Supervised Hierarchical Clustering with Exponential Linkage

Nishant Yadav [1] Ari Kobren [1] Nicholas Monath [1] Andrew McCallum [1]

Abstract

In supervised clustering, standard techniques for learning a pairwise dissimilarity function often suffer from a discrepancy between the training and clustering objectives, leading to poor cluster quality. Rectifying this discrepancy necessitates matching the procedure for training the dissimilarity function to the clustering algorithm. In this paper, we introduce a method for training the dissimilarity function in a way that is tightly coupled with hierarchical clustering, in particular single linkage. However, the appropriate clustering algorithm for a given dataset is often unknown. Thus we introduce an approach to supervised hierarchical clustering that smoothly interpolates between single, average, and complete linkage, and we give a training procedure that simultaneously learns a linkage function and a dissimilarity function. We accomplish this with a novel Exponential Linkage function that has a learnable parameter that controls the interpolation. In experiments on four datasets, our joint training procedure consistently matches or outperforms the next best training procedure/linkage function pair and gives up to 8 points improvement in dendrogram purity over discrepant pairs.

1. Introduction

Clustering algorithms are pervasive in data analysis, preprocessing, modeling and visualization (Brown et al., 1992; Alon et al., 1999; Seo & Shneiderman, 2002). While often an unsuperivsed problem, in many cases, there exists a small collection of data points that are labeled with their *ground-truth* cluster assignments. This enables *supervised clustering*, where a dissimilarity function is learned from the labeled data and then used by a clustering algorithm to partition unlabeled data (Finley & Joachims, 2005). Supervised clustering is a common approach in many practical application areas, such as record linkage (Bilenko & Mooney, 2002; Bilenko et al., 2006), coreference resolution (Ng & Cardie, 2002b; Huang et al., 2006; Stoyanov et al., 2009) and image segmentation (Liu et al., 2013).

In both supervised and unsupervised settings, the choice of clustering algorithm bears a significant impact on the quality of the resulting partition of the data. This is because each clustering algorithm is designed with specific inductive biases, for example: k-means performs best when the data points in each cluster are close to that cluster's mean, DBSCAN (Ester et al., 1996) detects clusters composed of contiguous, high-density regions, and hierarchical agglomerative clustering with average linkage discovers clusters for which all within-cluster data points pairs are similar on average. In addition to choosing a clustering algorithm, supervised clustering demands that practitioners choose a training procedure for learning the dissimilarity function that will be used by the algorithm. Often in practice, the training and clustering objectives are mismatched, leading to poor cluster quality.

This is especially relevant for the family of hierarchical agglomerative clustering (HAC) variants, which are widely deployed (Culotta et al., 2007; Lee et al., 2012; Levin et al., 2012; Kenyon-Dean et al., 2018) and are the subject of significant theoretical study (Dasgupta, 2016; Moseley & Wang, 2017). Variants in the HAC family iteratively merge clusters greedily according to the dissimilarity between pairs of clusters which is measured using a *linkage function*. HAC variants are differentiated based on the linkage function. Common linkages, such as single, average, and complete linkage use pairwise data point dissimilarities to compute dissimilarities between pairs of clusters. In the supervised setting, learning is typically performed by training the pairwise dissimilarity function to predict dissimilarity for all within- and across-cluster data points pairs, *regardless* of the selected linkage function (Bilenko & Mooney, 2002; Huang et al., 2006; Stoyanov et al., 2009; Guha et al., 2015).

However, naively training the dissimilarity function to predict whether all data point pairs belong to the same cluster does not lead to robust generalization on unseen data. Especially when clustering is performed with the single linkage

[1]College of Information and Computer Sciences, University of Massachusetts Amherst, USA. Correspondence to: Nishant Yadav <nishantyadav@cs.umass.edu>, Ari Kobren <akobren@cs.umass.edu>, Nicholas Monath <nmonath@cs.umass.edu>, Andrew McCallum <mccallum@cs.umass.edu>.

