough to) a stationary point.

Definition 6. The Frank-Wolfe gap g_t of $f : \mathcal{D} \to \mathbb{R}$ at iterate $\mathbf{x}_t$ is defined as

$$\begin{aligned} g_t &= \max_{\mathbf{s} \in \mathcal{D}} \nabla f(\mathbf{x}_t)^T (\mathbf{s} - \mathbf{x}_t) \\ &\iff \\ g_t &= \nabla f(\mathbf{x}_t)^T \mathbf{d}_t^{FW}. \end{aligned} \tag{3.13}$$

A point $\mathbf{x}_t$ is stationary if and only if $g_t = 0$, meaning there are no ascent directions. The Frank-Wolfe gap is thus a reasonable measure of nonstationarity and is frequently used as a stopping criterion [12]. Specifically, a threshold ϵ is defined, and if $g_t \leq \epsilon$, then we conclude the iterate is sufficiently close to a stationary point and stop the algorithm.

3.4 Related Work

This section summarizes the most closely related work. Dominant Set Clustering is introduced in [2] and used replicator dynamics to solve problem (3.2). To fix some of the shortcomings with replicator dynamics, e.g. quadratic per-iteration time complexity, the infection and immunization dynamics (InImDyn) is presented in [7] with linear per-iteration time complexity. An alternative to replicator dynamics is also presented in [6], which proposes a clustering algorithm based on identifying sharp changes of the component values in the dynamics (3.5), which correspond to cuts of the data. The DSC framework is summarized in survey [3] and provides a good overview.

Standard, pairwise, and away-steps FW for convex objectives are summarized in [4]. Convergence rate, in terms of the Frank-Wolfe gap, is shown to be $\mathcal{O}(1/\sqrt{t})$ for standard FW with nonconvex objective in [12]. The same convergence rate is shown for away-steps FW with nonconvex objective and simplex constraint in [13]. Support identification in finite time with pairwise FW and away-steps FW is proved in [14] for nonconvex objective with simplex constraint.

Our contribution in this context is the analysis of the Frank-Wolfe variants when applied to problem (3.2), as well as their experimental evaluation when used for clustering with DSC.

4

Frank-Wolfe for StQP

This chapter derives algorithms for the Frank-Wolfe variants when applied to problem (3.2)*. Their convergence rates are analyzed at the end of the chapter.*

4.1 Simplex Domain

The vertices of the simplex Δ are the standard basis vectors $\mathbf{e}_i$, component i set to 1 and remaining components set to 0. Thus for $\mathbf{x} \in \Delta$ we have

$$\mathbf{x} = \sum_{i=1}^{n} \lambda_{\mathbf{e}_i} \mathbf{e}_i,$$

from (3.7), and $\lambda_{\mathbf{e}_i} = x_i$. That is, the coefficients in the convex combination correspond to the components of $\mathbf{x}$. The set of vertices with nonzero coefficients at iterate $\mathbf{x}_t$, (3.8), is then equivalent to the support

$$\sigma_t = \{i \in V : x_i^{(t)} > 0\}.$$

Due to the structure of the simplex Δ, (3.9) is equivalent to

$$\begin{cases} \mathbf{s}_t & \in \Delta \\ s_i^{(t)} & = 1, \quad \text{where } i \in \arg\max_i \nabla_i f(\mathbf{x}_t) \\ s_j^{(t)} & = 0, \quad \text{for } j \neq i, \end{cases} \tag{4.1}$$

and (3.10) is equivalent to

$$\begin{cases} \mathbf{v}_t & \in \Delta \\ v_i^{(t)} & = 1, \quad \text{where } i \in \arg\min_{i \in \sigma_t} \nabla_i f(\mathbf{x}_t) \\ v_j^{(t)} & = 0, \quad \text{for } j \neq i. \end{cases} \tag{4.2}$$

That is, the max and min value of the linearization are the largest and smallest components of the gradient, respectively (subject to $i \in \sigma_t$ in the latter case). Note $\nabla f(\mathbf{x}_t) = 2\mathbf{A}\mathbf{x}_t$.

4.2 Step Sizes

We compute the optimal step sizes for FW, PFW, and AFW. Iterate subscripts are omitted for clarity. Define the step size function

$$\begin{aligned}\psi(\gamma) &= f(\mathbf{x}+\gamma\mathbf{d}) \\ &= (\mathbf{x}+\gamma\mathbf{d})^T\mathbf{A}(\mathbf{x}+\gamma\mathbf{d}) \\ &= \mathbf{x}^T\mathbf{A}\mathbf{x} + 2\gamma\mathbf{d}^T\mathbf{A}\mathbf{x} + \gamma^2\mathbf{d}^T\mathbf{A}\mathbf{d} \\ &= f(\mathbf{x}) + \gamma\nabla f(\mathbf{x}_t)^T\mathbf{d} + \gamma^2\mathbf{d}^T\mathbf{A}\mathbf{d},\end{aligned} \tag{4.3}$$

where $\mathbf{A}$ satisfies (3.2). This is a single variable second degree polynomial in γ. The function is concave if the coefficient $\mathbf{d}^T\mathbf{A}\mathbf{d} \leq 0$ – second derivative test – and admits a global maximum in that case.

In the following it is assumed that $\mathbf{s}$ and $\mathbf{v}$ satisfy (4.1) and (4.2), and their nonzero components are i and j, respectively.

(FW direction) Substitute $\mathbf{d}^{FW} = \mathbf{s} - \mathbf{x}$ into $\mathbf{d}^T\mathbf{A}\mathbf{d}$.

$$\begin{aligned}\mathbf{d}^T\mathbf{A}\mathbf{d} &= (\mathbf{s}-\mathbf{x})^T\mathbf{A}(\mathbf{s}-\mathbf{x}) \\ &= \mathbf{s}^T\mathbf{A}\mathbf{s} - 2\mathbf{s}^T\mathbf{A}\mathbf{x} + \mathbf{x}^T\mathbf{A}\mathbf{x} \\ &= -(2\mathbf{s}^T\mathbf{A}\mathbf{x} - \mathbf{x}^T\mathbf{A}\mathbf{x}) \\ &= \mathbf{x}^T\mathbf{A}\mathbf{x} - 2\mathbf{a}_{i*}^T\mathbf{x}.\end{aligned} \tag{4.4}$$

(Pairwise direction) Substitute $\mathbf{d}^{PFW} = \mathbf{s} - \mathbf{v}$ into $\mathbf{d}^T\mathbf{A}\mathbf{d}$.

$$\begin{aligned}\mathbf{d}^T\mathbf{A}\mathbf{d} &= (\mathbf{s}-\mathbf{v})^T\mathbf{A}(\mathbf{s}-\mathbf{v}) \\ &= \mathbf{s}^T\mathbf{A}\mathbf{s} - 2\mathbf{v}^T\mathbf{A}\mathbf{s} + \mathbf{v}^T\mathbf{A}\mathbf{v} \\ &= -2a_{ij}.\end{aligned} \tag{4.5}$$

(Away direction) Substitute $\mathbf{d}^{A} = \mathbf{x} - \mathbf{v}$ into $\mathbf{d}^T\mathbf{A}\mathbf{d}$.

$$\begin{aligned}\mathbf{d}^T\mathbf{A}\mathbf{d} &= (\mathbf{x}-\mathbf{v})^T\mathbf{A}(\mathbf{x}-\mathbf{v}) \\ &= \mathbf{x}^T\mathbf{A}\mathbf{x} - 2\mathbf{v}^T\mathbf{A}\mathbf{x} + \mathbf{v}^T\mathbf{A}\mathbf{v} \\ &= \mathbf{x}^T\mathbf{A}\mathbf{x} - 2\mathbf{a}_{j*}^T\mathbf{x}.\end{aligned} \tag{4.6}$$

Recall $\mathbf{A}$ has nonnegative entries and zeros on the main diagonal. Therefore $\mathbf{s}^T\mathbf{A}\mathbf{s} = 0$ and $\mathbf{v}^T\mathbf{A}\mathbf{v} = 0$. From $\mathbf{x}^T\mathbf{A}\mathbf{x} \leq \mathbf{s}^T\mathbf{A}\mathbf{x}$ we get that (4.4) is nonpositive, and it is also immediate that (4.5) is nonpositive. The corresponding step size functions are therefore always concave. In (4.6) we cannot conclude anything and the sign of $\mathbf{d}^T\mathbf{A}\mathbf{d}$ is dependent on the iterate.

The derivative of $\psi(\gamma)$ is

$$\frac{d\psi}{d\gamma}(\gamma) = \nabla f(\mathbf{x})^T\mathbf{d} + 2\gamma\mathbf{d}^T\mathbf{A}\mathbf{d}.$$

Solve for γ in $\frac{d\psi}{d\gamma}(\gamma) = 0$

$$\begin{aligned} \nabla f(\mathbf{x})^T \mathbf{d} + 2\gamma \mathbf{d}^T \mathbf{A} \mathbf{d} = 0 \\ \iff \\ \gamma^* = -\frac{\nabla f(\mathbf{x})^T \mathbf{d}}{2\mathbf{d}^T \mathbf{A} \mathbf{d}} = -\frac{\mathbf{x}^T \mathbf{A} \mathbf{d}}{\mathbf{d}^T \mathbf{A} \mathbf{d}}. \end{aligned} \tag{4.7}$$

Since $\nabla f(\mathbf{x})^T \mathbf{d} \geq 0$, we also see here that $\mathbf{d}^T \mathbf{A} \mathbf{d} < 0$ has to hold in order for the step size to make sense.

Substitute the directions and corresponding $\mathbf{d}^T \mathbf{A} \mathbf{d}$ into (4.7).

FW direction and (4.4).

$$\gamma^{FW} = -\frac{\mathbf{x}^T \mathbf{A} \mathbf{d}}{\mathbf{d}^T \mathbf{A} \mathbf{d}} = \frac{\mathbf{a}_{i*}^T \mathbf{x} - \mathbf{x}^T \mathbf{A} \mathbf{x}}{2\mathbf{a}_{i*}^T \mathbf{x} - \mathbf{x}^T \mathbf{A} \mathbf{x}}. \tag{4.8}$$

Pairwise direction and (4.5).

$$\gamma^{PFW} = -\frac{\mathbf{x}^T \mathbf{A} \mathbf{d}}{\mathbf{d}^T \mathbf{A} \mathbf{d}} = \frac{\mathbf{a}_{i*}^T \mathbf{x} - \mathbf{a}_{j*}^T \mathbf{x}}{2a_{ij}}. \tag{4.9}$$

Away direction and (4.6).

$$\gamma^{A} = -\frac{\mathbf{x}^T \mathbf{A} \mathbf{d}}{\mathbf{d}^T \mathbf{A} \mathbf{d}} = \frac{\mathbf{x}^T \mathbf{A} \mathbf{x} - \mathbf{a}_{j*}^T \mathbf{x}}{2\mathbf{a}_{j*}^T \mathbf{x} - \mathbf{x}^T \mathbf{A} \mathbf{x}}. \tag{4.10}$$

4.3 Algorithms

This section combines the results from the previous two sections and arrives at standard FW, PFW, and AFW algorithms for problem (3.2), following the high-level structure of Algorithm 2. All variants have $\mathcal{O}(n)$ as the per-iteration time complexity; the linear operations are arg max, arg min, and vector addition.

