# Semi-Supervised Max-Sum Clustering

Konstantin Voevodski
Google, Cambridge, USA
kvodski@google.com

## ABSTRACT

We study max-sum clustering in a semi-supervised setting. Our objective function maximizes the pairwise within-cluster similarity with respect to some null hypothesis regarding the similarity. This is a natural objective that does not require any additional parameters, and is a generalization of the well-known modularity objective function. We show that for such an objective function in a semi-supervised setting we can compute an additive approximation of the optimal solution in the general case, and a constant-factor approximation when the optimal objective value is large. The supervision that we consider is in the form of cluster assignment queries and same-cluster queries; we also study the setting where the query responses are noisy. Our algorithm also generalizes to the min-sum objective function, for which we can achieve similar performance guarantees. We present computational experiments to show that our framework is effective for clustering text data - we are able to find clusterings that are close to the queried clustering and have a good objective value.

## CCS CONCEPTS

• **Computing methodologies** → **Semi-supervised learning settings**; **Cluster analysis**.

## KEYWORDS

semi-supervised learning, clustering, max-sum clustering, min-sum clustering, modularity, text clustering

**ACM Reference Format:**
Konstantin Voevodski. 2020. Semi-Supervised Max-Sum Clustering. In *Proceedings of the 29th ACM International Conference on Information and Knowledge Management (CIKM '20), October 19–23, 2020, Virtual Event, Ireland*. ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/3340531.3411896

## 1 INTRODUCTION

Clustering is usually studied in an unsupervised setting where we would like to assign a set of data points to clusters without any information about the correct clustering. One way to approach this problem is to define some objective function (or "score") that describes the qualify of the clustering, and then try to optimize this objective. Two of the most commonly used objective functions are the *k-means* and *modularity* objectives. However, they are both known to be hard to optimize. The $k$-means objective is NP-hard even for $k = 2$ [13], and is hard to approximate in Euclidean space [5]. Modularity is NP-complete [11] and hard to approximate [12]. Therefore in practice both of these objectives are optimized by algorithms with limited performance guarantees.

We may also approach clustering as a semi-supervised problem where we are given some information regarding the correct cluster assignments. We may think of the supervision as an *oracle* that answers queries about the correct clustering. The supervision may be in the form of high-level properties of the clustering, such as split/merge requests (see Awasthi et al. [4], Awasthi and Zadeh [6], Balcan and Blum [7]). It may also be in the form of information about the individual data points, such as cluster assignments of points [2] or same-cluster queries for pairs of points [3].

In practical settings semi-supervised approaches are easy to motivate. Clustering is an under-specified task - in the unsupervised setting depending on what clustering algorithm we run we may get drastically different outputs. It is not clear which one is "good" versus "bad", and whether some (or any) of the clusterings can even be used for the intended application. This ambiguity is resolved in practice by searching for any prior knowledge about the data points. We try to seek out domain experts (those very familiar with the data) to provide any information about how some of the points are clustered in a "good" clustering. Then this information is used to choose a clustering output by one of the unsupervised methods. Given this workflow, it is natural to first collect any prior knowledge about the cluster assignments, and then consider a clustering task given this information.

Semi-supervised approaches are especially relevant for clustering text data. Text clustering can be very challenging in the unsupervised setting. For example, consider newspaper articles that may be clustered according to topic, writing style, or the sentiment of the writer. A limited amount of supervision is helpful to give a clustering algorithm some idea about what a "good" clustering looks like. On the other hand, it may be hard to collect enough labeled data points required for a supervised learning algorithm, especially if the number of clusters is large. In Section 4 we show that our semi-supervised algorithm works well for clustering text data - we are able to find fairly accurate clusterings using a limited amount of supervision.

In order to design a semi-supervised clustering algorithm we have to explicitly define what we mean by a *correct* clustering. One approach is to simply say that the oracle is providing the correct clustering, and assume that we have access to a similarity function that is also consistent with this clustering [4, 6, 7]. Another approach is to consider some clustering objective function, and assume that the correct clustering is the one that optimizes this objective. Ashtiani et al. [3] and Ailon et al. [1] study the latter framework in the context of the $k$-means objective function.

Here we consider semi-supervised clustering in the context of the max-sum objective function. Our objective function is the difference between the pairwise within-cluster similarity and some null


*CIKM '20, October 19–23, 2020, Virtual Event, Ireland*

ACM ISBN 978-1-4503-6859-9/20/10.
https://doi.org/10.1145/3340531.3411896

hypothesis regarding the similarity. The within-cluster similarity and the null hypothesis are specified by non-negative functions $f$ and $g$, respectively. This is a natural objective that does not require a predetermined number of clusters or any other parameters. It simply requires the clusters to be enriched for similarity with respect to some null (default) similarity of limited interest. This objective also generalizes the well-known *modularity* objective function (see Section 2.1).

In order to further motivate our max-sum objective function, observe that there may be several meaningful ways to define a null hypothesis. We may define a null hypothesis from $f$ itself by considering the average pairwise similarity, or represent the data as a similarity graph and consider the degrees of the nodes (see Section 2.1). We may also define $f$ and $g$ independently of each other, where $g$ may be used to incorporate additional domain knowledge about the problem. For example, consider a computational biology application where we are clustering proteins. We may want to define $f$ in terms of physical protein interactions and $g$ using known functional annotations. Then optimizing our objective with $f$ and $g$ defined in this manner may give us clusters of physical interactions that are not already annotated.

## 1.1 Our Results

We design approximation algorithms for max-sum clustering in a semi-supervised setting. Given approximation parameters $\delta$ and $\epsilon$, with probability at least $1-\delta$, our algorithm outputs a clustering with objective value at least $OPT - 3/4 \cdot \epsilon n^2$, where $OPT$ is the objective value of the optimal clustering and $n$ is the number of points. The algorithm requires only $O(\epsilon^{-3}\log\frac{k}{\epsilon\delta})$ cluster-assignment oracle queries, where $k$ is the number of clusters in the optimal clustering. This algorithm may also be modified to use $O(k\epsilon^{-3}\log\frac{k}{\epsilon\delta})$ same-cluster oracle queries.

We also consider the setting where the oracle response may be incorrect with probability $\alpha$ (see Section 2). In this *noisy* setting we show that with probability at least $1-\delta$, our algorithm outputs a clustering with expected objective value at least $OPT - 3/4 \cdot \epsilon n^2 - 4\alpha n^2$. The number of cluster-assignment queries required by the algorithm remains unchanged. We further show that our approach generalizes to the min-sum objective, for which we can obtain similar performance guarantees.

Our theoretic analysis for the max-sum objective is most relevant when $OPT = \Omega(n^2)$. This may be the case when we are clustering with a similarity function - in particular when we are clustering text (see Basu et al. [9]) or biological data (see Voevodski et al. [24]). This may also be the case when we are clustering dense graphs (see Greene et al. [19] and Wang et al. [25]) or dense subsets of sparse graphs. In such scenarios we can compute a constant-factor approximation of the optimal solution using only $O(\log\frac{k}{\delta})$ cluster-assignment oracle queries.

In Section 4 we show that our approach is effective for clustering text data. Our algorithm is able to find clusterings that are significantly more accurate than the ones output by alternative semi-supervised and unsupervised approaches. We also show that our supervised algorithm is better at optimizing our objective function when compared to a commonly-used unsupervised algorithm.

## 1.2 Related Work

Supervised clustering has been studied in the context of the $k$-means objective function. On the applied side, Basu et al. [8] and Basu et al. [9] propose semi-supervised variants of Lloyd's algorithm for solving $k$-means. The algorithm of Basu et al. [8] incorporates information about the cluster assignments of individual points. The algorithm of Basu et al. [9] incorporates information about the cluster assignments of pairs of points. They show promising experimental results on the difficult Newsgroups data set. In Section 4 we test our algorithm on the same data.