Proceedings of the 36th International Conference on Machine Learning, Long Beach, California, PMLR 97, 2019. Copyright 2019 by the author(s).

(SL) variant, which defines the dissimilarity between two clusters to be their minimum inter-cluster pairwise dissimilarity, the all-pairs training objective and SL constitute a significant mismatch. We demonstrate this phenomenon empirically and present an algorithm—specifically tailored to HAC with SL—for learning dissimilarity functions that empirically exhibit improved generalization.

Even if equipped with a training algorithm for each HAC variant, it is often difficult to determine which variant is most appropriate for a dataset at hand. Ideally, the choice of the variant would be left to a learning algorithm, which optimizes over a parameterized family of HAC variants that includes single, average and complete linkage.

In this paper, we establish such a family by expressing each of these three linkages as a weighted sum of across-cluster dissimilarities. The weight of an across-cluster dissimilarity is formed by scaling that dissimilarity by a real-value hyperparameter, exponentiating the result and normalizing, i.e., computing the softmax with respect to the all other pairwise dissimilarities involved in the merge. We call this family of HAC linkages the *Exponential linkage (*EXPLINK*) family*.

We present a training algorithm that jointly selects a linkage from the EXPLINK family and learns a pairwise dissimilarity function that is suited to that linkage. The algorithm uses HAC in its inner loop, selecting specific training examples that promote pure mergers and simultaneously optimizes over the EXPLINK family via gradient descent. Crucially, the algorithm obviates the practitioners' need to choose an appropriate linkage and a matching training procedure.

We experiment with a cross-product of HAC variants and a variety of training procedures on four datasets. First, we find that our specially designed training algorithm for HAC with SL leads to improved clustering results over all-pairs training on all datasets. Furthermore, we find that matching the training algorithm and HAC variant leads to improved performance over linkage-agnostic training procedures, such as all-pairs, by up to 8 points of dendrogram purity. Finally, the results reveal that our joint training procedure outperforms or matches the performance of the next best training-procedure/linkage function combination on four datasets. We highlight this result as being useful, especially in practice, because it renders selection of the linkage function before training unnecessary.

2. Supervised Clustering

Clustering is the problem of partitioning a dataset into disjoint subsets. Let $\mathbf{X} = \{x_i\}_{i=1}^{m}, \mathbf{X} \subset \mathcal{X}$, be a dataset of m points. A *clustering* of $\mathbf{X}$ is a collection of disjoint subsets (i.e., clusters) of $\mathbf{X}$, $\mathcal{C} = \{C_i\}_{i=1}^{K'}$, such that $\bigcup_{i=1}^{K'} C_i = \mathbf{X}$. The clustering of $\mathbf{X}$ is computed by a clustering algorithm, $\mathcal{A}$. Often the algorithm makes use of a function, $f_\theta : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$, which computes the dissimilarity of data point pairs in $\mathcal{X}$. We assume that there exists a *ground-truth* clustering, $\mathcal{C}^\star = \{C_i^\star\}_{i=1}^{K}$ of $\mathbf{X}$.

In supervised clustering, the input dataset has associated labels, which are used to produce an algorithm to accurately cluster unseen data.

Definition 1. ***(Supervised Clustering)*** *Let $\mathcal{S} = \{(\mathbf{X}_1, \mathbf{Y}_1) \ldots (\mathbf{X}_n, \mathbf{Y}_n)\} \subset 2^{\mathcal{X}} \times \mathcal{Y}$ be a training set, where each $\mathbf{X}_i = \{x_j\}_{j=1}^{m_i} \subset \mathcal{X}$ is a collection of m_i points and $\mathbf{Y}_i \in \mathcal{Y}$ encodes the ground-truth partition of $\mathbf{X}_i$. The goal is to learn an algorithm $\mathcal{A} : 2^{\mathcal{X}} \to \mathcal{Y}$ that accurately clusters a new dataset $\mathbf{X}$, where $\forall\ 1 \leq i \leq n$, $\mathbf{X} \cap \mathbf{X}_i = \emptyset$ (Finley & Joachims, 2005).*

In practice and in previous work, rather than learning $\mathcal{A}$ directly, a dissimilarity function f_θ is learned instead and used to construct $\mathcal{A}$ (Finley & Joachims, 2005; Culotta et al., 2007; Lee et al., 2012; Levin et al., 2012; Kim et al., 2016; Kenyon-Dean et al., 2018).