Standard FW

Algorithm 3 FW for StQP

```
 1: procedure FW($\mathbf{A}$, $\epsilon$, $T$)
 2:     Select $\mathbf{x}_0 \in \Delta$
 3:     $\mathbf{r}_0 := \mathbf{A}\mathbf{x}_0$
 4:     $f_0 := \mathbf{r}_0^T \mathbf{x}_0$
 5:     for $t = 0, ..., T-1$ do
 6:         $\mathbf{s}_t := \mathbf{e}_i$, where $i \in \arg\max_{\ell} r_\ell^{(t)}$
 7:         $g_t := r_i^{(t)} - f_t$
 8:         if $g_t \le \epsilon$ then break
 9:         $\gamma_t := \frac{r_i^{(t)} - f_t}{2r_i^{(t)} - f_t}$
10:         $\mathbf{x}_{t+1} := (1-\gamma_t)\mathbf{x}_t + \gamma_t \mathbf{s}_t$
11:         $\mathbf{r}_{t+1} := (1-\gamma_t)\mathbf{r}_t + \gamma_t \mathbf{a}_{*i}$
12:         $f_{t+1} := (1-\gamma_t)^2 f_t + 2\gamma_t(1-\gamma_t) r_i^{(t)}$
13:     end for
14:     return $\mathbf{x}_t$
15: end procedure
```

Lemma 1. *For $\mathbf{x}_{t+1} = (1-\gamma_t)\mathbf{x}_t + \gamma_t\mathbf{s}_t$, lines 11 and 12 in Algorithm 3 satisfy*

$$\begin{aligned} \mathbf{r}_{t+1} &= \mathbf{A}\mathbf{x}_{t+1}, \\ f_{t+1} &= \mathbf{x}_{t+1}^T \mathbf{A}\mathbf{x}_{t+1}. \end{aligned}$$

Proof. By definition (lines 3 and 4), $\mathbf{r}_0 = \mathbf{A}\mathbf{x}_0$ and $f_0 = \mathbf{x}_0^T\mathbf{A}\mathbf{x}_0$. Let $\mathbf{x} = \mathbf{x}_t$, $\mathbf{s} = \mathbf{s}_t$, and $\gamma = \gamma_t$. Assume $\mathbf{r}_t = \mathbf{A}\mathbf{x}$ and $f_t = \mathbf{x}^T\mathbf{A}\mathbf{x}$ holds. Expand the definition of $\mathbf{A}\mathbf{x}_{t+1}$ and proceed by induction.

$$\begin{aligned} \mathbf{A}\mathbf{x}_{t+1} &= \mathbf{A}((1-\gamma)\mathbf{x} + \gamma\mathbf{s}) \\ &= (1-\gamma)\mathbf{A}\mathbf{x} + \gamma\mathbf{A}\mathbf{s} \\ &= (1-\gamma)\mathbf{r}_t + \gamma\mathbf{a}_{*i} \\ &= \mathbf{r}_{t+1}, \\ \mathbf{x}_{t+1}^T\mathbf{A}\mathbf{x}_{t+1} &= ((1-\gamma)\mathbf{x} + \gamma\mathbf{s})^T\mathbf{A}((1-\gamma)\mathbf{x} + \gamma\mathbf{s}) \\ &= (1-\gamma)^2\mathbf{x}^T\mathbf{A}\mathbf{x} + 2\gamma(1-\gamma)\mathbf{s}^T\mathbf{A}\mathbf{x} + \gamma^2\mathbf{s}^T\mathbf{A}\mathbf{s} \\ &= (1-\gamma)^2\mathbf{x}^T\mathbf{A}\mathbf{x} + 2\gamma(1-\gamma)\mathbf{s}^T\mathbf{A}\mathbf{x} \\ &= (1-\gamma)^2 f_t + 2\gamma(1-\gamma) r_i^{(t)} \\ &= f_{t+1}. \end{aligned}$$

Note $\mathbf{s}^T\mathbf{A}\mathbf{s} = 0$ from the definition of $\mathbf{s}$ and $\mathbf{A}$. ■

Pairwise FW

Algorithm 4 Pairwise FW for StQP

```
1: procedure PFW($\mathbf{A}$, $\epsilon$, $T$)
2:     Select $\mathbf{x}_0 \in \Delta$
3:     $\mathbf{r}_0 := \mathbf{A}\mathbf{x}_0$
4:     $f_0 := \mathbf{r}_0^T \mathbf{x}_0$
5:     for $t = 0, ..., T-1$ do
6:         $\sigma_t := \{i \in V : x_i^{(t)} > 0\}$
7:         $\mathbf{s}_t := \mathbf{e}_i$, where $i \in \arg\max_{\ell} r_\ell^{(t)}$
8:         $\mathbf{v}_t := \mathbf{e}_j$, where $j \in \arg\min_{\ell \in \sigma_t} r_\ell^{(t)}$
9:         $g_t := r_i^{(t)} - f_t$
10:        if $g_t \leq \epsilon$ then break
11:        $\gamma_t := \min\left(x_j^{(t)}, \frac{r_i^{(t)} - r_j^{(t)}}{2a_{ij}}\right)$
12:        $\mathbf{x}_{t+1} := \mathbf{x}_t + \gamma_t(\mathbf{s}_t - \mathbf{v}_t)$
13:        $\mathbf{r}_{t+1} := \mathbf{r}_t + \gamma_t(\mathbf{a}_{*i} - \mathbf{a}_{*j})$
14:        $f_{t+1} := f_t + 2\gamma_t(r_i^{(t)} - r_j^{(t)}) - 2\gamma_t^2 a_{ij}$
15:    end for
16:    return $\mathbf{x}_t$
17: end procedure
```

Lemma 2. *For $\mathbf{x}_{t+1} = \mathbf{x}_t + \gamma_t(\mathbf{s}_t - \mathbf{v}_t)$, lines 13 and 14 in Algorithm 4 satisfy*

$$\mathbf{r}_{t+1} = \mathbf{A}\mathbf{x}_{t+1},$$
$$f_{t+1} = \mathbf{x}_{t+1}^T \mathbf{A}\mathbf{x}_{t+1}.$$

Proof. Proceed as in proof of Lemma 1. Let $\mathbf{x} = \mathbf{x}_t$, $\mathbf{s} = \mathbf{s}_t$, $\mathbf{v} = \mathbf{v}_t$, and $\gamma = \gamma_t$.

$$\begin{aligned}
\mathbf{A}\mathbf{x}_{t+1} &= \mathbf{A}(\mathbf{x} + \gamma(\mathbf{s} - \mathbf{v})) \\
&= \mathbf{A}\mathbf{x} + \gamma(\mathbf{A}\mathbf{s} - \mathbf{A}\mathbf{v}) \\
&= \mathbf{r}_t + \gamma(\mathbf{a}_{*i} - \mathbf{a}_{*j}) \\
&= \mathbf{r}_{t+1}, \\
\mathbf{x}_{t+1}^T \mathbf{A}\mathbf{x}_{t+1} &= (\mathbf{x} + \gamma(\mathbf{s} - \mathbf{v}))^T \mathbf{A}(\mathbf{x} + \gamma(\mathbf{s} - \mathbf{v})) \\
&= \mathbf{x}^T\mathbf{A}\mathbf{x} + 2\gamma(\mathbf{s} - \mathbf{v})^T\mathbf{A}\mathbf{x} + \gamma^2(\mathbf{s} - \mathbf{v})^T\mathbf{A}(\mathbf{s} - \mathbf{v}) \\
&= \mathbf{x}^T\mathbf{A}\mathbf{x} + 2\gamma(\mathbf{s}^T\mathbf{A}\mathbf{x} - \mathbf{v}^T\mathbf{A}\mathbf{x}) - 2\gamma^2 a_{ij} \\
&= f_t + 2\gamma(r_i^{(t)} - r_j^{(t)}) - 2\gamma^2 a_{ij} \\
&= f_{t+1}.
\end{aligned}$$

■

Away-steps FW

Algorithm 5 Away-steps FW for StQP

```
procedure AFW($\mathbf{A}$, $\epsilon$, $T$)
    Select $\mathbf{x}_0 \in \Delta$
    $\mathbf{r}_0 := \mathbf{A}\mathbf{x}_0$
    $f_0 := \mathbf{r}_0^T \mathbf{x}_0$
    for $t = 0, ..., T-1$ do
        $\sigma_t := \{i \in V : x_i^{(t)} > 0\}$
        $\mathbf{s}_t := \mathbf{e}_i$, where $i \in \arg\max_{\ell} r_\ell^{(t)}$
        $\mathbf{v}_t := \mathbf{e}_j$, where $j \in \arg\min_{\ell \in \sigma_t} r_\ell^{(t)}$
        $g_t := r_i^{(t)} - f_t$
        if $g_t \leq \epsilon$ then break
        if $(r_i^{(t)} - f_t) \geq (f_t - r_j^{(t)})$ then                ▷ FW direction
            $\gamma_t := \frac{r_i^{(t)} - f_t}{2r_i^{(t)} - f_t}$
            $\mathbf{x}_{t+1} := (1-\gamma_t)\mathbf{x}_t + \gamma_t \mathbf{s}_t$
            $\mathbf{r}_{t+1} := (1-\gamma_t)\mathbf{r}_t + \gamma_t \mathbf{a}_{*i}$
            $f_{t+1} := (1-\gamma_t)^2 f_t + 2\gamma_t(1-\gamma_t) r_i^{(t)}$
        else                                                          ▷ Away direction
            $\gamma_t := x_j^{(t)} / (1 - x_j^{(t)})$
            if $(2r_j^{(t)} - f_t) > 0$ then
                $\gamma_t \leftarrow \min\left(\gamma_t, \frac{f_t - r_j^{(t)}}{2r_j^{(t)} - f_t}\right)$
            end if
            $\mathbf{x}_{t+1} := (1+\gamma_t)\mathbf{x}_t - \gamma_t \mathbf{v}_t$
            $\mathbf{r}_{t+1} := (1+\gamma_t)\mathbf{r}_t - \gamma_t \mathbf{a}_{*j}$
            $f_{t+1} := (1+\gamma_t)^2 f_t - 2\gamma_t(1+\gamma_t) r_j^{(t)}$
        end if
    end for
    return $\mathbf{x}_t$
end procedure
```

Lines 12-15 are identical to the updates in Algorithm 3 and are included in Lemma 1. We therefore only show the away direction.

Lemma 3. *For $\mathbf{x}_{t+1} = (1+\gamma_t)\mathbf{x}_t - \gamma_t \mathbf{v}_t$, lines 22 and 23 in Algorithm 5 satisfy*

$$\mathbf{r}_{t+1} = \mathbf{A}\mathbf{x}_{t+1},$$
$$f_{t+1} = \mathbf{x}_{t+1}^T \mathbf{A}\mathbf{x}_{t+1}.$$

Proof. Proceed as in proof of Lemma 1. Let $\mathbf{x} = \mathbf{x}_t$, $\mathbf{v} = \mathbf{v}_t$, and $\gamma = \gamma_t$.

$$\begin{aligned}
\mathbf{A}\mathbf{x}_{t+1} &= \mathbf{A}((1+\gamma)\mathbf{x} - \gamma\mathbf{v}) \\
&= (1+\gamma)\mathbf{A}\mathbf{x} - \gamma\mathbf{A}\mathbf{v} \\
&= (1+\gamma)\mathbf{r}_t - \gamma\mathbf{a}_{*j} \\
&= \mathbf{r}_{t+1}, \\
\mathbf{x}_{t+1}^T\mathbf{A}\mathbf{x}_{t+1} &= ((1+\gamma)\mathbf{x} - \gamma\mathbf{v})^T\mathbf{A}((1+\gamma)\mathbf{x} - \gamma\mathbf{v}) \\
&= (1+\gamma)^2\mathbf{x}^T\mathbf{A}\mathbf{x} - 2\gamma(1+\gamma)\mathbf{v}^T\mathbf{A}\mathbf{x} + \gamma^2\mathbf{v}^T\mathbf{A}\mathbf{v} \\
&= (1+\gamma)^2\mathbf{x}^T\mathbf{A}\mathbf{x} - 2\gamma(1+\gamma)\mathbf{v}^T\mathbf{A}\mathbf{x} \\
&= (1+\gamma)^2 f_t - 2\gamma(1+\gamma)r_j^{(t)} \\
&= f_{t+1}.
\end{aligned}$$

■

Algorithm 5 (AFW) is actually equivalent to the infection and immunization dynamics (InImDyn) with the pure strategy selection function, introduced in [7] as an alternative to replicator dynamics. However, InImDyn is derived from the perspective of evolutionary game theory as opposed to Frank-Wolfe. We summarize this observation in the following proposition.

Proposition 1. *Algorithm 5 (AFW) is equivalent to Algorithm 1 in [7].*

4.4 Convergence Rates

In [12] it was shown that the Frank-Wolfe gap for standard FW decreases at rate $\mathcal{O}(1/\sqrt{t})$ for nonconvex objective functions, where t is the number of iterations. The same convergence rate was shown in [13] for nonconvex AFW over the simplex. When the objective function is convex, linear convergence rates for PFW and AFW are shown in [4]. We have not found a proof for the convergence rate in terms of the Frank-Wolfe gap for nonconvex PFW.