On the theoretical side, Ashtiani et al. [3] present a supervised $k$-means algorithm that requires $O(k^2 \log k + k \log n)$ same-cluster queries. They prove that if the data has a certain structure (a margin assumption regarding the separation of clusters), their supervised algorithm is able to solve $k$-means instances that are otherwise NP-hard. In a related work, Ailon et al. [1] remove the margin assumption and still get an approximation for the $k$-means problem that is otherwise NP-hard using $O(k^2 \log k)$ same-cluster queries. Our algorithm can have better query complexity (depending on the setting of the approximation parameters), handles noise in the oracle responses (the algorithms of Ashtiani et al. [3] and Ailon et al. [1] do not), and works better in practice (we compare performance in Section 4).

Clustering with noisy queries is also considered by Mazumdar and Saha [21]. They present algorithms that use noisy same-cluster queries, and give an $\Omega(\frac{nk}{(1-2\alpha)^2})$ lower bound on the number of queries required to find the correct clustering for an oracle that is incorrect with probability $\alpha$. User supervision is also studied in the context of metric learning, where the goal is to adapt a metric given pairwise constraints regarding the distances (see Bilenko et al. [10] and Davis et al. [14]).

Our max-sum objective function is a generalization of *modularity* [23]. Modularity is very well-studied - see, in particular, Fortunato and Barthelemy [16], Muff et al. [22], Ziv et al. [26]. Modularity is known to be NP-complete [11] and APX-hard [12], therefore in practice heuristics are used to optimize it. In Section 4 we compare performance with one such heuristic (greedy hierarchical clustering).

Our algorithm is inspired by the randomized algorithm of Goldreich et al. [18] for approximating the maximum cut in a graph. Goldreich et al. [18] observe that we can use sampling to estimate the similarity of a point to each side of a cut. Of course we do not know the maximum cut, therefore the algorithm of Goldreich et al. [18] must enumerate all possible binary partitions of the sampled points such that one of them matches the partition of the sampled points in the optimal cut. Goldreich et al. [18] also show that this idea may be extended to $k$-way cuts where now all possible $k$-way partitions must be enumerated.

Giotis and Guruswami [17] show that we can approximate the correlation clustering *max agreements* objective function using effectively the same algorithm. The algorithm of Giotis and Guruswami [17] uses sampling to estimate both within-cluster similarity and between-cluster dissimilarity. As before, they must enumerate all possible partitions of the sampled points into $k$ clusters.

We make the observation that for our max-sum objective function we can use sampling to estimate the similarity of a point to

each cluster with respect to both $f$ and $g$ (see Section 3). In addition, given that the sample size is independent of the size of the data set, we may consider our problem in a semi-supervised setting where we simply ask for the cluster assignments of the sampled points. Note that if the sample size is $t$ and we have $k$ clusters, the supervision removes the $k^t$ complexity of enumerating all possible cluster assignments of the sampled points. Therefore in a semi-supervised setting we can approximate our objective without a limitation on the number of clusters. In contrast, an unsupervised algorithm is only feasible when $k$ is a very small constant. We further observe that our approach generalizes to the min-sum objective function, where we can use the supervision to estimate the sum of the distances to each cluster.

## 2 PRELIMINARIES

Given a data set $X$ of size $n$, we use $f\colon \binom{X}{2} \to \mathbb{R}_{\geq 0}$ to denote a pairwise similarity function for the points in $X$. We use $g\colon \binom{X}{2} \to \mathbb{R}_{\geq 0}$ to denote a null hypothesis regarding the pairwise similarity of the points in $X$. Given a $k$-clustering of the points $C = \{C_1, C_2, \dots C_k\}$, we define its objective value $\Phi(C)$ as follows:

$$\Phi(C) = \sum_{l=1}^{k} \sum_{(x_i,x_j)\in C_l} f(x_i, x_j) - g(x_i, x_j).$$

Our max-sum optimization problem is to find a clustering that maximizes this objective value. We assume that both $f$ and $g$ are non-negative and are normalized to output values in $[0, 1]$. We may think of the null hypothesis as some null (default) pairwise similarity that we find to be of limited interest. As we discuss next, the choice of the null hypothesis significantly affects the structure of the optimal clustering.

### 2.1 Defining a Null Hypothesis

There are several meaningful ways to define a null hypothesis. We may define it using the constant function $g(x_i, x_j) = s_{ave}$, where $s_{ave}$ is the average pairwise similarity of the points: $s_{ave} = \frac{1}{n^2}\sum_{x_i,x_j\in X} f(x_i, x_j)$. We refer to this null hypothesis as the *average-similarity null hypothesis*. This definition compares the within-cluster similarity to what we would get by chance (if we assigned points to a cluster of this size uniformly at random).

We can also consider a null hypothesis that corresponds to the *modularity* objective function. In particular, we may represent our data set as an undirected similarity graph, where the similarities are given by $f$. We can then consider a null hypothesis $g(x_i, x_j) = d(i) \cdot d(j)/vol$, where $d(i)$ is the degree of vertex $i$ and $vol$ is the volume of the graph (the sum of the degrees of its nodes). We refer to this null hypothesis as the *degree-based null hypothesis*.

Modularity is known to reward larger clusters. For a set of nodes $S$, let us use $vol(S)$ to denote the sum of the degrees of its nodes. Fortunato and Barthelemy [16] observe that if we have two sparsely connected communities $S_1$ and $S_2$, merging them may still increase modularity as long as $vol(S_1)$ and $vol(S_2)$ are small compared to $vol$. In particular, if we have $vol(S_1) = vol(S_2) = v$, we can verify that $e$ edges (of weight 1) between $S_1$ and $S_2$ are sufficient for the merge to increase modularity as long as we have $v < \sqrt{e \cdot vol/2}$. In other words, $\sqrt{e \cdot vol/2}$ is the *resolution limit* for detecting communities connected by $e$ edges.

In order to address this limitation, we may consider a null hypothesis of the form $g(x_i, x_j) = \eta \cdot d(i) \cdot d(j)/vol$, where $\eta$ is a parameter that tunes the density of the clusters with respect to $f$. A larger setting of $\eta$ favors denser clusters of smaller size, while a smaller setting of $\eta$ favors sparser clusters of larger size. Note that if $\eta = 0$, then the optimal solution puts all the vertices into a single cluster. We can verify that if we use this definition with $\eta > 1$, the resolution limit decreases to $\sqrt{e \cdot vol/(2\eta)}$.

Also, observe that we may define the null hypothesis $g$ independently of $f$ to explicitly discourage certain pairs of objects from being assigned to the same cluster. This can be useful in applications where we have two similarity functions, and we want the clusters to be consistent with one of them and not the other, such as the computational biology application described in Section 1.

### 2.2 Notation and Definitions

We use $C^* = \{C_1^*, C_2^*, \dots C_k^*\}$ to denote the clustering that optimizes our objective function. We refer to $C^*$ as the *optimal* clustering. We use $OPT = \Phi(C^*)$ to denote its objective value and use $k$ to denote the number of clusters in $C^*$. We consider two clusterings $C$ and $C'$ equivalent if there is a bijection $\sigma$ between the two sets of cluster indices such that for all $C_i \in C$ we have $C_i = C'_{\sigma(i)}$.

We model the supervision in the form of oracle queries that reveal information about the optimal clustering $C^*$. More precisely, we assume that we have access to a *cluster assignment oracle* that given a point reveals its cluster assignment in $C^*$ (see Definition 2.1).

Definition 2.1 (Cluster Assignment Oracle). *Given a point $x \in X$, the oracle returns the index $i$ such that $x \in C_i^*$.*

We also consider a *same-cluster* oracle that given two points reveals whether or not they belong to the same cluster in $C^*$ (see Definition 2.2).