This work focuses on supervised *hierarchical* clustering, because of its wide usage in practice. A hierarchical clustering algorithm is one that returns a tree structure for which each leaf corresponds to a unique data point and each internal node corresponds to the cluster of its descendant leaves. Apart from facilitating data exploration and analysis (Seo & Shneiderman, 2002), the primary advantage of hierarchical clustering over *flat* clustering is that the tree simultaneously represents multiple alternative flat clusterings of a dataset, known as *tree consistent partitions* (Heller & Ghahramani, 2005). This alleviates the requirement of specifying the number of clusters *a priori*. In supervised hierarchical clustering, data labels correspond to a flat clustering, as in the non-hierarchical setting, and provide signal to discover trees that encode high quality tree consistent partitions.

2.1. Hierarchical Agglomerative Clustering

Hierarchical agglomerative clustering (HAC) is an iterative algorithm that builds a tree, $\mathcal{T}$, over a dataset one node at a time, according to a *linkage function*. A linkage function $l : 2^{\mathcal{X}} \times 2^{\mathcal{X}} \to \mathbb{R}$ scores the merger of two *nodes*, where each node corresponds to a cluster containing data points stored at its descendant leaves. The algorithm is initialized by creating one node for each data point. The algorithm proceeds in a series of *rounds*. In each round of HAC, the two nodes that minimize the linkage function are *merged*, by making them siblings of one another and creating a new node to serve as their parent. The algorithm terminates after the final merge, which creates the root of the tree. HAC has enjoyed significant study in the theoretical community and usage by practitioners (Eisen et al., 1998; Diez et al., 2015; Yim & Ramdeen, 2015; Gan et al., 2015; Xu et al., 2016; Moseley & Wang, 2017; Ieva et al., 2018; Tie et al., 2018).

3. Illustrative Example

The clustering constructed via HAC depends on the choice of both the linkage function and the pairwise dissimilarity function that powers the linkage function. Given the opportunity to learn the dissimilarity function, the training objective optimized during learning should, in some sense, "match" the chosen linkage. In this section, we empirically show that mismatch between training and clustering objective can result in poor clustering performance. Our example centers on HAC with single linkage (SL).

3.1. Single Linkage

SL computes the dissimilarity between two nodes in $\mathcal{T}$ as the minimum dissimilarity among their corresponding data points. Let $v, v' \in \mathcal{T}$ be nodes in the tree and let C, C' be the sets of data points corresponding to the descendant leaves of v and v', respectively. SL is computed as follows: $l_{\text{SL}}(v, v'; \theta) = \min_{(x_i,x_j)\in C\times C'} f_\theta(x_i, x_j)$. SL is closely related to $\mathbf{X}$'s minimum spanning tree (MST). Consider a clique-structured graph with the data points as nodes and the weight of each edge equal to the dissimilarity of its endpoints, the edges in the MST of $\mathbf{X}$ correspond to the sequence of mergers performed during an invocation of SL (ignoring ties) (Gower & Ross, 1969). SL clustering does not require that all pairs of points within a cluster be similar; rather, for each pair of points in a cluster there must exist a low dissimilarity path. This characteristic of SL facilitates discovery of clusters with arbitrary structure, but also causes its sensitivity to outliers.