Following the terminology and techniques in [12, 4, 13], in this section we present a unified framework to analyze convergence rates for Algorithms 3, 4, and 5. The analysis is split into a number of different cases, where each case handles a unique search direction and step size combination. For the step sizes, we consider one case when the optimal step size is used ($\gamma_t < \gamma_{max}$), and a second case when it has been truncated ($\gamma_t = \gamma_{max}$). The former case is referred to as a good step, as in this case we can provide a lower bound on the progress $f(\mathbf{x}_{t+1}) - f(\mathbf{x}_t)$ in terms of the Frank-Wolfe gap. The latter case is referred to as a drop step or a swap step. It is called a drop step when the cardinality of the support is reduced by one, i.e. $|\sigma_{t+1}| = |\sigma_t| - 1$, and it is called a swap step when it remains unchanged, i.e. $|\sigma_{t+1}| = |\sigma_t|$. When $\gamma_t = \gamma_{max}$ we cannot provide a bound on the progress in terms of the Frank-Wolfe gap, and instead we bound the number of drop/swap steps. Furthermore, this case can only happen for PFW and AFW as the step size for FW always satisfies $\gamma_t < \gamma_{max}$. Swap steps can only happen for PFW.

Let $\mathbf{y} = \mathbf{x}_t + \gamma_t \mathbf{d}_t$, for some direction $\mathbf{d}_t$, $\mathbf{r}(\mathbf{x}) = \mathbf{A}\mathbf{x}$, and $f(\mathbf{x}) = \mathbf{x}^T\mathbf{A}\mathbf{x}$. From (4.3) we have

$$f(\mathbf{y}) = f(\mathbf{x}_t) + 2\gamma_t \mathbf{r}(\mathbf{x}_t)^T \mathbf{d}_t + \gamma_t^2 \mathbf{d}_t^T \mathbf{A}\mathbf{d}_t. \tag{4.11}$$

Using

$$\gamma_t = -\frac{(\mathbf{x}_t)^T \mathbf{A}\mathbf{d}_t}{\mathbf{d}_t \mathbf{A}\mathbf{d}_t} = -\frac{\mathbf{r}(\mathbf{x}_t)^T \mathbf{d}_t}{\mathbf{d}_t^T \mathbf{A}\mathbf{d}_t}$$

from (4.7), we get

$$\begin{aligned} f(\mathbf{y}) &= f(\mathbf{x}_t) - 2\frac{(\mathbf{r}(\mathbf{x}_t)^T\mathbf{d}_t)^2}{\mathbf{d}_t^T\mathbf{A}\mathbf{d}_t} + \frac{(\mathbf{r}(\mathbf{x}_t)^T\mathbf{d}_t)^2}{\mathbf{d}_t^T\mathbf{A}\mathbf{d}_t} \\ &= f(\mathbf{x}_t) - \frac{(\mathbf{r}(\mathbf{x}_t)^T\mathbf{d}_t)^2}{\mathbf{d}_t^T\mathbf{A}\mathbf{d}_t} \\ &\iff \\ (\mathbf{r}(\mathbf{x}_t)^T\mathbf{d}_t)^2 &= -\mathbf{d}_t^T\mathbf{A}\mathbf{d}_t \left(f(\mathbf{y}) - f(\mathbf{x}_t)\right). \end{aligned} \tag{4.12}$$

Let

$$\tilde{g}_t = \min_{0 \le \ell \le t} g_\ell, \quad \underline{M} = \min_{i,j: i \ne j} a_{ij}, \quad \overline{M} = \max_{i,j: i \ne j} a_{ij},$$

be the smallest Frank-Wolfe gap after t iterations, and the smallest and largest off-diagonal elements of $\mathbf{A}$.

Let $\mathbf{s}_t$ satisfy (4.1) and $\mathbf{v}_t$ satisfy (4.2). Denote their nonzero components by i and j, respectively. Let I be the indexes that take a good step. That is, for $t \in I$ we have $\gamma_t < \gamma_{max}$.

Lemma 4. *The smallest Frank-Wolfe gap for Algorithms 3, 4, and 5 satisfy*

$$\tilde{g}_t \le 2\sqrt{\frac{\beta\left(f(\mathbf{x}_t) - f(\mathbf{x}_0)\right)}{|I|}}, \tag{4.13}$$

where $\beta = 2\overline{M} - \underline{M}$ for FW and AFW, and $\beta = 2\overline{M}$ for PFW.

Proof. We consider the FW, away, and pairwise directions $\mathbf{d}_t$ with step size $\gamma_t < \gamma_{max}$. Note that $f(\mathbf{y}) = f(\mathbf{x}_{t+1})$ holds in (4.12) for such directions and step sizes.

Let $h_t = f(\mathbf{x}_{t+1}) - f(\mathbf{x}_t)$ and $g_t = 2(r(\mathbf{x}_t)_i - f(\mathbf{x}_t))$.

(FW direction) Substitute $\mathbf{d}_t = \mathbf{s}_t - \mathbf{x}_t$ and (4.4) into (4.12).

$$(r(\mathbf{x}_t)_i - f(\mathbf{x}_t))^2 = (2r(\mathbf{x}_t)_i - f(\mathbf{x}_t))h_t \implies g_t^2 \le 4(2\overline{M} - \underline{M})h_t$$

(Away direction) For this direction with $\gamma_t < \gamma_{max}$ we have

$$r(\mathbf{x}_t)_i - f(\mathbf{x}_t) < f(\mathbf{x}_t) - r(\mathbf{x}_t)_j,$$

from line 11 in Algorithm 5. Substitute $\mathbf{d}_t = \mathbf{x}_t - \mathbf{v}_t$ and (4.6) into (4.12).

$$(f(\mathbf{x}_t) - r(\mathbf{x}_t)_j)^2 = (2r(\mathbf{x}_t)_j - f(\mathbf{x}_t))h_t \implies g_t^2 \leq 4(2\overline{M} - \underline{M})h_t.$$

(Pairwise direction) Substitute $\mathbf{d}_t = \mathbf{s}_t - \mathbf{v}_t$ and (4.5) into (4.12).

$$(r(\mathbf{x}_t)_i - r(\mathbf{x}_t)_j)^2 = 2a_{ij}h_t \implies g_t^2 \leq 8\overline{M}h_t.$$

Using previously defined I and β we get

$$4\beta\,(f(\mathbf{x}_t) - f(\mathbf{x}_0)) = 4\beta\sum_{\ell=0}^{t-1} h_\ell \geq 4\beta\sum_{\ell\in I} h_\ell \geq \sum_{\ell\in I} g_t^2 \geq |I|\tilde{g}_t^2$$
$$\implies$$
$$\tilde{g}_t^2 \leq \frac{4\beta\,(f(\mathbf{x}_t) - f(\mathbf{x}_0))}{|I|} \iff \tilde{g}_t \leq 2\sqrt{\frac{\beta\,(f(\mathbf{x}_t) - f(\mathbf{x}_0))}{|I|}},$$

for either direction $\mathbf{d}_t$. ■

Theorem 2. *The smallest Frank-Wolfe gap for Algorithm 3 (FW) satisfies*

$$\tilde{g}_t^{FW} \leq 2\sqrt{\frac{(2\overline{M} - \underline{M})\,(f(\mathbf{x}_t) - f(\mathbf{x}_0))}{t}}. \tag{4.14}$$

Proof. Since standard FW only takes good steps we have $|I| = t$. The result follows from Lemma 4. ■

Theorem 3. *The smallest Frank-Wolfe gap for Algorithm 4 (PFW) satisfies*

$$\tilde{g}_t^{PFW} \leq 2\sqrt{\frac{6n!\overline{M}\,(f(\mathbf{x}_t) - f(\mathbf{x}_0))}{t}}. \tag{4.15}$$

Proof. When $\gamma_t = \gamma_{max}$ we either have $|\sigma_{t+1}| = |\sigma_t| - 1$ or $|\sigma_{t+1}| = |\sigma_t|$, called drop and swap step, respectively. We need to upper bound the number of these steps in order to get a lower bound for $|I|$.

The following reasoning is from the analysis of PFW with convex objective in [4].

Let n be the dimension of $\mathbf{x}_t$, $m = |\sigma_t|$, and $\mathbf{d}_t = \mathbf{s}_t - \mathbf{v}_t$. Since we are performing line search, we always have $f(\mathbf{x}_\ell) < f(\mathbf{x}_t)$ for all $\ell < t$ that are nonstationary. This means the sequence $\mathbf{x}_0, ..., \mathbf{x}_t$ will not have any duplicates. The set of component values does not change when we perform a swap step:

$$\{x_\ell^{(t)} : \ell = 1, ..., n\} \cap \{x_\ell^{(t+1)} : \ell = 1, ..., n\} = \emptyset.$$

That is, the components are simply permuted after a swap step. The number of possible unique permutations is $\kappa = n!/(n-m)!$. After we have performed κ swap steps, a drop step can be taken which will change the component values. Thus in the worst case, κ swap steps followed by a drop step will be performed until $m = 1$

before a good step is taken. The number of swap/drop steps between two good steps is then bounded by

$$\sum_{\ell=1}^{m} \frac{n!}{(n-\ell)!} \leq n! \sum_{\ell=0}^{\infty} \frac{1}{\ell!} = n!e \leq 3n!.$$

Result (4.15) follows from Lemma 4 and

$$|I| \geq \frac{t}{3n!}.$$

■

Theorem 4. *The smallest Frank-Wolfe gap for Algorithm 5 (AFW) satisfies*

$$\tilde{g}_t^{AFW} \leq 2\sqrt{\frac{2(2\overline{M} - \underline{M})\,(f(\mathbf{x}_t) - f(\mathbf{x}_0))}{t + 1 - |\sigma_0|}}. \tag{4.16}$$

Proof. When $\gamma_t = \gamma_{max}$, $\mathbf{d}_t$ must be the away direction. In this case the support is reduced by one, i.e. $|\sigma_{t+1}| = |\sigma_t| - 1$. Denote these indexes by D. Let $I_A \subseteq I$ be the indexes that adds to the support, i.e. $|\sigma_{t+1}| > |\sigma_t|$ for $t \in I_A$. Similar as before, we need to upper bound $|D|$ in order to get a lower bound for $|I|$.

We have $|I_A| + |D| \leq t$ and $|\sigma_t| = |\sigma_0| + |I_A| - |D]$. Combining the inequalities we get

$$1 \leq |\sigma_t| \leq |\sigma_0| + t - 2|D| \implies |D| \leq \frac{|\sigma_0| - 1 + t}{2}.$$

Result (4.16) then follows from Lemma 4 and

$$|I| = t - |D| \geq t - \frac{(|\sigma_0| - 1 + t)}{2} = \frac{t + 1 - |\sigma_0|}{2}.$$

■

Corollary 1. *The smallest Frank-Wolfe gap for Algorithms 3, 4, and 5 decrease at rate $\mathcal{O}(1/\sqrt{t})$.*

5

Experiments

This chapter presents implementation details, methodology, and experimental results for RD, FW, PFW, and AFW on different datasets

5.1 Method

The replicator dynamics equation (3.5) and Algorithms 3, 4, and 5 are straightforward to implement in any high-level language with support for numerical computation. We have used vectorized implementations in Python and NumPy [15] for the experiments.

Let δ be the cutoff parameter and $\mathbf{x}^*$ a solution from one of the solvers. The support (cluster) is then defined as the set $\{i \in V : x_i^* > \delta\}$. The cutoff is set to $\delta = 2 \cdot 10^{-12}$ unless stated otherwise. Denote the regularization parameter from Section 3.1.3 by α, the number of objects by n, the number of clusters in the dataset by k, the maximum number of clusters to extract by K, and the number of iterations by t.

The Frank-Wolfe gap (Definition 6) and the distance between two consecutive iterates will be used as the stopping criterion for the FW variants and RD, respectively. Specifically, let ϵ be the threshold, then the respective algorithm is stopped if $g_t \leq \epsilon$ or if $||\mathbf{x}_{t+1} - \mathbf{x}_t|| \leq \epsilon$. In the experiments we have set ϵ to `np.sys.float_info.epsilon`, which is $\epsilon \approx 2.2 \cdot 10^{-16}$.

The experiments are divided into two groups. The first group uses the peeling strategy in Algorithm 1, where the `DOMINANT-SET` procedure refers to either RD, FW, PFW, or AFW. The second group uses a combination of the multistart and peeling strategies.