Definition 2.2 (Same-Cluster Oracle). *Given points $x, y \in X$, the oracle returns true if there is a cluster $C_i^*$ such that $x \in C_i^*$ and $y \in C_i^*$, and returns false otherwise.*

We observe that cluster-assignment queries may be reduced to same-cluster queries (see Proposition 2.3). This relationship was also observed in Ashtiani et al. [3].

Proposition 2.3 (Relationship Between Oracle Models). *Any $t$ cluster-assignment queries may be reduced to at most $kt$ same-cluster queries.*

Proof. Given $t$ points $x_1, x_2, \dots x_t$, we show how to assign cluster indices to them using only same-cluster queries. Assign $x_1$ a new cluster index. Then for $j = 2, ..., t$, query $x_j$ w.r.t. all $x_i$ for $i < j$, skipping any $x_i$ with redundant cluster indices. If the same-cluster oracle returns true for any $x_i$, assign $x_j$ the same cluster index as $x_i$. Otherwise assign $x_j$ a new cluster index. □

In order to more realistically model the supervision we consider a *noisy cluster assignment oracle*. Given a noise parameter $\alpha$, the oracle reveals the correct cluster assignment in $C^*$ with probability $1 - \alpha$ (see Definition 2.4).

Definition 2.4 ($\alpha$-Noisy Cluster Assignment Oracle). *Given a point $x \in X$, with probability $1-\alpha$, the oracle returns the index $i$ such that $x \in C_i^*$. Otherwise the oracle returns some index $j \neq i$.*

We also consider the setting where the supervision simply reveals some fixed *user clustering* $C^u$ (see Definition 2.5).

Definition 2.5 (Cluster Assignment User Query). *Given a point $x \in X$, the query returns the index $i$ such that $x \in C_i^u$, where $C^u$ is some fixed user clustering.*

For a subset of points $S \subseteq X$, we define $f(x,S) = \sum_{y \in S} f(x,y)$, and define $g(x,S)$ similarly. When $S = \emptyset$, we define $f(x,S) = g(x,S) = 0$.

We use $\mathbf{E}[X]$ to denote the expected value of random variable $X$. Proposition 2.6 restates a bound regarding sums of random variables (due to Hoeffding [20]). We use it to estimate $f(x,S)$ and $g(x,S)$ for subsets of points $S \subseteq X$.

Proposition 2.6. *Let $X_1, X_2, ..., X_t$ be $t$ independent random variables with $X_i \in [0,1]$. Let $\mu = (1/t) \cdot \sum_i \mathbf{E}[X_i]$, and $\bar{X} = (1/t) \cdot \sum_i X_i$. Then for any $\gamma \in [0,1]$, we have $Pr[\bar{X} > \mu + \gamma] \leq exp(-2t\gamma^2)$, and $Pr[\bar{X} < \mu - \gamma] \leq exp(-2t\gamma^2)$.*

## 3 ALGORITHM

We first give the general idea behind our algorithm, and then follow with its description and performance analysis.

### 3.1 General Idea

Suppose we sample a sufficiently large set of points $S \subset X$ uniformly at random, and then generate a clustering $\mathcal{D}$ of the points in $S$ by using *cluster-assignment* queries. For each point $x \in X$ we may then assign it to cluster $l = \text{argmax}_j f(x, D_j) - g(x, D_j)$. Here $f(x, D_j) = \sum_{y \in D_j} f(x,y)$ and $g(x, D_j) = \sum_{y \in D_j} g(x,y)$. Our sample size enables us to bound $|f(x, D_j) - f(x, C_j^*)|$ and $|g(x, D_j) - g(x, C_j^*)|$. We will then either put $x$ into the correct cluster in $C^*$ or put it in a different cluster where its contribution to the objective value does not change much. Therefore when we assign $x$ in this manner the objective value cannot get much worse. However, this argument only holds if all other points are still assigned as in $C^*$. If a lot of the other points change clusters we cannot get a bound on the objective value of the output clustering.

Instead we will partition our data set into $m$ equal sized sets $X_1, ..., X_m$, and for $i = 1, 2, \ldots m$, assign the points in $X_i$ by sampling points in $X \setminus X_i$. Then regardless of how many points in $X_i$ change clusters, we can still bound the change in the objective value w.r.t. all pairs of points $(x,y)$ such that $x \in X_i$ and $y \in X \setminus X_i$. We follow with the exact description of the algorithm and its performance analysis.

### 3.2 Description and Performance Analysis

Our algorithm first partitions the point set $X$ into $m$ equal-sized sets. The partition may be done in any manner. We use $X_1, ..., X_m$ to refer to these sets. In iteration $i = 1, \ldots m$, the algorithm assigns points in $X_i$ using a small sample of points from $X \setminus X_i$. In our semi-supervised framework, the information that we use from the sample of points in $X \setminus X_i$ is a combination of the cluster assignments we have made so far (if any), and queries for the cluster assignments.

**Algorithm 1** Supervised-Max-Sum($X, k, f, g, \epsilon, \delta$)

```
Initialize m = 2/ε, t = (32^2/(2ε^2)) log(64mk/(εδ))
Partition X into m sets X_1, X_2, ..., X_m of equal size
Initialize C = C_1, C_2, ... C_k to k empty clusters
for i = 1 to m do
    D = Assign-Clusters(X \ X_i, k, C, t)
    for x ∈ X_i do
        Let l = argmax_j f(x, D_j) − g(x, D_j)
        Assign x to C_l
    end for
end for
Output C
```

We use $\mathcal{D}$ to refer to the corresponding clustering of the sampled points. We then assign each point in $X_i$ based on its similarity to the clusters in $\mathcal{D}$. The entire algorithm is described in Algorithm 1. In our description we use Cluster-Assignment-Query($x$) to denote the query for the cluster assignment of $x$.

We next state the theorem regarding the algorithm performance. Theorem 3.1 is most relevant when $OPT$ is large. In such cases we output a constant-factor approximation of the optimal solution using a very limited number of oracle queries. Corollary 3.2 formalizes this observation.

Theorem 3.1. *Given a data set $X$ of size $n$, a cluster assignment oracle, and approximation parameters $\delta, \epsilon$, with probability at least $1-\delta$, Algorithm 1 outputs a clustering with objective value at least $OPT - 3/4 \cdot \epsilon n^2$ using $O(\epsilon^{-3} log \frac{k}{\epsilon\delta})$ oracle queries and runs in time $O(n\epsilon^{-2} log \frac{k}{\epsilon\delta})$.*

Corollary 3.2. *Suppose $OPT = \Omega(n^2)$. Given a data set $X$ of size $n$, a cluster assignment oracle, and approximation parameter $\delta$, with probability at least $1-\delta$, Algorithm 1 outputs a constant-factor approximation of the optimal solution using $O(\log \frac{k}{\delta})$ oracle queries and runs in time $O(n \log \frac{k}{\delta})$.*

Proof. Suppose $OPT = \Omega(n^2)$. It follows that there exists a constant $c$ such that $OPT \geq cn^2$. We can then set $\epsilon = 2c/3$ to obtain a solution with objective value at least $OPT - (3/4)(2c/3)(n^2) = OPT - (c/2)(n^2) \geq cn^2 - (c/2)(n^2) = (c/2)(n^2)$. Given that $OPT$ may be at most $n^2$, this objective value is a $(2/c)$-approximation of the optimal solution. For this setting of $\epsilon$, Algorithm 1 requires $O(\log \frac{k}{\delta})$ oracle queries and runs in time $O(n \log \frac{k}{\delta})$. □