3.2. All-pairs Training

A commonly used objective for training dissimilarity functions for supervised (hierarchical) clustering is classification loss on all within- and across-cluster data point pairs (Bilenko & Mooney, 2002; Huang et al., 2006; Stoyanov et al., 2009; Guha et al., 2015), which we will call *all-pairs* (AP). Let $\mathbf{X} = \{x_i\}_{i=1}^m$ be a set of points with ground-truth clusters, $\mathcal{C}^\star = \{C_i^\star\}_{i=1}^K$. Let $\mathbf{x_{i,j}} = (x_i, x_j)$ be a pair of points and let $\mathcal{W}_{\mathcal{C}^\star}, \mathcal{A}_{\mathcal{C}^\star}$ be set of within- and across-cluster pairs of points w.r.t $\mathcal{C}^\star$, respectively . Define all pairs (AP) loss as

$$J_{\text{AP}}(\theta; \mathcal{C}^\star) = \sum_{\mathbf{x_{i,j}}\in\mathcal{W}_{\mathcal{C}^\star}} f_\theta(\mathbf{x_{i,j}}) - \sum_{\mathbf{x_{k,l}}\in\mathcal{A}_{\mathcal{C}^\star}} f_\theta(\mathbf{x_{k,l}}) \quad (1)$$

However, this objective has a mismatch with SL. Under this training objective, misclassifying within- and across-cluster pairs is equally penalized. Yet, misclassified across-cluster pairs are precisely the outliers that lead to the demise of SL. Moreover, when clustering with SL, it is possible to recover the ground-truth partition of a dataset even if the dissimilarity function outputs high dissimilarity for some within-cluster pairs.

We present an example showing that AP training leads to an ineffective dissimilarity function for SL clustering. Fig. 1a shows a dataset in $\mathbb{R}^2$ with two ground-truth clusters, differentiated by color. We use AP to train a linear dissimilarity function $f_\theta(x, x') = \theta_{1:2}^T|x - x'| + \theta_0$, and then use f_θ to cluster via HAC with SL.

Figure 1b contains the result, which shows that the learned function leads to an imperfect clustering. We plot the learned decision boundary, the decision boundary after tuning the bias to minimize clustering errors, the optimal decision boundary, and the vectors $|x - x'|, \forall x, x' \in \mathbf{X}$ (Figure 1c). The figure shows that the decision boundary learned by AP is significantly different from the optimal decision boundary. Visual inspection anecdotally confirms that AP training gives rise to a decision boundary that minimizes overall classification error while the optimal decision boundary does not tolerate any misclassifications of across-cluster pairs, while allowing many within-cluster pairs to be misclassified. Note that it may be possible to use AP with a more complex model to avoid some clustering errors, but employing such a model is more prone to overfit the training set, which is a primary concern in the supervised clustering regime.

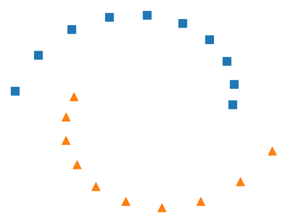
3.3. An MST-based Training Algorithm

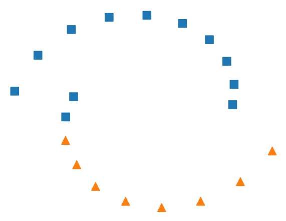
In training a dissimilarity function for SL, the goal should be to minimize misclassifications of across-cluster pairs while retaining the ability to recognize a sufficient number of within-cluster pairs. With these considerations in mind, we propose an MST-based loss function for learning a dissimilarity function that is well-matched with SL. At a high-level, minimizing the loss amounts to learning a low cost MST for each ground-truth cluster while maximizing dissimilarity of across-cluster edges. Note that, like SL, the loss only requires that a handful of within-cluster data point pairs be similar.