5.1.1 Clustering Metrics

The objective of clustering is to assign objects to groups. For a problem with n objects, the clustering assignment is represented by a discrete n-dimensional vector $\mathbf{c}$. Specifically, $c_i \in \{0, 1, ..., K-1, K\}$, for $i = 1, ..., n$. If $c_i = c_j$, then objects i and j are in the same cluster/group. The discrete values $0, 1, ..., K-1, K$ are called labels and represent the different clusters. Label 0 is designated to represent "no cluster" – an object is either assigned to a unique cluster or left unassigned. If

$c_i = 0$, then object i is unassigned and otherwise it is assigned to the cluster with corresponding label c_i. The predicted and ground truth clustering assignment are denoted by $\mathbf{c}^P$ and $\mathbf{c}^{GT}$, respectively.

To evaluate the clustering quality of the algorithms we will compare the predicted cluster assignments to the ground truth. One difficulty with this is that we are not interested in the specific labels in $\mathbf{c}^P$ and $\mathbf{c}^{GT}$, but rather if they represent the same partitioning of the data. For example, we might have $c_i^P \neq c_i^{GT}$ and

$$\{j : c_j^P = c_i^P\} \cap \{j : c_j^{GT} = c_i^{GT}\} = \emptyset,$$

meaning their labels are different but object i is in the same cluster in both assignments.

Define

$$\begin{aligned} C_P &= \{i \in V : c_i^P \neq 0\}, \\ C_{GT} &= \{i \in V : c_i^{GT} \neq 0\}. \end{aligned}$$

The sets of predicted objects assigned to a cluster and objects actually assigned to a cluster, respectively. Let $C = C_P \cap C_{GT}$ be their intersection. Denote their respective complement by C_P^c, C_{GT}^c, and C^c.

Evaluation of the clustering quality is split into two types of metrics. Assignment rate considers the accuracy in predicting whether an object is unassigned or assigned to any cluster. Adjusted rand score and V-measure are two different metrics for the similarity between $\mathbf{c}^P$ and $\mathbf{c}^{GT}$; in our experiments their similarity comparison will be limited to the objects in C. The perfect score is 1 for all metrics.

Assignment Rate (AR)

The assignment rate is defined as

$$\text{AR } = 1 - \frac{|C^c| - |C_P^c \cap C_{GT}^c|}{n}. \tag{5.1}$$

The fraction is the ratio of the number of objects wrongly predicted as assigned / unassigned, which is then subtracted from 1. The assignment rate is in $[0, 1]$.

Adjusted Rand Index (ARI)

The adjusted rand index [16] is the rand index adjusted for chance. The rand index is the ratio of object pairs that are either in the same cluster, or in different clusters, in both the predicted and ground truth assignments. This is then adjusted for chance such that random assignments of the objects results in a value close to 0. The adjusted rand index is in $[-1, 1]$.

V-Measure

The V-measure [17] is the harmonic mean of the homogeneity and completeness scores. Homogeneity is satisfied if each predicted cluster only contains objects from the same ground truth cluster, and completeness holds if all objects from the same ground truth cluster have been assigned to the same predicted cluster. The V-measure is in $[0, 1]$.

5.1.2 Similarity Matrix

Each experiment specifies the pairwise similarity measure used, and the final matrix is then constructed from Algorithm 6.

Algorithm 6 Canonicalize the similarity matrix by scaling, regularization, and zeroing the main diagonal.

```
1: procedure CANONICALIZE($\mathbf{A}$, $\alpha$) ▷ Pairwise similarities $\mathbf{A}$, regularization parameter $\alpha \geq 0$.
2:     Set the main diagonal of $\mathbf{A}$ to 0
3:     Let $M := \max_{i,j} a_{ij}$
4:     Divide the elements in $\mathbf{A}$ by $M$
5:     Add $\alpha$ to all off-diagonal elements in $\mathbf{A}$
6:     return $\mathbf{A}$
7: end procedure
```

Line 2 ensures the requirements in Definition 3 are satisfied. Note that solutions to (3.2) are invariant to scaling of the objective function by a positive factor, and line 3 does therefore not change the solutions. The regularization parameter α is used to control the cluster sizes, as explained in Section 3.1.3.

5.2 Peeling

All algorithms need to be initiated with a starting point $\mathbf{x}_0$, and how it is selected will likely determine the local optima the algorithms converge to. Let $\bar{\mathbf{x}}^B = \frac{1}{n}\mathbf{e}$ be the barycenter of the simplex, where $\mathbf{e}^T = (1, 1, ..., 1)$, and

$$\begin{cases} \bar{\mathbf{x}}^V & \in \Delta \\ \bar{x}_i^V & = 1, \quad \text{where } i \in \arg\max_i \nabla_i f(\bar{\mathbf{x}}^B) \\ \bar{x}_j^V & = 0, \quad \text{for } j \neq i. \end{cases} \tag{5.2}$$

B for barycenter and V for vertex. The first point $\bar{\mathbf{x}}^B$ avoids initial bias to a particular solution as the weights are uniform. Since $\nabla f(\bar{\mathbf{x}}^B) = 2\mathbf{A}\bar{\mathbf{x}}^B$, the nonzero component of $\bar{\mathbf{x}}^V$ corresponds to the row of $\mathbf{A}$ with largest total sum – that is, it is heavily biased to a node that is highly similar to a lot of other nodes.

The starting point for RD is $\bar{\mathbf{x}}^{RD} = \bar{\mathbf{x}}^B$, as was done in [10]. Note that if a component of $\bar{\mathbf{x}}^{RD}$ starts at zero it will remain at zero for the entire duration of the dynamics according to equation (3.5). Furthermore, $(\bar{\mathbf{x}}^V)^T\mathbf{A}\bar{\mathbf{x}}^V = 0$ since $\mathbf{A}$ has zeros on the main diagonal, and the denominator in (3.5) is thus zero for this point. Therefore $\bar{\mathbf{x}}^V$ is not a viable starting point for RD.

The starting point for standard FW is $\bar{\mathbf{x}}^{FW} = \bar{\mathbf{x}}^{V}$, and was found experimentally to work well. As was shown in Section 4.4, FW never performs any drop steps since the step size always satisfies $\gamma_t < \gamma_{max}$, and using $\bar{\mathbf{x}}^{B}$ as starting point will therefore lead to a solution that has full support; this was found experimentally to hold true as well. Experiment results for PFW and AFW are reported using both $\bar{\mathbf{x}}^{B}$ and $\bar{\mathbf{x}}^{V}$ as starting points. The PFW and AFW variants will be denoted by PFW-B, PFW-V, AFW-B, and AFW-V, respectively, depending on the starting point.

5.2.1 Synthetic Dataset

Here we use a generative approach to examine the clustering quality relative to noise in the dataset. We fix $K = k = 5$, and for a specified n, the objects are uniformly assigned to one of the k clusters.

Let $\mu \sim \mathcal{U}(0,1)$ be uniformly distributed and

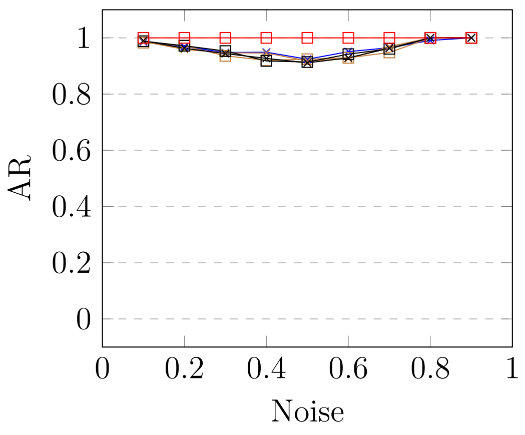
$$\begin{cases} z &= 0, \quad \text{with probability } p \\ z &= 1, \quad \text{with probability } 1-p, \end{cases}$$

where p is the noise ratio. The similarity matrix $\mathbf{A} = (a_{ij})$ is then constructed as follows:

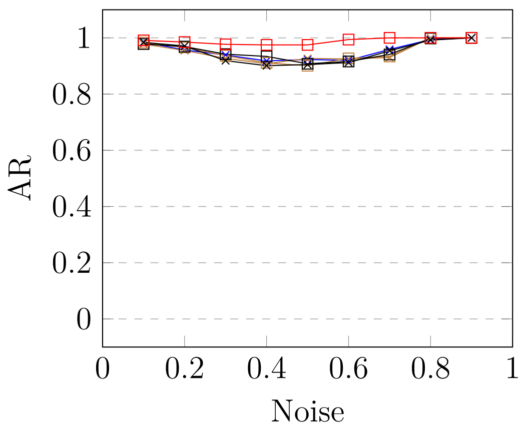
$$\begin{cases} a_{ij} = a_{ji} &= z\mu, \quad \text{if } i \text{ and } j \text{ are in the same cluster} \\ a_{ij} &= 0, \quad \text{otherwise,} \end{cases}$$

The regularization parameter is set to $\alpha = 4$. For each parameter configuration, a similarity matrix is generated and clustered 5 times and the result is then averaged.

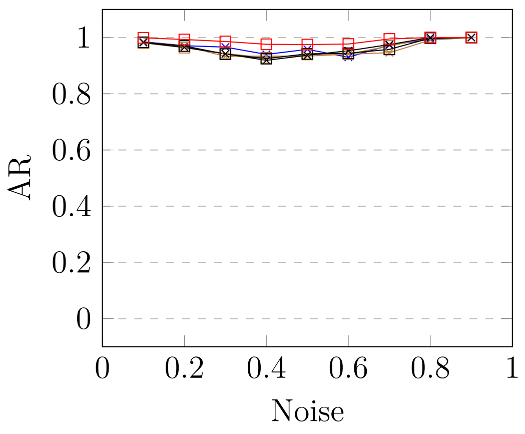
In Figure 5.1 it is evident how sensitive RD is to correct values of the number of iterations t and cutoff δ. The performance of all FW variants is consistent throughout the different parameter configurations, while it varies greatly for RD.


AR
Noise
0
0.2
0.4
0.6
0.8
1

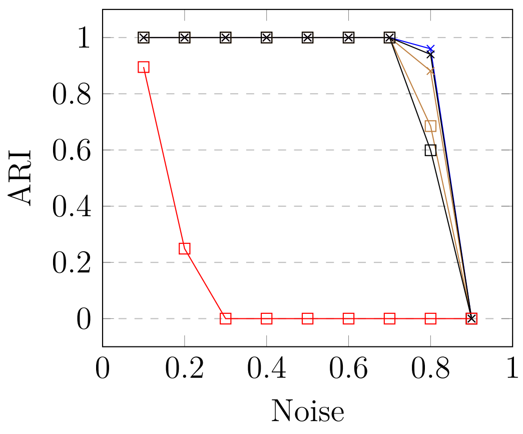
(a)


AR
Noise
0
0.2
0.4
0.6
0.8
1

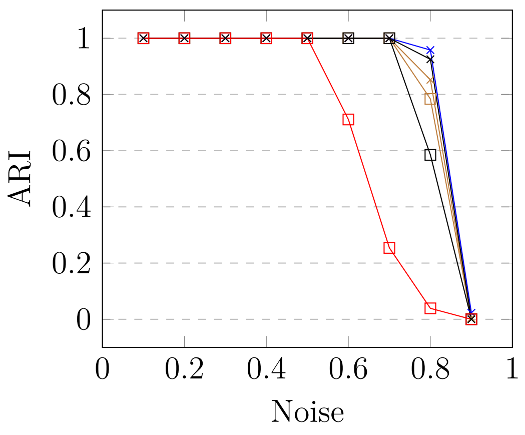
(b)


AR
Noise
0
0.2
0.4
0.6
0.8
1

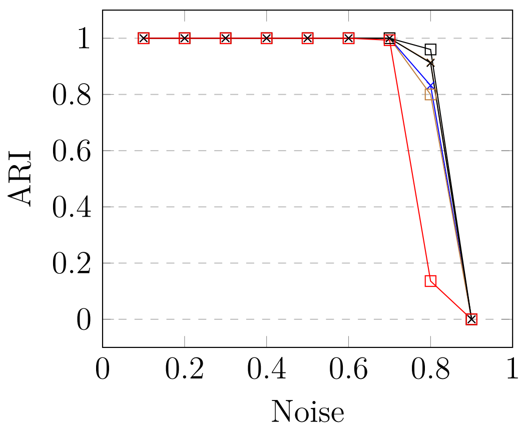
(c)