We now follow with a detailed analysis of the algorithm that enables us to prove Theorem 3.1. In order to proceed with our analysis, we define the following sequence of *intermediate* clusterings $H^1, H^2, \ldots H^{m+1}$. These *intermediate* clusterings correspond to the execution of the outer-most for-loop in Algorithm 1. The clustering $H^i$ assigns the points in $X_1, \ldots X_{i-1}$ as clustered by the algorithm so far, and assigns the rest of the points as clustered in $C^*$. In particular, we have $H^1 = C^*$, and $H^{m+1}$ is equivalent to the clustering output by Algorithm 1. We will use subscripts to refer to the clusters of $H^i$. For example, $H_j^i$ refers to cluster $j$ of $H^i$. Given that in iteration $i$ we sample from vertices in $X \setminus X_i$, we will use $\hat{H}^i$

**Algorithm 2** Assign-Clusters($X, k, C, t$)

Let $S = U^t(X)$ // Sample $t$ points from $X$ u.a.r.
Initialize $\mathcal{D} = D_1, D_2, \ldots D_k$ to $k$ empty clusters
**for** $x \in S$ **do**
  **if** $x$ is assigned to some $C_j \in C$ **then**
    Assign $x$ to $D_j$
  **else**
    Let $j$ = Cluster-Assignment-Query($x$)
    Assign $x$ to $D_j$
  **end if**
**end for**
Output $\mathcal{D}$

to refer to $H^i$ with vertices from $X_i$ removed. In other words, we have $\hat{H}^i_j = H^i_j \setminus X_i$ for $j = 1, \ldots k$.

We next observe that sampling enables us to accurately estimate similarity to the clusters in $\hat{H}^i$. This enables us to bound the decrease in the objective function in each iteration of the algorithm. Definition 3.3 formalizes what we mean by accurately.

Definition 3.3. *Consider iteration $i$ of Algorithm 1. We say that a point $x \in X_i$ is* good *with respect to $f$ if the following holds for each $j \in 1, \ldots k$:*

$$\left| \frac{f(x, D_j)}{t} - \frac{f(x, \hat{H}^i_j)}{|X \setminus X_i|} \right| \leq \epsilon/32. \tag{1}$$

*Otherwise we say that $x$ is* bad *with respect to $f$. Similarly, we say that a point $x \in X_i$ is* good *with respect to $g$ if the following holds for each $j \in 1, \ldots k$:*

$$\left| \frac{g(x, D_j)}{t} - \frac{g(x, \hat{H}^i_j)}{|X \setminus X_i|} \right| \leq \epsilon/32. \tag{2}$$

*Otherwise we say that $x$ is* bad *with respect to $g$.*

Lemma 3.4 shows that for each $X_i$ most points are *good* with respect to $f$. Clearly, the same statement also holds with respect to $g$.

Lemma 3.4. *With probability at least $1 - \delta/2$, for each $i = 1, \ldots m$, at most an $\epsilon/16$ fraction of the points $x \in X_i$ are* bad *with respect to $f$.*

Proof. Let us fix some $X_i$ and some point $x \in X_i$. Consider the sequence of points $s_1, \ldots s_t$ sampled by Algorithm 2 from $X \setminus X_i$. For each cluster $j \in 1, \ldots k$ in $\hat{H}^i$, we define $t$ random variables $\Psi^j_1, \ldots \Psi^j_t$ (corresponding to the sampled points) in the following manner: $\Psi^j_l = f(x, s_l)$ if $s_l \in \hat{H}^i_j$ and $\Psi^j_l = 0$ otherwise.

Observe that by definition, $\sum_{l=1}^{t} \Psi^j_l = f(x, D_j)$. In addition, for each $l \in 1, \ldots t$, we have $\mathbf{E}(\Psi^j_l) = f(x, \hat{H}^i_j)/|X \setminus X_i|$. Then by Proposition 2.6 we have:

$$Pr\left[ \left| \frac{f(x, D_j)}{t} - \frac{f(x, \hat{H}^i_j)}{|X \setminus X_i|} \right| > \epsilon/32 \right] < 2e^{-2t(\epsilon/32)^2}.$$

For our choice of $t$, we have $2e^{-2t(\epsilon/32)^2} = \epsilon\delta/(32mk)$. Therefore for any fixed point $x$ and cluster $j$, the probability that Equation 1 is not satisfied is less than $\epsilon\delta/(32mk)$. Taking the union bound, the probability that $x$ is *bad* (Equation 1 is not satisfied for some cluster $j$) is less than $\epsilon\delta/(32m)$. Using Markov's inequality, the probability that more than an $\epsilon/16$ fraction of the points are *bad* is less than or equal to $(\epsilon\delta/(32m))/(\epsilon/16) = \delta/(2m)$. The lemma follows by taking the union bound over all $m$ sets $X_i$. □

Using Lemma 3.4 we can bound the loss in the objective function in each iteration of the algorithm and prove Theorem 3.1.

Proof of Theorem 3.1. We show that $\Phi(H^{i+1})$ is at most $6/16 \cdot \epsilon^2 n^2$ smaller than $\Phi(H^i)$ . Given that we start with $H^1 = C^*$, and $H^{m+1}$ is the clustering output by the algorithm, it follows that the output clustering must have objective value within $m \cdot 6/16 \cdot \epsilon^2 n^2 = (2/\epsilon) \cdot 6/16 \cdot \epsilon^2 n^2 = 3/4 \cdot \epsilon n^2$ of $OPT$. By construction, our algorithm requires at most $mt = O(\epsilon^{-3} \log \frac{k}{\epsilon\delta})$ queries, and runs in time $nt = O(n\epsilon^{-2} \log \frac{k}{\epsilon\delta})$.

We now give the argument for $\Phi(H^{i+1})$ in relation to $\Phi(H^i)$. By definition, the difference between $H^{i+1}$ and $H^i$ is only in the cluster assignments of the points in $X_i$. We first consider the placement of the points in $X_i$ with respect to each other. We assume that we take the largest possible loss for each such pair. Given that both $f$ and $g$ are normalized to output values in $[0, 1]$, the largest possible loss for each such pair is 1, and we may lose up to $|X_i|^2$ in the objective value in total.

Then it suffices to focus on the placement of the points in $X_i$ with respect to the points in $X \setminus X_i$. We can verify that when we assign points that are *good* with respect to both $f$ and $g$ (see Definition 3.3), we can lose at most $4 \cdot \epsilon n/32 = \epsilon n/8$ in the objective value.

Finally, for points that are *bad* with respect to $f$ or $g$, we again assume that we take the largest possible loss, which is at most $1 \cdot |X \setminus X_i| < n$. Using Lemma 3.4 with respect to $f$ and $g$, and then taking the union bound, we can see that with probability at least $1 - \delta$ the fraction of such points is at most $\epsilon/16 + \epsilon/16 = \epsilon/8$.

Recalling that $|X_i| = \epsilon n/2$, the total loss in the objective value for $H^{i+1}$ with respect to $H^i$ may be at most $(\epsilon n/2)^2 + (\epsilon n/2)(\epsilon n/8) + (\epsilon/8)(\epsilon n/2)(n) = 6/16 \cdot \epsilon^2 n^2$. □

In Section 2 we observe that *cluster-assignment* oracle queries may be reduced to *same-cluster* oracle queries. It follows that we can then modify Algorithm 1 to use *same-cluster* queries.

Corollary 3.5. *Algorithm 1 can be modified to use $O(k\epsilon^{-3} \log \frac{k}{\epsilon\delta})$ same-cluster oracle queries while achieving the same performance guarantee.*

Proof. The corollary follows immediately from Theorem 3.1 and Proposition 2.3. The cluster-assignment queries required by Algorithm 2 may be reduced to same-cluster queries using the procedure in the proof of Proposition 2.3. Note that we must run the procedure exactly once (after collecting the cluster-assignment queries required by all iterations of Algorithm 2) such that the assigned cluster indices are consistent across all iterations. □

We also observe that if the clusters in the optimal clustering $C^*$ are large, then Algorithm 1 may be modified to not take the number of clusters as input.