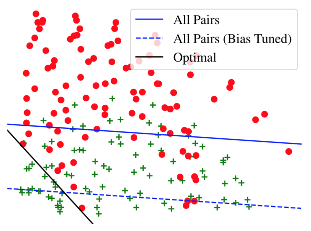
Let $\mathbf{X} = \{x_i\}_{i=1}^m, \mathbf{X} \subset \mathcal{X}$, be a dataset with ground-truth clustering, $\mathcal{C}^\star = \{C_i^\star\}_{i=1}^K$. Let $f_\theta : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ be a function, parameterized by θ, that computes the dissimilarity of a pair of points in $\mathcal{X}$. Let $\texttt{MST}_\theta(C)$ return pairs of points that correspond to the endpoints of the edges in the MST of ground-truth cluster C, with respect to the dissimilarity function f_θ. Then, define the following loss:

$$J_{\text{SL}}(\theta; \mathcal{C}^\star) = \sum_{C\in\mathcal{C}^\star}\left(\sum_{\mathbf{x_{i,j}}\in\texttt{MST}_\theta(C)} f_\theta(\mathbf{x_{i,j}}) - \sum_{x_k\in C}\min_{x_l\notin C} f_\theta(\mathbf{x_{k,l}})\right) \quad (2)$$

This loss can be minimized iteratively. During iteration t, construct an MST for each ground-truth cluster with respect to $f_{\theta^{(t)}}$ and take gradient steps to minimize the corresponding dissimilarities. For each ground-truth cluster $C \in \mathcal{C}^\star, \forall x \in C$, find the point $x' \in \mathbf{X} \setminus C$ that is least dissimilar to x, and take a gradient step to increase the corresponding dissimilarity. Pseudocode appears in Algorithm 1.


(a) Two clusters in $\mathbb{R}^2$


(b) Imperfect clustering



(c) Decison Boundaries

Figure 1. **SL discrepancy with AP**. Figure 1a and 1b depict a dataset and the clustering achieved by training the dissimilarity function with AP followed by clustering with SL and extraction of flat clustering that minimizes errors. Figure 1c shows the absolute difference of within-(green) and across-cluster(red) data points pairs along with decision boundary learned by AP, the decision boundary after tuning bias to minimize clustering errors, and the optimal decision boundary. The learned boundary minimizes classification error of within- and across-cluster pairs, while the optimal boundary does not misclassify any across-cluster pairs but misclassifies many within-cluster pairs.

Algorithm 1 `train_SL`$(\mathbf{X}, \mathcal{C}^\star, T, \gamma)$

```
Init: $\theta$
for $t = 1, \ldots, T$ do
  $J \leftarrow 0$
  for $C \in \mathcal{C}^\star$ do
    for $(x_i, x_j) \in \texttt{MST}_\theta(C)$ do
      $J \leftarrow J + f_\theta(x_i, x_j)$
    for $x \in C$ do
      $J \leftarrow J - \min_{x' \in \mathbf{X} \setminus C} f_\theta(x, x')$
  $\theta \leftarrow \theta - \gamma \frac{\partial J}{\partial \theta}$
```

A variant of the loss function in Equation 2 can be obtained by using a threshold $\tau \in \mathbb{R}$, and margin $\mu \in \mathbb{R}$. In this case, loss is only incurred when pairs of within-cluster data points in the MST have dissimilarity greater than $\tau - \mu$ or when across-cluster pairs have dissimilarity less than $\tau + \mu$.

As anecdotal evidence of this training paradigm, we note that the optimal decision boundary depicted in Figure 1c was, in fact, learned using this training procedure.

4. Exponential Linkage Clustering

While we provided a specialized training procedure for SL that leads to better generalization, in general, the most suitable linkage function for a dataset is, *a priori*, unknown.

In this section, we introduce EXPONENTIAL linkage (EXPLINK), a parametric family of linkage functions for HAC that smoothly interpolates between single, average and complete linkage–three widely used linkage functions. We begin by formally defining EXPLINK (§4.1). We present an example illustrating the advantage of a linkage from EXPLINK over standard linkages (§4.2). Then, we present a training procedure for jointly learning the interpolation parameter of EXPLINK and the dissimilarity function (§4.3).