ARI
Noise
0
0.2
0.4
0.6
0.8
1

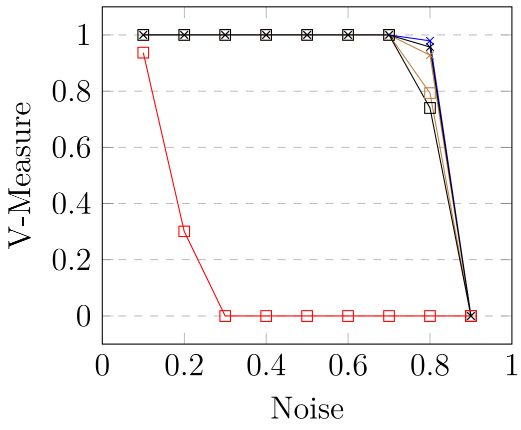
(d)


ARI
Noise
0
0.2
0.4
0.6
0.8
1

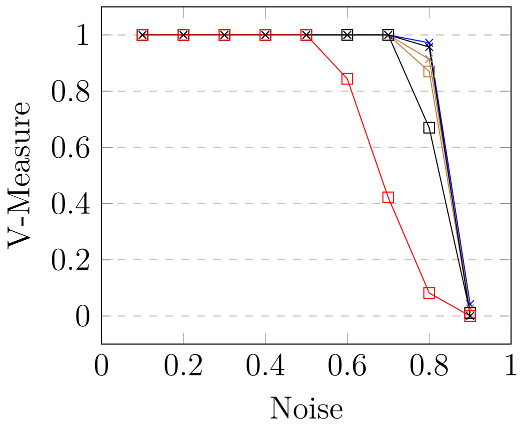
(e)


ARI
Noise
0
0.2
0.4
0.6
0.8
1

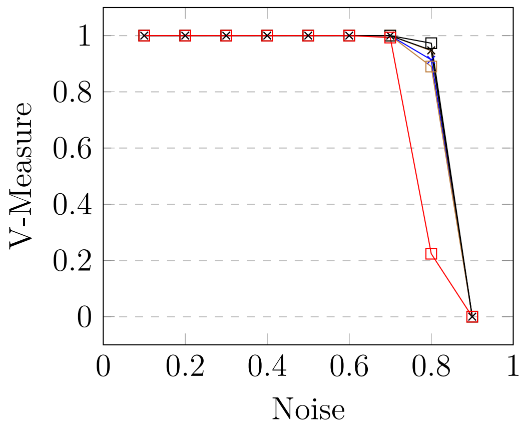
(f)



(g)



(h)



(i)

Figure 5.1: Synthetic dataset results for RD, FW, PFW, and AFW. PFW-B and AFW-B have squares; PFW-V and AFW-V have crosses. (a)-(i) $n = 200$. (a,d,g) $t = 400$, $\delta = 2 \cdot 10^{-12}$. (b,e,h) $t = 400$, $\delta = 2 \cdot 10^{-3}$. (c,f,i) $t = 4000$, $\delta = 2 \cdot 10^{-12}$.

5.2.2 Newsgroups Dataset

In this section we study the clustering quality on the newsgroups dataset[1] from scikit-learn [18], similar as [6, 19]. The dataset consists of 18000 newsgroups posts on 20 categories split into subsets for training and test. The test subset was used for our experiments, and it was further split into 4 smaller datasets based on randomizing the categories.

- newsgroups1: soc.religion.christian, comp.os.ms-windows.misc, talk.politics.guns, alt.atheism, talk.politics.misc.
- newsgroups2: comp.windows.x, sci.med, rec.autos, talk.religion.misc, sci.crypt.
- newsgroups3: misc.forsale, comp.sys.mac.hardware, talk.politics.mideast, sci.electronics, rec.motorcycles.

[1]newsgroups dataset

- newsgroups4: comp.sys.ibm.pc.hardware, comp.graphics, rec.sport.hockey, rec.sport.baseball, sci.space.

The text posts are converted to tf-idf[2] vectors and then further reduced in dimension using PCA[3] with 20 components. The similarity matrix $\mathbf{A}$ is constructed using the cosine similarity[4] and then the off-diagonal elements are shifted by 1 to ensure nonnegative entries.

Each of the datasets in the list have $k = 5$ clusters and $1700 \leq n \leq 2000$ posts, and $K = 5$ is also used for peeling.

The regularization parameter was set to $\alpha = 15$, and appeared to be a reasonable compromise between the two methods. Using smaller or larger α resulted in too small clusters or too slow convergence, respectively, for replicator dynamics.

Tables 5.1, 5.3, 5.5, and 5.7 show the results for the different datasets. Since all objects in the ground truth dataset are assigned to a cluster, the assignment rate is the ratio of objects that were selected during the clustering. High AR and low ARI/V-measure in this context means that many objects have been assigned to the same cluster. This is what happens for RD with $t = 1000$ – the progress is insufficient and all objects are therefore considered part of the same support/cluster. The results are fairly consistent with the synthetic dataset results: the FW variants are mostly similar across the different number of iterations while there is more variance for RD. For $t = 8000$ we also see a significant speed difference; see [7] for additional speed comparisons between RD and AFW/InImDyn.

Tables 5.2, 5.4, 5.6, and 5.8 show corresponding results after a post-processing step. The post-processing step assigns each unassigned object to the cluster which it has highest average similarity with. Specifically, let $C_0 \subseteq V$ be the unassigned objects and $C_i \subseteq V$, $1 \leq i \leq K$, the predicted clusters. Object $j \in C_0$ is then assigned to cluster C_i that satisfies

$$i \in \arg\max_{\ell \geq 1} \frac{1}{|C_\ell|} \sum_{p \in C_\ell} a_{jp}.$$

The assignment rate is always 1 after the post-processing step since all objects have now been assigned to a cluster. From the tables we see that ARI and V-measure are mostly similar as before and after the post-processing, but there is a complete partitioning of the data in the latter case.

[2]term-frequency times inverse document-frequency

[3]principal component analysis

[4]cosine similarity

	t	Duration	AR	ARI	V-Measure
FW	1000	0.36s	0.6325	0.4695	0.5388
	4000	1.35s	0.6885	0.4593	0.5224
	8000	2.41s	0.6969	0.4673	0.5325
PFW-B	1000	0.43s	0.7429	0.1944	0.4289
	4000	1.86s	0.6605	0.467	0.5327
	8000	2.62s	0.642	0.471	0.5335
PFW-V	1000	0.52s	0.6471	0.5178	0.5745
	4000	1.6s	0.6487	0.4565	0.5237
	8000	2.47s	0.642	0.471	0.5335
AFW-B	1000	0.35s	0.8527	0.076	0.2854
	4000	1.69s	0.6258	0.3887	0.5316
	8000	2.93s	0.6599	0.4676	0.5328
AFW-V	1000	0.46s	0.6415	0.5184	0.5736
	4000	1.38s	0.6482	0.518	0.5754
	8000	2.75s	0.6476	0.4618	0.5257
RD	1000	1.06s	1.0	0.0	0.0
	4000	4.56s	0.9081	0.1852	0.3003
	8000	11.4s	0.6997	0.4121	0.5384

Table 5.1: Dataset newsgroups1 results.

	AR	ARI	V-Measure
FW	1.0	0.4068	0.4969
	1.0	0.4639	0.5225
	1.0	0.4766	0.5351
PFW-B	1.0	0.2063	0.3919
	1.0	0.4623	0.5324
	1.0	0.4356	0.5219
PFW-V	1.0	0.5091	0.5763
	1.0	0.4298	0.5149
	1.0	0.4356	0.5219
AFW-B	1.0	0.0966	0.2751
	1.0	0.3162	0.4806
	1.0	0.4615	0.5319
AFW-V	1.0	0.5066	0.5744
	1.0	0.5047	0.5719
	1.0	0.4308	0.5148
RD	1.0	0.0	0.0
	1.0	0.1892	0.3042
	1.0	0.3659	0.4937

Table 5.2: Dataset newsgroups1 results after post-processing.

	t	Duration	AR	ARI	V-Measure
	1000	0.37s	0.6587	0.5594	0.5929
FW	4000	1.38s	0.6674	0.5479	0.5866
	8000	2.6s	0.6679	0.5473	0.5864
	1000	0.45s	0.7508	0.135	0.3555
PFW-B	4000	1.57s	0.6172	0.6257	0.6364
	8000	2.06s	0.6172	0.6257	0.6364
	1000	0.59s	0.6281	0.6095	0.6241
PFW-V	4000	1.85s	0.6172	0.6257	0.6364
	8000	3.1s	0.6172	0.6257	0.6364
	1000	0.41s	0.8653	0.0979	0.316
AFW-B	4000	1.9s	0.6172	0.6257	0.6364
	8000	3.39s	0.6172	0.6257	0.6364
	1000	0.48s	0.663	0.5548	0.5907
AFW-V	4000	1.75s	0.6172	0.6257	0.6364
	8000	3.38s	0.6172	0.6257	0.6364
	1000	0.76s	1.0	0.0	0.0
RD	4000	4.67s	1.0	0.1795	0.333
	8000	13.52s	0.7585	0.4391	0.5161

Table 5.3: Dataset newsgroups2 results.

	AR	ARI	V-Measure
	1.0	0.5158	0.5733
FW	1.0	0.5084	0.5699
	1.0	0.5084	0.5699
	1.0	0.178	0.3814
PFW-B	1.0	0.5332	0.5834
	1.0	0.5332	0.5834
	1.0	0.5331	0.5824
PFW-V	1.0	0.5332	0.5834
	1.0	0.5332	0.5834
	1.0	0.131	0.344
AFW-B	1.0	0.5332	0.5834
	1.0	0.5332	0.5834
	1.0	0.5099	0.5699
AFW-V	1.0	0.5332	0.5834
	1.0	0.5332	0.5834
	1.0	0.0	0.0
RD	1.0	0.1795	0.333
	1.0	0.4123	0.5065

Table 5.4: Dataset newsgroups2 results after post-processing.

	t	Duration	AR	ARI	V-Measure
FW	1000	0.41s	0.6756	0.5206	0.5879
	4000	1.35s	0.6468	0.5309	0.5975
	8000	2.63s	0.6473	0.5314	0.5978
PFW-B	1000	0.49s	0.758	0.217	0.4617
	4000	1.79s	0.6468	0.5317	0.6004
	8000	2.88s	0.6468	0.5317	0.6004
PFW-V	1000	0.56s	0.6468	0.5317	0.6004
	4000	1.96s	0.6468	0.5317	0.6004
	8000	3.71s	0.6468	0.5317	0.6004
AFW-B	1000	0.37s	0.8373	0.1381	0.3594
	4000	1.83s	0.6462	0.5316	0.6003
	8000	3.19s	0.6468	0.5317	0.6004
AFW-V	1000	0.49s	0.6468	0.5322	0.5993
	4000	1.63s	0.6468	0.5317	0.6004
	8000	2.99s	0.6468	0.5317	0.6004
RD	1000	0.86s	1.0	0.0	0.0
	4000	4.69s	0.9089	0.2212	0.3465
	8000	12.9s	0.8012	0.3526	0.4556

Table 5.5: Dataset newsgroups3 results.

	AR	ARI	V-Measure
FW	1.0	0.5751	0.595
	1.0	0.572	0.5962
	1.0	0.5729	0.5972
PFW-B	1.0	0.2764	0.4859
	1.0	0.5734	0.5992
	1.0	0.5734	0.5992
PFW-V	1.0	0.5734	0.5992
	1.0	0.5734	0.5992
	1.0	0.5734	0.5992
AFW-B	1.0	0.1782	0.3967
	1.0	0.5734	0.5992
	1.0	0.5734	0.5992
AFW-V	1.0	0.5741	0.5983
	1.0	0.5734	0.5992
	1.0	0.5734	0.5992
RD	1.0	0.0	0.0
	1.0	0.2394	0.3575
	1.0	0.4227	0.4973

Table 5.6: Dataset newsgroups3 results after post-processing.

	t	Duration	AR	ARI	V-Measure
FW	1000	0.42s	0.653	0.5097	0.5706
	4000	1.38s	0.6169	0.4672	0.5437
	8000	2.6s	0.7002	0.5014	0.5514
PFW-B	1000	0.43s	0.8092	0.2247	0.4483
	4000	1.82s	0.6697	0.6211	0.6484
	8000	3.04s	0.6697	0.6211	0.6484
PFW-V	1000	0.58s	0.6591	0.6446	0.6717
	4000	2.02s	0.6565	0.6462	0.675
	8000	2.74s	0.6565	0.6462	0.675
AFW-B	1000	0.35s	0.9041	0.1109	0.3361
	4000	1.87s	0.6687	0.6191	0.6463
	8000	3.55s	0.6697	0.6211	0.6484
AFW-V	1000	0.5s	0.6525	0.5071	0.5651
	4000	1.84s	0.6565	0.6462	0.675
	8000	3.6s	0.6565	0.6462	0.675
RD	1000	0.93s	1.0	0.0	0.0
	4000	5.46s	1.0	0.3197	0.4112
	8000	14.52s	0.8559	0.4528	0.5328

Table 5.7: Dataset newsgroups4 results.