Theorem 3.6. *If each cluster in $C^*$ has size $\Omega(n/k)$, then Algorithm 1 can be modified to not take $k$ as input. Given a data set $X$*

*of size n, a cluster assignment oracle, and approximation parameters $\delta, \epsilon$, with probability at least $1 - 2\delta$, the modified algorithm outputs a clustering with objective value at least $OPT - 3/4 \cdot \epsilon n^2$ using $O(k \log\frac{k}{\delta} + \epsilon^{-3} \log\frac{k}{\epsilon\delta})$ oracle queries and runs in time $O(n\epsilon^{-2} \log\frac{k}{\epsilon\delta})$.*

Proof. The algorithm may be modified as follows. We first sample a set of points $Q \subset X$ uniformly at random, and query their cluster assignments. Let $I$ be the set of unique cluster indices returned by the queries. We then run Algorithm 1 only considering clusters $C_i$ and $D_i$ for $i \in I$. In particular, any points belonging to other clusters may be ignored in Algorithm 2 when constructing $\mathcal{D}$. We can verify that if $|Q| = O(k \log\frac{k}{\delta})$, with probability at least $1 - \delta$, we will sample a point from each cluster in $C^*$. When this is the case, the analysis of the algorithm remains unchanged, and with probability at least $1 - \delta$ we output a clustering with objective value at least $OPT - 3/4 \cdot \epsilon n^2$. Taking the union bound over the two events, we achieve the performance guarantee with probability at least $1 - 2\delta$. The asymptotic run time of the algorithm remains unchanged. □

## 3.3 Noisy Cluster Assignment Oracle

In practice it's hard to expect the supervision to be completely correct. Instead, the query responses may only be correct on average. We model this scenario using noisy cluster assignment queries (see Definition 2.4). Theorem 3.7 gives a bound on the expected objective value of the clustering output by the algorithm in this setting.

Theorem 3.7. *Given a data set $X$ of size $n$, an $\alpha$-noisy cluster assignment oracle, and approximation parameters $\delta, \epsilon$, with probability at least $1 - \delta$, Algorithm 1 outputs a clustering with expected objective value at least $OPT - 3/4 \cdot \epsilon n^2 - 4\alpha n^2$ using $O(\epsilon^{-3} \log\frac{k}{\epsilon\delta})$ oracle queries and runs in time $O(n\epsilon^{-2} \log\frac{k}{\epsilon\delta})$.*

Proof. The proof is similar to the proof of Theorem 3.1 with the following modification. We need to reconsider the analysis regarding the placement of the points in $X_i$ with respect to the points in $X \setminus X_i$. In particular, we reconsider the analysis regarding the placement of the points that are *good* with respect to both $f$ and $g$.

To represent the oracle noise, consider the random variables $\Omega_1, \ldots \Omega_t$ corresponding to the sampled points $s_1, \ldots s_t$, which are defined in the following manner: $\Omega_l = 1$ if the oracle assigns $s_l$ incorrectly, and $\Omega_l = 0$ if the oracle assigns $s_l$ correctly or the oracle is not asked to assign $s_l$. We will use $D_1, \ldots D_k$ to refer to the execution of the algorithm without any oracle noise, and use $\tilde{D}_1, \ldots \tilde{D}_k$ to refer to the execution with the $\alpha$-noisy oracle.

Consider some point $x \in X_i$ and some cluster $j$. We observe that $|f(x, D_j) - f(x, \tilde{D}_j)| \leq \sum_{l=1}^{t} \Omega_t$, and similarly $|g(x, D_j) - g(x, \tilde{D}_j)| \leq \sum_{l=1}^{t} \Omega_t$. To verify this, consider that each point incorrectly assigned by the oracle may incorrectly assign a point in $D_{l \neq j}$ to $D_j$, or may incorrectly assign a point in $D_j$ to some $D_{l \neq j}$. Given that both $f$ and $g$ are normalized to output values in $[0, 1]$, each incorrectly assigned point may then add or remove at most 1 from $f(x, D_j)$ and $g(x, D_j)$. Also, by definition, we have $\mathbf{E}[\sum_{l=1}^{t} \Omega_t] \leq \alpha t$. Then it follows that $\mathbf{E}[|f(x, D_j) - f(x, \tilde{D}_j)|] \leq \alpha t$ and $\mathbf{E}[|g(x, D_j) - g(x, \tilde{D}_j)|] \leq \alpha t$.

Again, let us consider some point $x \in X_i$ that is *good* with respect to both $f$ and $g$. We can verify that in expectation we lose at most $4 \cdot \epsilon n/32 + 4 \cdot \alpha n$ in the objective value when we assign such point. This gives an additional expected loss of $4\alpha n$ for any point in $X_i$ that is *good* with respect to both $f$ and $g$. The analysis for all other points remains unchanged. In each iteration, we then in expectation lose at most an additional $|X_i| \cdot 4\alpha n = (n/m) \cdot 4\alpha n$ in the objective value. Given that we have a total of $m$ iterations, overall in expectation we lose at most an additional $4\alpha n^2$ in the objective value. □

## 3.4 Oracle Queries Versus User Queries

Assuming that we have access to an oracle that reveals the optimal clustering may be unrealistic even in a noisy setting. Here we consider a weaker assumption that our queries simply reveal some fixed user clustering $C^u$ (see Definition 2.5). We can still expect $C^u$ to have a good objective value, and we can use the queries to compute a clustering where the objective is not much worse. Theorem 3.8 states the performance of our algorithm in this setting.

Theorem 3.8. *Given a data set $X$ of size $n$, access to cluster assignment user queries, and approximation parameters $\delta, \epsilon$, with probability at least $1 - \delta$, Algorithm 1 outputs a clustering with objective value at least $\phi - 3/4 \cdot \epsilon n^2$ using $O(\epsilon^{-3} \log\frac{k'}{\epsilon\delta})$ queries and runs in time $O(n\epsilon^{-2} \log\frac{k'}{\epsilon\delta})$, where $\phi$ is the objective value of the user clustering $C^u$, and $k'$ is the number of clusters in $C^u$.*

Proof. The proof follows immediately from the analysis in Section 3.2, where it suffices to replace $C^*$ with $C^u$. We can still define a sequence of *intermediate* clusterings $H^1, H^2, \ldots H^{m+1}$, where $H^1 = C^u$, and $H^{m+1}$ is the output clustering. The argument in Theorem 3.1 regarding the difference between $\Phi(H^{i+1})$ and $\Phi(H^i)$ remains unchanged. Given that we start with $\Phi(H^1) = \phi$, the objective value of the output clustering must be within $3/4 \cdot \epsilon n^2$ of $\phi$. The arguments regarding the number of queries required by the algorithm and the run time remain unchanged. □

## 3.5 Min-Sum Objective Function

Our algorithm may be modified to optimize the min-sum objective function. Given a $k$-clustering of the points $C = \{C_1, C_2, \ldots C_k\}$, here we use $\Phi(C)$ to denote its min-sum objective value, which is defined as follows:

$$\Phi(C) = \sum_{l=1}^{k} \sum_{(x_i, x_j) \in C_l} d(x_i, x_j).$$

Here $d$: $\binom{X}{2} \to \mathbb{R}_{\geq 0}$ is a distance function for the points in $X$. As before, we assume that $d$ is non-negative and is normalized to output values in $[0, 1]$. Note that our algorithm does not require $d$ to be a metric. Clearly, for the min-sum objective we would like to minimize $\Phi(C)$, given that we want a clustering where points in the same cluster have small pairwise distances. Also, note that the min-sum objective requires a predetermined number of clusters - if we are allowed to use any setting of $k$ then there is a trivial optimal solution that puts each point in its own cluster.