4.1. Exponential Linkage Function

Let $C_u, C_v \subset \mathcal{X}$, and let $\mathbf{C}_{u,v} = (C_u, C_v)$. We define EXPONENTIAL linkage (EXPLINK) as:

$$\Psi^\alpha(\mathbf{C}_{u,v}) = \frac{\sum_{\mathbf{x_{i,j}} \in C_u \times C_v} e^{\alpha f(\mathbf{x_{i,j}})} f(\mathbf{x_{i,j}})}{\sum_{\mathbf{x_{i,j}} \in C_u \times C_v} e^{\alpha f(\mathbf{x_{i,j}})}} \tag{3}$$

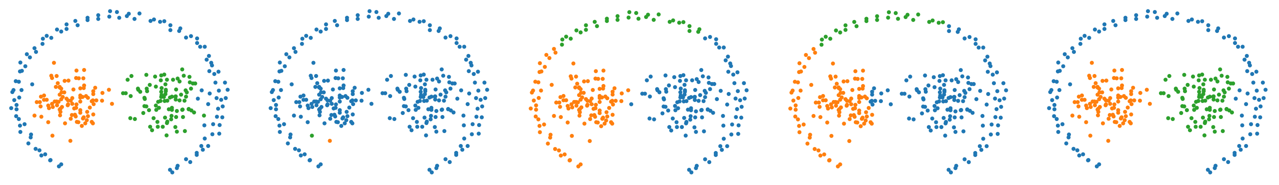
where $\alpha \in \mathbb{R}$ is a hyperparameter that interpolates between members of the EXPLINK family.

EXPLINK computes the dissimilarity between two clusters via a weighted average of all inter-cluster pairwise dissimilarities. Note that as $\alpha \to -\infty$, Ψ^α approaches the minimum pairwise dissimilarity among its arguments, i.e., SL. Similarly, when $\alpha \to \infty$, Ψ^α approaches complete linkage and when $\alpha = 0$, Ψ^α is average linkage.

Different members of the EXPLINK family encode different inductive biases with respect to which pairwise dissimilarities are most important when calculating dissimilarity between two groups of data points.

4.2. Illustrative Synthetic Example

In this section, we demonstrate the flexibility of EXPLINK family. Figure 2 shows a synthetic dataset in $\mathbb{R}^2$ containing three ground-truth clusters. HAC is run with single, average, complete linkage, and EXPLINK with $\alpha{=}-1$ with the dissimilarity function set to Euclidean distance. Flat clusters are obtained by cutting the resultant trees to obtain three clusters. The dataset is interesting because it contains both spherical clusters with noisy boundaries and non-spherical cluster composed of a contiguous, high-density region. Each of the standard linkage functions fail to recover ground-truth clusters for different reasons. SL fails because of the chaining effect (Janowitz, 1978) on noisy cluster boundaries. Average and complete linkages fail because they tend to dis-



(a) Ground-Truth Clusters (b) Single Linkage (c) Average Linkage (d) Complete Linkage (e) EXPLINKwith $\alpha = -1$

Figure 2. Fig. 2a shows three clusters differentiated by color. Fig. 2b, 2c, 2d, 2e shows three flat clusters obtained with HAC using single, average, complete linkage, and EXPLINK with $\alpha = -1$ respectively. All of the standard linkages fail to recover the ground-truth clusters while EXPLINK with $\alpha = -1$, which corresponds to a linkage function that assigns higher weight to lower dissimilarity edges, is able to recover the desired clustering.

cover spherical clusters. However, EXPLINK with $\alpha = -1$, which assigns higher weight to less dissimilar data point pairs, recovers the ground-truth clusters. This anecdotal evidence suggests that various members of the EXPLINK family are capable of recovering clusters of complex shapes.