	AR	ARI	V-Measure
FW	1.0	0.4663	0.5252
	1.0	0.4409	0.5084
	1.0	0.4973	0.5396
PFW-B	1.0	0.288	0.4878
	1.0	0.587	0.6094
	1.0	0.587	0.6094
PFW-V	1.0	0.605	0.6226
	1.0	0.6072	0.6268
	1.0	0.6072	0.6268
AFW-B	1.0	0.1313	0.3577
	1.0	0.588	0.6097
	1.0	0.587	0.6094
AFW-V	1.0	0.4592	0.5183
	1.0	0.6072	0.6268
	1.0	0.6072	0.6268
RD	1.0	0.0	0.0
	1.0	0.3197	0.4112
	1.0	0.4858	0.5556

Table 5.8: Dataset newsgroups4 results after post-processing.

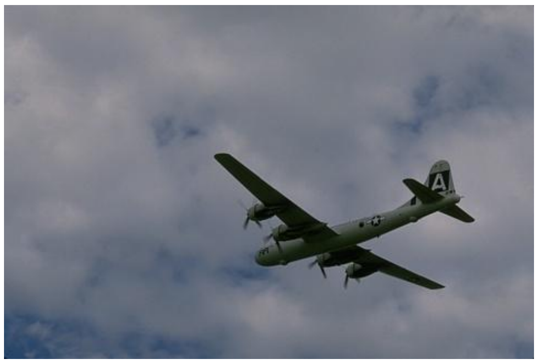


Figure 5.2: Original image with dimensions 120×80.

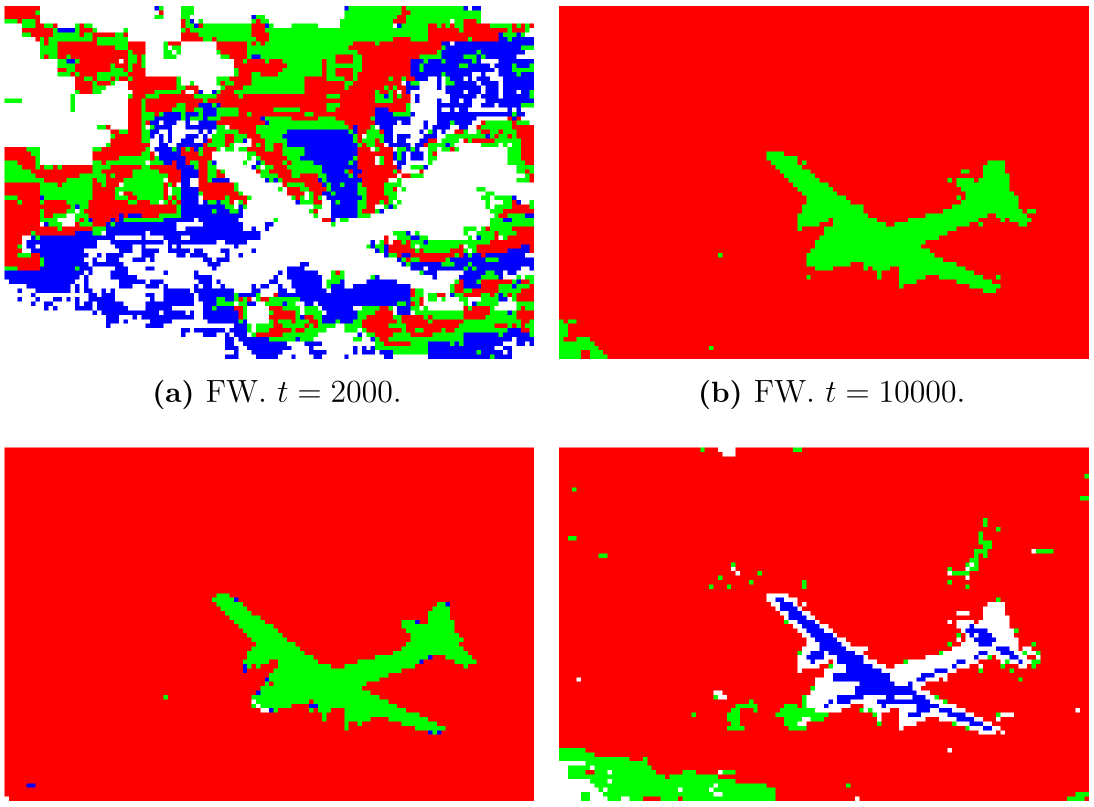
5.2.3 Image Segmentation

We consider segmentation of colored images in HSV space. Define feature vector $\mathbf{f}(i) = [v, vs\sin(h), vs\cos(h)]^T$ as in [2], where h, s, and v are the HSV values of pixel i. The similarity matrix $\mathbf{A}$ is then defined as follows:

1. Compute $||\mathbf{f}(i) - \mathbf{f}(j)||$, for every pair of pixels i, j, into matrix $\mathbf{D}^{L2}$.
2. Compute path-based distances [20, 21] from matrix $\mathbf{D}^{L2}$ into matrix $\mathbf{D}^{P}$.
3. Finally, $\mathbf{A} = \max(\mathbf{D}^{P}) - \mathbf{D}^{P}$, where max is over the elements in $\mathbf{D}^{P}$.

Figures 5.3, 5.4, and 5.5 show segmentation results of the airplane in Figure 5.2. An image with dimensions 120×80 results in an clustering problem with $n = 120 \cdot 80 = 9600$. The experiments extracted max $K = 3$ clusters. All FW variants used $\alpha = 1500$ and RD used $\alpha = 0$. Other α values for RD resulted in too slow progress.

When starting the FW variants with point 5.2, at least $t = \ell$ iterations are needed in order to extract a cluster with size ℓ. This explains the poor results for $t = 2000$ when using this starting point. The barycenter starting point for PFW and AFW is therefore better in this case, as it is more efficient to remove components from the support instead of adding to it. Furthermore, while the results looks reasonable in Figure 5.3c with $t = 50$ iterations, RD has not actually converged as can be seen in Figure 5.3d with $t = 250$ iterations. The results deteriorate due to the small value of α – the clusters are too small and can therefore not represent the complete airplane/background. Also note that running RD for 50 iterations takes about the same time as running the FW variants for 10000 iterations, due to $n = 9600$ and their quadratic and linear per-iteration time complexities.

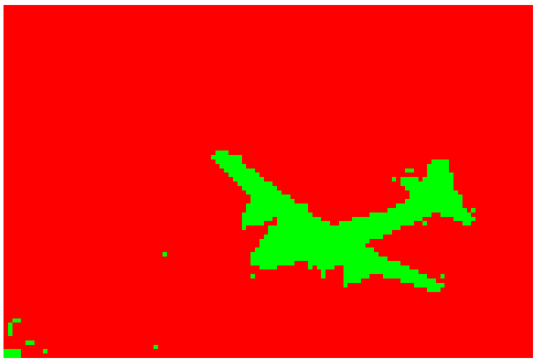

(a) FW. $t = 2000$.

(b) FW. $t = 10000$.

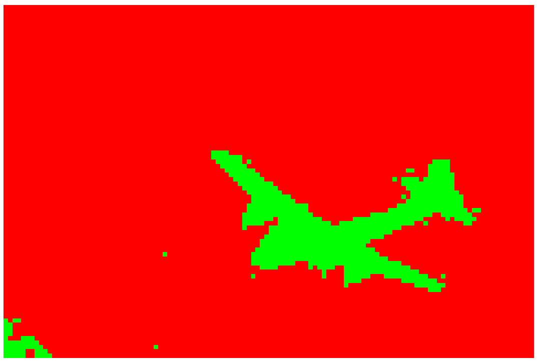
(c) RD. $t = 50$.

(d) RD. $t = 250$.

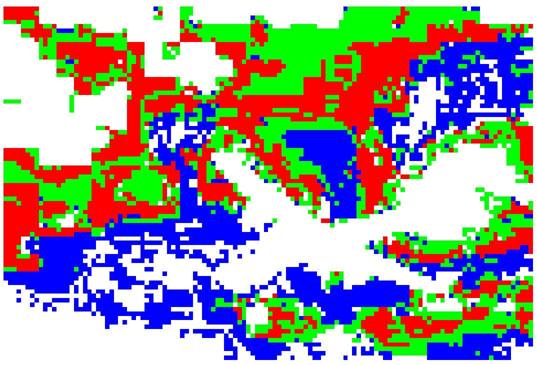
Figure 5.3: Image segmentation results for standard FW and RD.



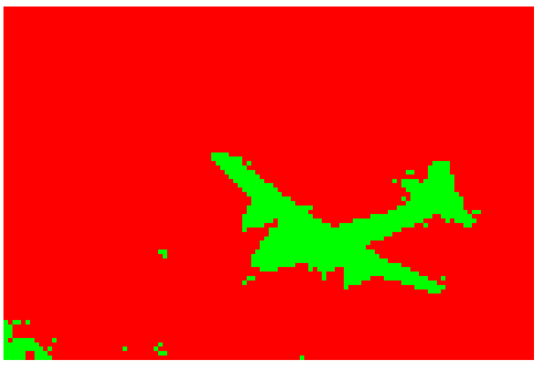
(a) PFW-B. $t = 2000$.



(b) PFW-B. $t = 10000$.

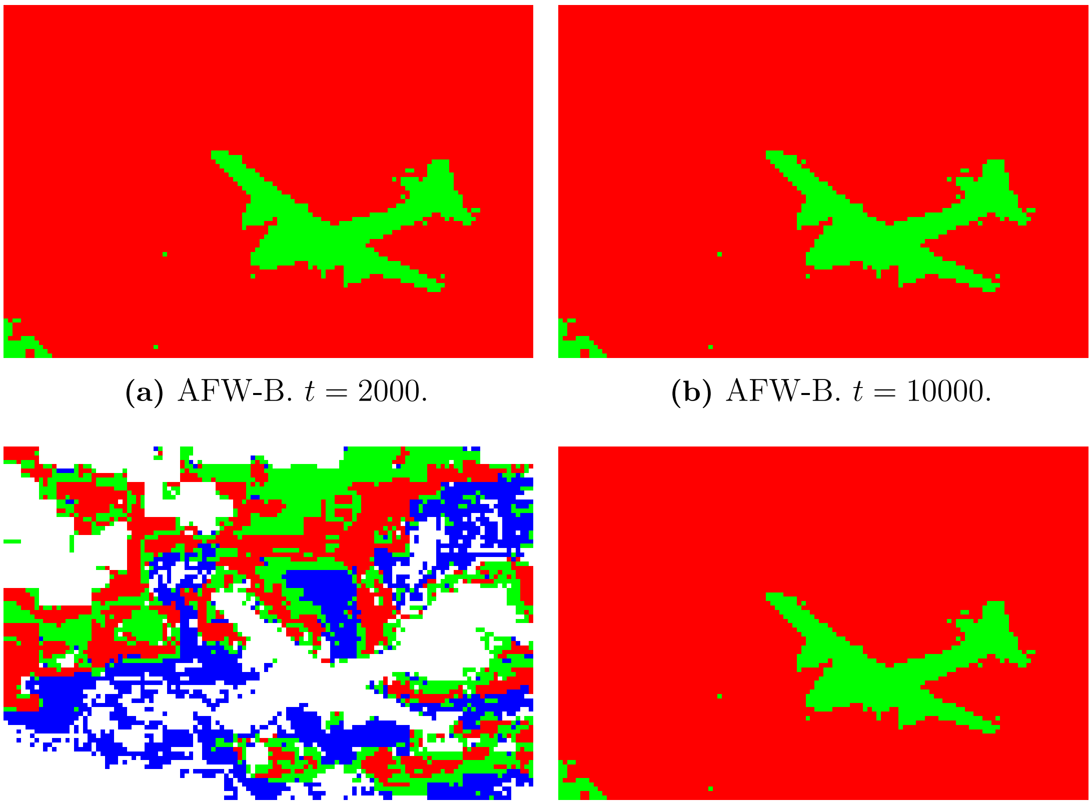


(c) PFW-V. $t = 2000$.