The algorithm for the min-sum objective is described in Algorithm 3. As before, for a subset of points $S \subseteq X$, we define

**Algorithm 3** Supervised-Min-Sum($X, k, d, \epsilon, \delta$)

Initialize $m = 2/\epsilon$, $t = \frac{32^2}{2\epsilon^2}\log\frac{64mk}{\epsilon\delta}$
Partition $X$ into $m$ sets $X_1, X_2, ..., X_m$ of equal size
Initialize $C = C_1, C_2, \dots C_k$ to $k$ empty clusters
**for** $i = 1$ **to** $m$ **do**
  $\mathcal{D}$ = Assign-Clusters($X \setminus X_i, k, C, t$)
  **for** $x \in X_i$ **do**
    Let $l = \text{argmin}_j d(x, D_j)$
    Assign $x$ to $C_l$
  **end for**
**end for**
Output $C$

$d(x, S) = \sum_{y \in S} d(x, y)$ if $S$ is non-empty, and $d(x, S) = 0$ otherwise.

Theorem 3.9 states the performance of Algorithm 3. Note that in this setting *OPT* refers to the objective value of the optimal min-sum clustering, and the oracle queries are also with respect to the optimal min-sum clustering.

Theorem 3.9. *Given a data set $X$ of size $n$, a cluster assignment oracle, and approximation parameters $\delta, \epsilon$, with probability at least $1 - \delta/2$, Algorithm 3 outputs a clustering with objective value at most $OPT + 5/8 \cdot \epsilon n^2$ using $O(\epsilon^{-3}\log\frac{k}{\epsilon\delta})$ oracle queries and runs in time $O(n\epsilon^{-2}\log\frac{k}{\epsilon\delta})$.*

Proof. Similar to the analysis in the previous sections, we can define a sequence of intermediate clusterings $H^1, H^2, \dots H^{m+1}$ and $\hat{H}^1, \hat{H}^2, \dots \hat{H}^{m+1}$ that correspond to the execution of Algorithm 3. For iteration $i$ of Algorithm 3, we say that a point $x \in X_i$ is *good* if we have $\left|\frac{d(x,D_j)}{t} - \frac{d(x,\hat{H}^i_j)}{|X \setminus X_i|}\right| \leq \epsilon/32$ for each $j \in 1, \dots k$. Otherwise we say that $x$ is *bad*.

We can show that $\Phi(H^{i+1})$ is at most $5/16 \cdot \epsilon^2 n^2$ larger than $\Phi(H^i)$, where $\Phi$ now refers to the min-sum objective function. Given that we start with $H^1$ equivalent to the optimal min-sum clustering, and $H^{m+1}$ is the clustering output by the algorithm, it follows that the output clustering must have objective value within $m \cdot 5/16 \cdot \epsilon^2 n^2 = (2/\epsilon) \cdot 5/16 \cdot \epsilon^2 n^2 = 5/8 \cdot \epsilon n^2$ of *OPT*. The query complexity and run-time of the algorithm remain unchanged.

We now give the argument for $\Phi(H^{i+1})$ in relation to $\Phi(H^i)$. As before, the difference between $H^{i+1}$ and $H^i$ is only in the cluster assignments of the points in $X_i$. We first consider the placement of the points in $X_i$ with respect to each other. As before, we can see that when we assign these points, the largest possible loss for each such pair is 1, and we may lose up to $|X_i|^2$ in the objective value in total.

We now consider the placement of the points in $X_i$ with respect to the points in $X \setminus X_i$. We can verify that when we assign *good* points, we can lose at most $2 \cdot \epsilon n/32 = \epsilon n/16$ in the objective value. For *bad* points we again assume that we take the largest possible loss, which is at most $1 \cdot |X \setminus X_i| < n$. Per the argument in Lemma 3.4, with probability at least $1 - \delta/2$, the fraction of *bad* points is at most $\epsilon/16$.

Recalling that $|X_i| = \epsilon n/2$, the total loss in the objective value for $H^{i+1}$ with respect to $H^i$ may be at most $(\epsilon n/2)^2 + (\epsilon n/2)(\epsilon n/16) + (\epsilon/16)(\epsilon n/2)(n) = 5/16 \cdot \epsilon^2 n^2$. □

We can also consider a noisy cluster-assignment oracle when optimizing the min-sum objective (see Definition 2.4). Theorem 3.10 states the performance of the algorithm in this setting.

Theorem 3.10. *Given a data set $X$ of size $n$, an $\alpha$-noisy cluster assignment oracle, and approximation parameters $\delta, \epsilon$, with probability at least $1 - \delta/2$, Algorithm 3 outputs a clustering with expected objective value at most $OPT + 5/8 \cdot \epsilon n^2 + 2\alpha n^2$ using $O(\epsilon^{-3}\log\frac{k}{\epsilon\delta})$ oracle queries and runs in time $O(n\epsilon^{-2}\log\frac{k}{\epsilon\delta})$.*

Proof. The proof is similar to the proof of Theorem 3.9 with the following modification. We need to reconsider the analysis regarding the placement of the points in $X_i$ with respect to the points in $X \setminus X_i$. In particular, we reconsider the analysis regarding the placement of the *good* points.

As before, to represent the oracle noise, consider the random variables $\Omega_1, \dots \Omega_t$ corresponding to the sampled points $s_1, \dots s_t$, which are defined in the following manner: $\Omega_l = 1$ if the oracle assigns $s_l$ incorrectly, and $\Omega_l = 0$ if the oracle assigns $s_l$ correctly or the oracle is not asked to assign $s_l$. We will use $D_1, \dots D_k$ to refer to the execution of the algorithm without any oracle noise, and use $\tilde{D}_1, \dots \tilde{D}_k$ to refer to the execution with the $\alpha$-noisy oracle.

Consider some point $x \in X_i$ and some cluster $j$. As before, we can verify that $|d(x, D_j) - d(x, \tilde{D}_j)| \leq \sum_{l=1}^{t} \Omega_t$. Also, by definition, we have $\mathbf{E}[\sum_{l=1}^{t} \Omega_t] \leq \alpha t$. Then it follows that $\mathbf{E}[|d(x, D_j) - d(x, \tilde{D}_j)|] \leq \alpha t$.

Again, let us consider some *good* point $x \in X_i$. We can verify that in expectation we lose at most $2 \cdot \epsilon n/32 + 2 \cdot \alpha n$ in the objective value when we assign such point. This gives an additional expected loss of $2\alpha n$ for any such point. The analysis for all other points remains unchanged. In each iteration, we then in expectation lose at most an additional $|X_i| \cdot 2\alpha n = (n/m) \cdot 2\alpha n$ in the objective value. Given a total of $m$ iterations, overall in expectation we lose at most an additional $2\alpha n^2$ in the objective value. □

As in the previous sections, we can modify Algorithm 3 to use same-cluster queries, and also consider the setting where the queries reveal some fixed user clustering rather than the optimal clustering. We can then prove the equivalent of Corollary 3.5 and Theorem 3.8 for Algorithm 3 regarding its performance in these settings.

## 4 EXPERIMENTAL RESULTS

We perform computational experiments on the Newsgroups data set available from the UCI Machine Learning Repository [15]. The points in the data set are text documents. Each text document is a post to one of 20 different online forums (called newsgroups). We use a pre-processed version of the data set.[1] In this data set duplicate posts are removed, and headers that may identify a newsgroup are removed from each post. The data set contains 18,828 points.

Each document is represented by a term frequency - inverse document frequency (tf-idf) vector encoding the uni-grams in the

[1]The data set is available at http://qwone.com/~jason/20Newsgroups.

document. We keep all standard ASCII characters in each document and do not remove any stop words. The tf-idf vectors are compared using cosine similarity. Given that all the feature vectors are non-negative, this gives a similarity function in $[0, 1]$ for all pairs of points.

We term our Algorithm 1 *Supervised Max-Sum*. In order to define our max-sum objective we need to specify a null hypothesis. In our experiments we use the degree-based null hypothesis (see Section 2.1). In Section 4.3 we also consider scaling the degree-based null hypothesis to reward less dense versus more dense clusters, and using the average-similarity null hypothesis (see Section 2.1).