(d) PFW-V. $t = 10000$.

Figure 5.4: Image segmentation results for PFW.



(a) AFW-B. $t = 2000$.

(b) AFW-B. $t = 10000$.

(c) AFW-V. $t = 2000$.

(d) AFW-V. $t = 10000$.

Figure 5.5: Image segmentation results for AFW.

5.3 Multistart

This experiment considers a combination of multistart with peeling, and it works as follows:

1. Sample a subset of objects, and based on them, construct a number of starting points for the same similarity matrix.
2. Given a solver, run it for each starting point.
3. Identify the nonoverlapping clusters from the solutions and remove corresponding objects from the similarity matrix as in peeling.
4. Repeat until no objects are left or sufficient number of clusters have been found.

This is interesting to investigate because the solver can be run parallel and in isolation with different starting points. However, if they all converge to the same cluster, then there is no benefit and it is usually worse than normal peeling due to the extra overhead from the setup. Though this experiment will not consider speed per se since Python is not very efficient when it comes to multiprocessing. Instead we will consider the number of passes through the data, where a pass is one run through the items in the list above.

After the solutions have been computed, they are sorted based on the function value $f(\mathbf{x})$. The sorted solutions are then traversed in a decreasing order, and if the support of the current solution overlaps more than ten percent with the support of previous solutions, it is discarded. Each pass will therefore extract at least one cluster, and if K is the maximum number of clusters to extract, there will be at most K passes. Thus in order for the method to be useful, less than K passes should be performed.

Figure 5.6 shows the type of dataset that is used. Each cluster is a 2D Gaussian with fixed mean and standard deviation 1 – see Figure 5.6a. Parameter p controls the noise ratio, and we fix $n = 1000$ and $K = k = 4$. Using $n_1 = pn$ and $n_2 = n - n_1$, a dataset is generated by sampling $0.1 \cdot n_2$, $0.2 \cdot n_2$, $0.3 \cdot n_3$, and $0.4 \cdot n_2$ points from the respective Gaussian, and n_1 points from a uniform distribution (the clutter in Figure 5.6c).

Let $\mathbf{D}$ be the matrix with pairwise Euclidean distances between all points in the dataset. The similarity matrix is then defined as $\mathbf{A} = \max(\mathbf{D}) - \mathbf{D}$, similar as in the image segmentation problem but with different distance measure. The regularization parameter is set to $\alpha = 15$.

To determine the starting points we sample 4 components from $\{1, ..., n\}$, denote them by i_1, i_2, i_3, i_4. The number 4 was chosen since it matches the number of CPUs on our system. Using similar notation as in Section 5.2, for given component $i \in \{i_1, i_2, i_3, i_4\}$, the starting points are defined as

$$\begin{cases} \bar{x}_i^V &= 1 \\ \bar{x}_j^V &= 0, \quad \text{for } j \neq i \end{cases}$$

and

$$\begin{cases} \bar{x}_i^B & = 0.5 \\ \bar{x}_j^B & = 0.5/(n-1), \qquad \text{for } j \neq i. \end{cases}$$

FW uses only $\bar{\mathbf{x}}^V$ while PFW and AFW use both $\bar{\mathbf{x}}^V$ and $\bar{\mathbf{x}}^B$.

The last thing remaining is the sampling methods for the components. For this we consider uniform sampling and Determinantal Point Processes [22], denoted UNI and DPP, respectively.

Uniform Sampling

Let ℓ be the number of components to sample and $a_{i*} = \sum_{j=1}^{n} a_{ij}$, the sum of the elements in row i of $\mathbf{A}$. Sort a_{i*} in decreasing order, divide them into blocks of size n/ℓ, and sample one component i uniformly from each block.

Determinantal Point Processes

A discrete DPP is a probability measure on subsets of a discrete set V, that is, on the power set 2^V. We consider a variant of DPP called L-ensemble. If $\mathbb{P}_{\mathbf{L}}$ is an L-ensemble and $Y \subseteq V$, the probability of sampling it according to $\mathbb{P}_{\mathbf{L}}$ then is

$$\mathbb{P}_{\mathbf{L}}(Y) \propto \det(\mathbf{L}_Y),$$

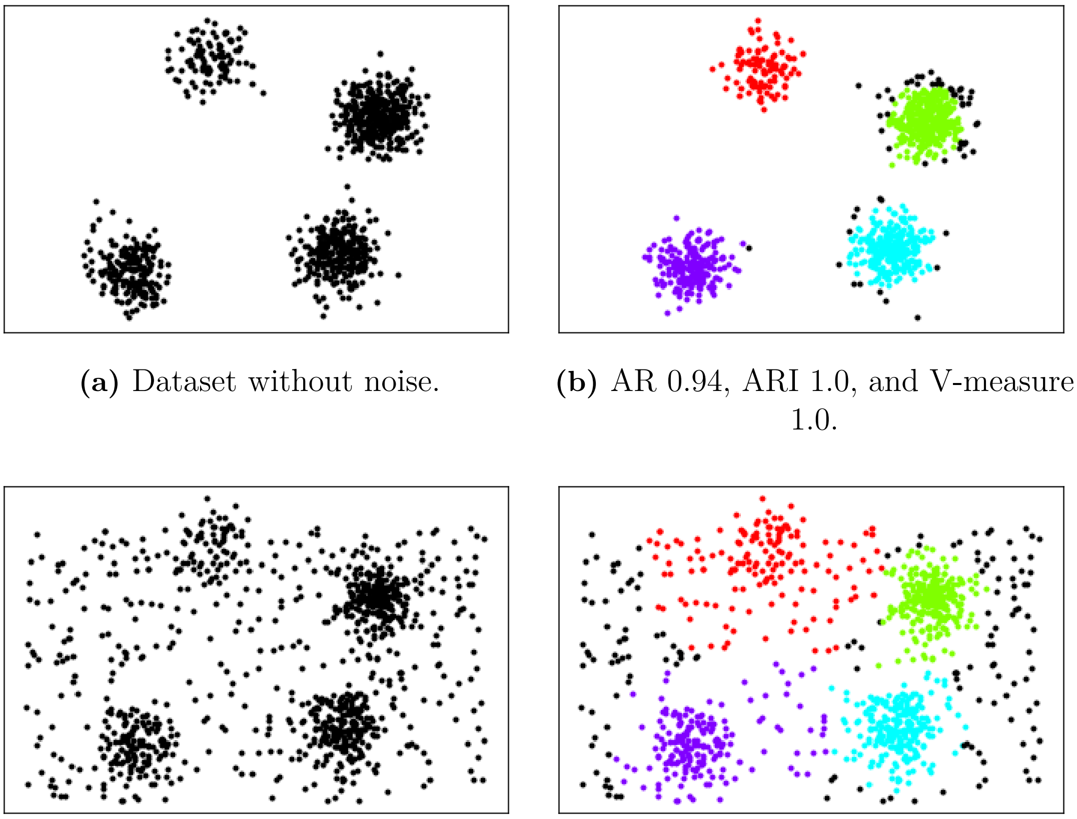
where $\mathbf{L}$ is real, symmetric, and positive semidefinite, called the likelihood matrix. The submatrix $\mathbf{L}_Y$ are the rows and columns of $\mathbf{L}$ indexed by Y [22]. If $\mathbf{L}$ is a similarity matrix, then subsets with diverse objects (low similarities between them) are more likely to be sampled. For example, if $Y = \{i, j\}$, then

$$\mathbb{P}_{\mathbf{L}}(Y) \propto \det(\mathbf{L}_Y) = \ell_{ii}\ell_{jj} - \ell_{ij}\ell_{ji}. \tag{5.3}$$

DPP is thus a reasonable sampling method for our purposes. Using the similarity matrix $\mathbf{A}$ as the DPP likelihood matrix, we are more likely to sample objects that are diverse, and therefore less likely to belong in the same cluster.

The similarity matrix $\mathbf{A}$ – or submatrices thereof – are real and symmetric, but not positive semidefinite since they have zeros on the main diagonal. However, any symmetric matrix can be transformed to be positive semidefinite by ensuring it is diagonally dominant – every diagonal element is larger than the sum of the absolute elements on the corresponding row.

In order to sample from a DPP with given likelihood matrix, we need to compute its eigendecomposition which is an $\mathcal{O}(n^3)$ operation. The sampling is therefore done in two steps. The first step is similar to the uniform sampling method: sort the components based on a_{i*}, split them into blocks of size $n/10$, and then uniformly sample $n^{2/3}/10$ objects from each block. In the second step we sample with DPP using as likelihood matrix the submatrix of $\mathbf{A}$ indexed by the $n^{2/3}$ objects from the first step. Note we ensure the submatrix of $\mathbf{A}$ is diagonally dominant instead of $\mathbf{A}$. In our experiment we used the Python library DPPy [23] to sample from a k-DPP, which ensures that the sampled set has k components (ℓ in our case).



(a) Dataset without noise.

(b) AR 0.94, ARI 1.0, and V-measure 1.0.

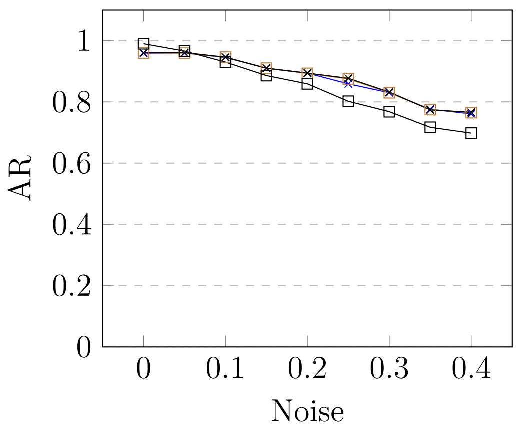
(c) Dataset with noise $p = 0.4$.

(d) AR 0.74, ARI 1.0, and V-measure 1.0.

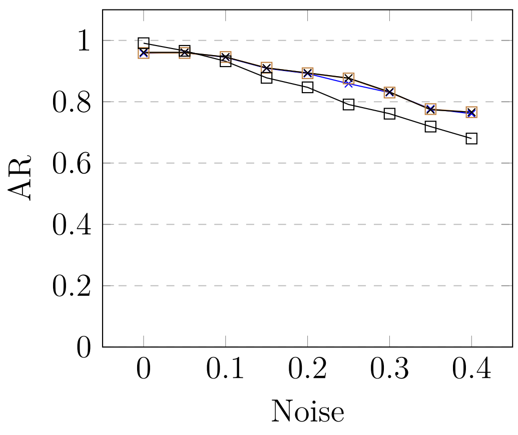
Figure 5.6: Sample datasets. (b) and (d) show clustering results from FW with peeling; PFW and AFW produce similar results.

Results

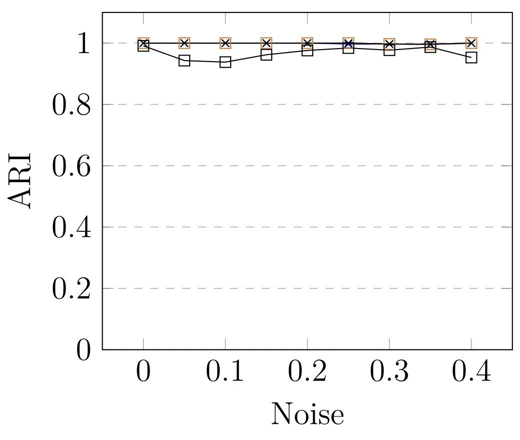
Figure 5.7 shows the experiment results for the different types of sampling methods and starting points. For a given dataset, sampling method, and solver, we generate starting points and run the experiment 10 times and report the average results. Each solver is run for $t = 1000$ iterations. For this type of dataset we do not see any significant difference between either FW, PFW, or AFW when using either DPP or UNI. The most noticeable effect is that AFW with $\bar{\mathbf{x}}^B$ as starting point seems to perform slightly worse.