## 4.1 Hyper-Parameter Settings

We need to set the parameters of Algorithm 1 to use a certain number of oracle queries. Observe that by construction, Algorithm 1 requires at most $mt$ queries, where $m$ is the number of partitions and $t$ is the maximum number of queries per partition. We set $m$ consistent with our theoretic analysis. Given that we require $O(\epsilon^{-3} log \frac{k}{\epsilon\delta})$ queries, we choose a larger setting of $\epsilon$ to limit the number of queries, setting $\epsilon = 2/3$, and therefore set $m = 2/\epsilon = 3$. We then vary $t$ heuristically to use more versus less queries.

## 4.2 Comparison with Semi-Supervised Methods

We compare our algorithm with three other semi-supervised approaches in terms of the ability to recover the given clustering. We compare performance with Algorithm 1 from Ashtiani et al. [3], which we term *Supervised K-Means*. *Supervised K-Means* uses both cluster-assignment queries and same-cluster queries, we only count the number of cluster-assignment queries. We also compare with the algorithm of Ailon et al. [1], which they term *Query-K-Means++*. This algorithm may use either cluster-assignment or same-cluster queries; we implement it using cluster-assignment queries. For the supervised $k$-means algorithms we convert cosine similarity to a distance by subtracting from 1.0.

*Supervised K-Means* iteratively constructs $k$ clusters, but does not specify what happens to any unclustered points at the end of the last iteration. In our implementation we assign each such point to the closest of the $k$ clusters using average distance. *Query-K-Means++* does not support using more than a limited number of queries - it takes the number of clusters as input and only queries points to find new clusters. Therefore its performance remains unchanged as we allow for using more queries.

We also compare with *Supervised K-Nearest Neighbors (KNN)* - taking the majority label of the $K$ nearest queried points. We set $K$ to the square root of the data set size (a common heuristic).

Our experiments test the recovery of the given clustering. For each algorithm, we query the given clustering, and then evaluate the output clustering by calculating the fraction of points that are classified correctly. The clusters in the output clustering are labeled using the cluster assignments of the queried points. We rerun any algorithm that uses randomness 10 times and report the average performance. The experimental results are displayed in Figure 1.

We can see that our algorithm performs significantly better than the alternative methods. We believe that our algorithm performs

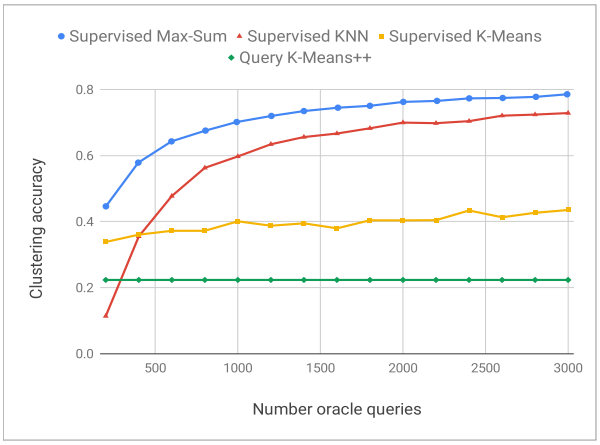


**Figure 1: Performance comparison with other semi-supervised algorithms. The horizontal axis displays the number of cluster-assignment oracle queries. The vertical axis displays the clustering accuracy (fraction of points that are classified correctly).**

better because our objective function is well-suited for this clustering problem. Moreover, our algorithm is the only approach that explicitly handles more realistic semi-supervised learning scenarios such as noise in the supervision (see Section 3.3) and the supervision revealing some fixed clustering with a good objective value rather than the optimal clustering (see Section 3.4).

## 4.3 Choice of Null Hypothesis

We also consider how the choice of the null hypothesis affects the accuracy of the output clustering. We compare performance with scaling the degree-based null hypothesis by a constant $\eta$, as well as using the average-similarity null hypothesis (see Section 2.1). We find that settings in the range $1 < \eta \leq 1.5$, which reward denser clusters of smaller size, improve accuracy (see Figure 2). On the other hand, setting $\eta < 1$, which rewards sparser clusters of larger size, decreases accuracy. We also observe that using the average-similarity null hypothesis significantly decreases accuracy.

## 4.4 Comparison with Unsupervised Methods

We also compare the performance of our algorithm with an unsupervised approach that optimizes the same objective function. Our unsupervised algorithm is a greedy hierarchical clustering. The hierarchical clustering is computed in a bottom-up fashion using average linkage. In each iteration the two most similar clusters are merged. We report the clustering with the largest objective value over all iterations, and term this algorithm *Unsupervised Max-Sum*. This algorithm scales very poorly even to medium-sized data sets, so we can only run it on a subset of the Newsgroups data. We sample 6000 data points from the Newsgroup data set uniformly at random, and term this data set Newsgroup-6000. The sample contains some points from each newsgroup.

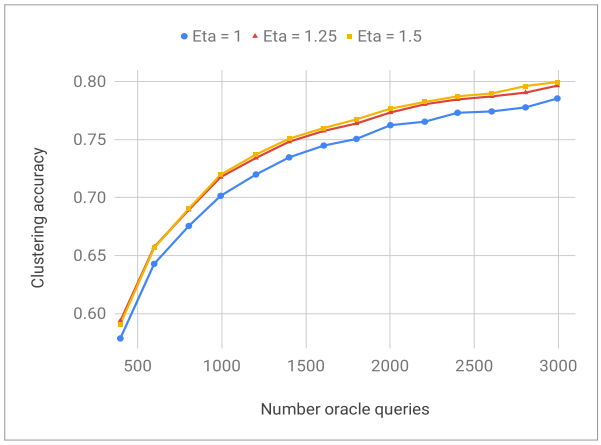


**Figure 2: Performance comparison for different choices of the null hypothesis. The horizontal axis displays the number of cluster-assignment oracle queries. The vertical axis displays the clustering accuracy (fraction of points that are classified correctly).**

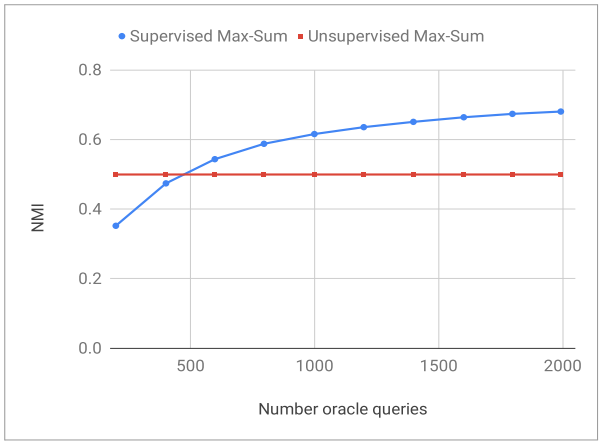


**Figure 3: Performance comparison with unsupervised max-sum optimization on the Newsgroups-6000 data set. The horizontal axis displays the number of cluster-assignment oracle queries. The vertical axis displays the normalized mutual information (NMI) of the output clustering with respect to the given clustering.**

Figure 3 compares the accuracy of *Supervised Max-Sum* and *Unsupervised Max-Sum*. Here we use *normalized mutual information* (NMI) to evaluate clustering accuracy because the unsupervised algorithm does not label the output clusters. Normalized mutual information is defined as $2 \cdot I(Y;Z)/(H(Y)+H(Z))$. Here $Y$ is the random variable specifying the cluster assignments of the points in the output clustering, and $Z$ is the random variable specifying the cluster assignments of the points in the given clustering. $H(Y)$ denotes the entropy of $Y$, $H(Z)$ denotes the entropy of $Z$, and $I(Y;Z)$ denotes the mutual information between $Y$ and $Z$.