AR
Noise
0
0.2
0.4
0.6
0.8
1
0
0.1
0.2
0.3
0.4

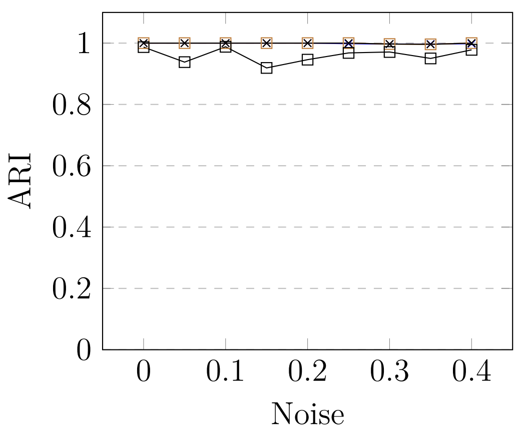
(a)


AR
Noise
0
0.2
0.4
0.6
0.8
1
0
0.1
0.2
0.3
0.4

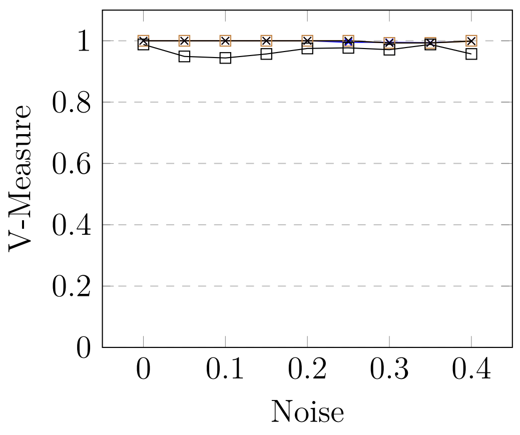
(b)


ARI
Noise
0
0.2
0.4
0.6
0.8
1
0
0.1
0.2
0.3
0.4

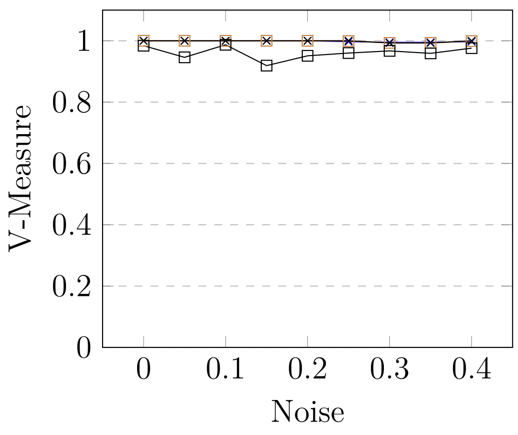
(c)


ARI
Noise
0
0.2
0.4
0.6
0.8
1
0
0.1
0.2
0.3
0.4

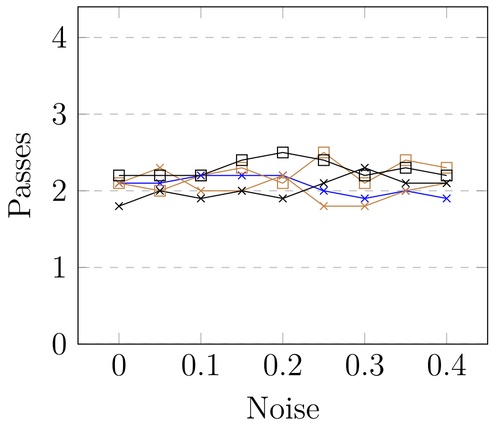
(d)


V-Measure
Noise
0
0.2
0.4
0.6
0.8
1
0
0.1
0.2
0.3
0.4

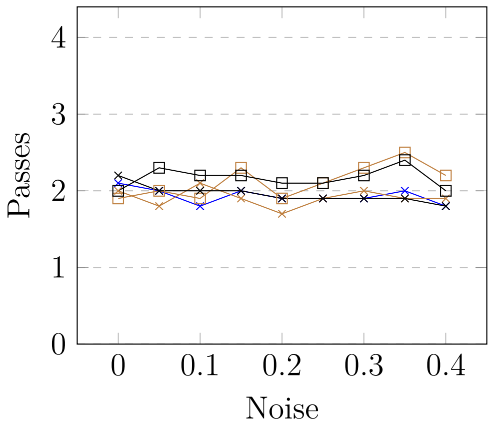
(e)


V-Measure
Noise
0
0.2
0.4
0.6
0.8
1
0
0.1
0.2
0.3
0.4

(f)



(g)



(h)

Figure 5.7: Multistart dataset results for FW, PFW, and AFW. PFW-B and AFW-B have squares; PFW-V and AFW-V have crosses. Left column (a,c,e,g) DPP and right column (b,d,f,h) UNI.

6 Conclusion

6.1 Discussion

In section 4.4 we have shown that all three Frank-Wolfe variants converge at rate $\mathcal{O}(1/\sqrt{t})$ in terms of the Frank-Wolfe gap. The bounds are tight for standard FW and AFW, while it is looser for PFW due to the unfortunate factorial in the numerator. In practice, however, they all appear to exhibit similar convergence, based on the experiment results in the previous chapter. An advantage with PFW and AFW over standard FW is that they can also use the barycenter as starting point. This was seen to be useful in the image segmentation experiment, as it took 2000 iterations to achieve the same result as standard FW did for 10000 iterations.

We reaffirm the conclusions from [6, 3] regarding replicator dynamics. That is, it is very sensitive to the number of iterations and the cutoff parameter, as can be seen in all results from the peeling experiments. The FW variants are more stable with respect to the cutoff parameter, and generally less sensitive to the number of iterations; a special case is when using a vertex as starting point – the FW variants need to run at least ℓ iterations in order to extract a cluster of size ℓ.

A general observation related to Dominant Set Clustering, which has not been emphasized in prior works, is the difficulty in constructing the similarity matrix for a given problem. As can be seen in the newsgroups experiment, the assignment rate is below 1 by a significant margin. The ground truth have all objects assigned to a cluster, but the predicted clusters contain fewer objects and some are therefore left unassigned. In other words, the occurrence of small dominant sets that do not encompass the complete cluster cause the reduced assignment rate. This is the type of problem the regularization parameter is made to solve – remove too small dominant sets so that bigger appear – but it is difficult to predict their size, and therefore the regularization parameter.

In addition to the regularization parameter, the post-processing step used in the newsgroups experiment is an easy and efficient heuristic that enlarges the clusters such that they cover the complete dataset. However, if the dataset includes objects that do not belong to a cluster, e.g. noise, then additional care has to be taken such that they are still left unassigned after the post-processing.

Finally, the multistart experiment showed some interesting results. DPP has a lot more overhead compared to uniform sampling, but did not provide any noticeable benefit in the number of passes needed. One explanation for this could be that the diagonal elements are a lot larger compared to the off-diagonal elements, due to ensuring the similarity matrix is diagonally dominant. This potentially eliminates a lot of the benefits of DPP as less emphasis is put on the off-diagonal elements, see Equation 5.3. It would be interesting to investigate other ways to ensure positive semidefiniteness of the similarity matrix, as well as further experiments with multistart and an efficient parallel implementation to investigate speedups relative to traditional peeling.

6.2 Ethical Considerations

The datasets used in the experiments are either synthetic or publicly available. This means there are no immediate privacy concerns or sensitive information that need to be accounted for. However, this may not be the case more generally and is therefore important to consider. A second point to consider is that the clusters are completely determined by how the similarity matrix is defined. If there are biases in the dataset that are reflected by the similarity matrix, the final clustering result is then likely to also be biased. One should thus be conscious of the type of data being analyzed.

6.3 Conclusion

We have studied variants of the Frank-Wolfe algorithm – standard FW, pairwise FW, and away-steps FW – in the context of the StQP defined by Dominant Set Clustering. AFW is equivalent to the infection and immunization dynamics, which has previously been applied to DSC as an alternative to replicator dynamics. We find that all FW variants yield solutions with similar quality, and on par or better than RD. The linear vs quadratic per-iteration time complexity for the FW variants and RD, respectively, is also an immediate advantage. Finally, PFW and AFW have the benefit over standard FW in that they are able to be started from the barycenter, which can be useful for problems with large clusters.

Bibliography

[1] Anil K Jain. Data clustering: 50 years beyond k-means. *Pattern recognition letters*, 31(8):651–666, 2010.

[2] Massimiliano Pavan and Marcello Pelillo. Dominant sets and pairwise clustering. *IEEE transactions on pattern analysis and machine intelligence*, 29(1):167–172, 2006.

[3] Samuel Rota Bulò and Marcello Pelillo. Dominant-set clustering: A review. *European Journal of Operational Research*, 262(1):1–13, 2017.

[4] Simon Lacoste-Julien and Martin Jaggi. On the global linear convergence of frank-wolfe optimization variants. In *Advances in Neural Information Processing Systems*, pages 496–504, 2015.

[5] Massimiliano Pavan and Marcello Pelillo. Dominant sets and hierarchical clustering. In *Proceedings of the Ninth IEEE International Conference on Computer Vision*, volume 2, pages 362–. IEEE, 2003.

[6] Morteza Haghir Chehreghani. Adaptive trajectory analysis of replicator dynamics for data clustering. *Machine Learning*, 104(2-3):271–289, 2016.

[7] Samuel Rota Bulò, Marcello Pelillo, and Immanuel M Bomze. Graph-based quadratic optimization: A fast evolutionary approach. *Computer Vision and Image Understanding*, 115(7):984–995, 2011.

[8] Martin Ester, Hans-Peter Kriegel, Jörg Sander, Xiaowei Xu, et al. A density-based algorithm for discovering clusters in large spatial databases with noise. In *Kdd*, volume 96, pages 226–231, 1996.

[9] Jianbo Shi and Jitendra Malik. Normalized cuts and image segmentation. *Departmental Papers (CIS)*, page 107, 2000.

[10] Massimiliano Pavan and Marcello Pelillo. A new graph-theoretic approach to clustering and segmentation. In *2003 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 2003. Proceedings.*, volume 1, pages I–I. IEEE, 2003.

[11] Marguerite Frank and Philip Wolfe. An algorithm for quadratic programming.

Naval research logistics quarterly, 3(1-2):95–110, 1956.

[12] Simon Lacoste-Julien. Convergence rate of frank-wolfe for non-convex objectives. *arXiv preprint arXiv:1607.00345*, 2016.

[13] Immanuel M Bomze, Francesco Rinaldi, and Damiano Zeffiro. Active set complexity of the away-step frank-wolfe algorithm. *arXiv preprint arXiv:1912.11492*, 2019.

[14] Immanuel M Bomze, Francesco Rinaldi, and Samuel Rota Bulò. First-order methods for the impatient: Support identification in finite time with convergent frank–wolfe variants. *SIAM Journal on Optimization*, 29(3):2211–2226, 2019.

[15] Travis E Oliphant. *A guide to NumPy*, volume 1. Trelgol Publishing USA, 2006.

[16] Lawrence Hubert and Phipps Arabie. Comparing partitions. *Journal of classification*, 2(1):193–218, 1985.

[17] Andrew Rosenberg and Julia Hirschberg. V-measure: A conditional entropy-based external cluster evaluation measure. In *Proceedings of the 2007 joint conference on empirical methods in natural language processing and computational natural language learning (EMNLP-CoNLL)*, pages 410–420, 2007.

[18] Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, et al. Scikit-learn: Machine learning in python. *Journal of machine learning research*, 12(Oct):2825–2830, 2011.

[19] Erik Thiel, Morteza Haghir Chehreghani, and Devdatt Dubhashi. A non–convex optimization approach to correlation clustering. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 5159–5166, 2019.

[20] Bernd Fischer and Joachim M. Buhmann. Path-based clustering for grouping of smooth curves and texture segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 25(4):513–518, 2003.

[21] Morteza Haghir Chehreghani. Classification with minimax distance measures. In *Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence*, pages 1784–1790. AAAI Press, 2017.

[22] Alex Kulesza, Ben Taskar, et al. Determinantal point processes for machine learning. *Foundations and Trends® in Machine Learning*, 5(2–3):123–286, 2012.

[23] Guillaume Gautier, Guillermo Polito, Rémi Bardenet, and Michal Valko. DPPy: DPP Sampling with Python. *Journal of Machine Learning Research - Machine Learning Open Source Software (JMLR-MLOSS)*, 2019. Code at http://github.com/guilgautier/DPPy/ Documentation at http://dppy.readthedocs.io/.