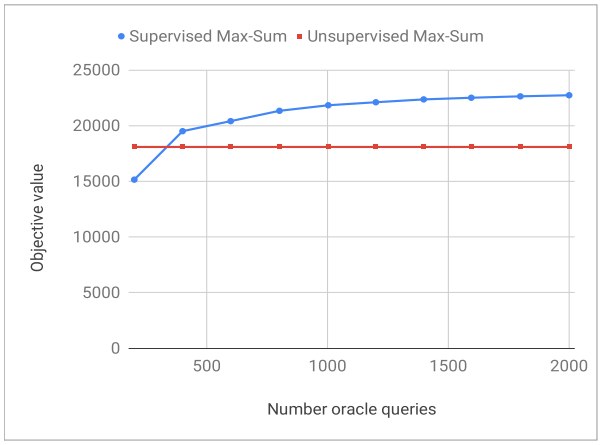


**Figure 4: Performance comparison with unsupervised max-sum optimization on the Newsgroups-6000 data set. The horizontal axis displays the number of cluster-assignment oracle queries. The vertical axis displays the objective value of the output clustering.**

We also compare the two algorithms in terms of the objective value of the output clustering. Figure 4 displays these results. Our experimental results show that our supervised algorithm is significantly better than the unsupervised algorithm in terms of both finding the given clustering (see Figure 3) and optimizing our objective function (see Figure 4).

## 5 DISCUSSION

In this work we design an algorithm to optimize the max-sum objective function in a semi-supervised setting. Our max-sum objective is a generalization of the well-known modularity objective function. We prove that our algorithm finds an additive approximation to the optimal solution in the general case, and a constant-factor approximation when $OPT$ (the value of the optimal solution) is large. We further study settings where the oracle response is noisy, and where the supervision reveals some fixed user clustering (with a good objective value) rather than the optimal clustering. Our algorithm also generalizes to the min-sum objective function, for which we can achieve similar performance guarantees. We present computational experiments to show that our algorithm is effective in practice - our clustering solutions are more accurate and have better objective value than the ones output by alternative semi-supervised and unsupervised methods.

Our work may be extended in several directions. For the max-sum objective we may further study instances where the number of clusters in the optimal clustering is not known, and some of

these clusters are small. It would be interesting to design a semi-supervised clustering algorithm for the noisy same-cluster oracle (here we only consider the noisy cluster-assignment oracle). Our analysis for the max-sum objective is most relevant when $OPT$ is large. It would be interesting to further study instances where $OPT$ is smaller, which would be relevant for data sets like social networks and other sparse similarity graphs.

## REFERENCES


[1] Nir Ailon, Anup Bhattacharya, Ragesh Jaiswal, and Amit Kumar. 2017. Approximate Clustering with Same-Cluster Queries. In *ITCS*.

[2] Hassan Ashtiani and Shai Ben-David. 2015. Representation Learning for Clustering: A Statistical Framework. In *UAI*. 82–91.

[3] Hassan Ashtiani, Shrinu Kushagra, and Shai Ben-David. 2016. Clustering with Same-Cluster Queries. In *NIPS*. 3216–3224.

[4] Pranjal Awasthi, Maria-Florina Balcan, and Konstantin Voevodski. 2014. Local Algorithms for Interactive Clustering. In *ICML*. 550–558.

[5] Pranjal Awasthi, Moses Charikar, Ravishankar Krishnaswamy, and Ali Kemal Sinop. 2015. The Hardness of Approximation of Euclidean k-means. In *SoCG*. 754–767.

[6] Pranjal Awasthi and Reza Bosagh Zadeh. 2010. Supervised Clustering. In *NIPS*. 91–99.

[7] Maria-Florina Balcan and Avrim Blum. 2008. Clustering with Interactive Feedback. In *ALT*. 316–328.

[8] Sugato Basu, Arindam Banerjee, and Raymond J. Mooney. 2002. Semi-Supervised Clustering by Seeding. In *ICML*. 19–26.

[9] Sugato Basu, Arindam Banerjee, and Raymond J. Mooney. 2004. Active Semi-Supervision for Pairwise Constrained Clustering. In *SDM*. 333–344.

[10] Mikhail Bilenko, Sugato Basu, and Raymond J. Mooney. 2004. Integrating Constraints and Metric Learning in Semi-supervised Clustering. In *ICML*. 81–88.

[11] Ulrik Brandes, Daniel Delling, Marco Gaertler, Robert Gorke, Martin Hoefer, Zoran Nikoloski, and Dorothea Wagner. 2008. On Modularity Clustering. *IEEE Transactions on Knowledge and Data Engineering* 20, 2 (2008), 172–188.

[12] Bhaskar Dasgupta and Devendra Desai. 2013. On the Complexity of Newman's Community Finding Approach for Biological and Social Networks. *J. Comput. System Sci.* 79, 1 (2013), 50–67.

[13] Sanjoy Dasgupta. 2008. *The Hardness of k-means Clustering*. Technical Report. Department of Computer Science and Engineering, University of California, San Diego.

[14] Jason V. Davis, Brian Kulis, Prateek Jain, Suvrit Sra, and Inderjit S. Dhillon. 2007. Information-theoretic Metric Learning. In *ICML*. 209–216.

[15] Dheeru Dua and Casey Graff. 2017. UCI Machine Learning Repository. http://archive.ics.uci.edu/ml

[16] Santo Fortunato and Marc Barthelemy. 2007. Resolution Limit in Community Detection. *Proceedings of the National Academy of Sciences* 104, 1 (2007).

[17] Ioannis Giotis and Venkatesan Guruswami. 2006. Correlation Clustering with a Fixed Number of Clusters. In *SODA*. 1167–1176.

[18] Oded Goldreich, Shafi Goldwasser, and Dana Ron. 1998. Property Testing and its Connection to Learning and Approximation. *J. ACM* 45, 4 (1998), 653–750.

[19] Casey S. Greene, Arjun Krishnan, Aaron K. Wong, Emanuela Ricciotti, Rene A. Zelaya, Daniel S. Himmelstein, Ran Zhang, Boris M. Hartmann, Elana Zaslavsky, Stuart C. Sealfon, Daniel I. Chasman, Garret A. FitzGerald, Kara Dolinski, Tilo Grosser, and Olga G. Troyanskaya. 2015. Understanding Multicellular Function and Disease with Human Tissue-Specific Networks. *Nature Genetics* 47 (04 2015).

[20] Wassily Hoeffding. 1963. Probability Inequalities for Sums of Bounded Random Variables. *J. Amer. Statist. Assoc.* 58, 301 (1963).

[21] Arya Mazumdar and Barna Saha. 2017. Clustering with Noisy Queries. In *NIPS*. 5790–5801.

[22] Stefanie Muff, Francesco Rao, and Amedeo Caflisch. 2005. Local Modularity Measure for Network Clusterizations. *Physical Review E* 72, 056107 (2005).

[23] Mark E. J. Newman and Michelle Girvan. 2004. Finding and Evaluating Community Structure in Networks. *Physical Review E* 69, 026113 (2004).

[24] Konstantin Voevodski, Maria-Florina Balcan, Heiko Roglin, Shang-Hua Teng, and Yu Xia. 2012. Active Clustering of Biological Sequences. *The Journal of Machine Learning Research* 13 (01 2012).

[25] Bo Wang, Armin Pourshafeie, Marinka Zitnik, Junjie Zhu, Carlos D. Bustamante, Serafim Batzoglou, and Jure Leskovec. 2018. Network Enhancement as a General Method to Denoise Weighted Biological Networks. *Nature Communications* 9 (12 2018).

[26] Etay Ziv, Manuel Middendorf, and Chris Wiggins. 2005. Information-Theoretic Approach to Network Modularity. *Physical Review E* 71, 046117 (2005).