4.3. Training EXPLINK

As above, let $\mathbf{X} = \{x_i\}_{i=1}^m$ be a set of data points with ground-truth clusters $\mathcal{C}^\star = \{C_i^\star\}_{i=1}^K$. Let $\{\mathcal{T}_k^{(i)}\}_{k=0}^{l_i}$ be the set of trees constructed before round i of HAC. Denote the cluster comprised of the leaves of tree $\mathcal{T}_k^{(i)}$ by $C_k^{(i)}$, and let $\mathcal{C}^{(i)} = \{C_k^{(i)}\}_{k=0}^{l_i}$. Finally, let $\mathcal{P}^{(i)}$ be the set of cluster pairs before round i, $\mathcal{P}_+^{(i)}$ be the set of pair of clusters in $\mathcal{P}^{(i)}$ which are subsets of the same ground-truth cluster, and $\mathcal{P}_-^{(i)}$ be the set of pair of clusters in $\mathcal{P}^{(i)}$ which are subsets of different ground-truth clusters. Then, define the EXPLINK loss function as follows:

$$J(\theta, \alpha) = \sum_{i=1}^{n'} \sum_{\mathbf{C}_{u,v} \in \mathcal{P}_-^{(i)}} \max\left\{0, \Psi^\alpha(\mathbf{C}_{u',v'}) - \Psi^\alpha(\mathbf{C}_{u,v})\right\} \quad (4)$$

where, $\mathbf{C}_{u',v'} = \operatorname{argmin}_{\mathbf{C}_{u,v} \in \mathcal{P}_+^{(i)}} \Psi^\alpha(\mathbf{C}_{u,v})$ and n' is the round of HAC after which no pure merger exists (i.e., $\forall n > n'$, $\mathcal{P}_+^{(n)} = \emptyset$). In words, a loss is incurred when two clusters which are subset of different ground-truth clusters would be merged when a pure merger i.e. a pair of clusters which both belong to the same ground-truth cluster also exists. The training procedure starts with initializing EXPLINK with a particular linkage by randomly picking a value of α. In round i of HAC, find $\mathbf{C}_{u',v'} \in \mathcal{P}_+^{(i)}$, the closest pair of clusters with respect to Ψ^α such that $C_{u'}$ and $C_{v'}$ are subsets of the *same* ground-truth cluster. Then, if a pair of clusters $C_{u,v} \in \mathcal{P}_-^{(i)}$, whose data points belong to *different* ground-truth clusters, has *smaller* linkage cost than $\mathbf{C}_{u',v'}$, then compute gradients to increase the linkage cost of $C_{u,v}$ and to decrease the linkage cost of $\mathbf{C}_{u',v'}$. Then, merge $C_{u'}$ and $C_{v'}$ and repeat this procedure in round $i+1$ if $\mathcal{P}^{(i+1)} \neq \emptyset$. Note that $\mathbf{C}_{u',v'}$ need not be the least dissimilar pair of clusters to merge as per Ψ^α as long as both $C_{u'}$ and $C_{v'}$ belong to the same ground-truth cluster and

Algorithm 2 train_ExpLink($\mathbf{X}, \mathcal{C}^\star, T, \gamma_1, \gamma_2$)

```
Init: $\theta$, $\alpha$
for $t = 1, \ldots, T$ do
  $J \leftarrow 0$
  $\mathcal{T}_j^{(0)} \leftarrow \{x_j\} \quad \forall x_j \in \mathbf{X}$
  for round $i = 1, \ldots, n'$ do
    $\{\mathcal{T}_k^{(i)}\}_k^{l_i} \leftarrow$ HAC-Round$(\{\mathcal{T}_k^{(i-1)}\}_k^{l_{i-1}})$
    $\{C^{(i)}\}_k^{l_i} \leftarrow \{\texttt{lvs}(\mathcal{T}_k^{(i)})\}_k^{l_i}$
    $\mathcal{C}^{(i)} \leftarrow \{C^{(i)}\}_k^{l_i}$
    $\mathcal{P}^{(i)} \leftarrow \{\mathbf{C}_{u,v} \in \mathcal{C}^{(i)} \times \mathcal{C}^{(i)} : C_u \neq C_v\}$
    $\mathcal{P}_+^{(i)} \leftarrow \{\mathbf{C}_{u,v} \in \mathcal{P}^{(i)} : \exists C_j^\star \text{ s.t. } C_u, C_v \subset C_j^\star\}$
    $\mathcal{P}_-^{(i)} \leftarrow \mathcal{P}^{(i)} \setminus \mathcal{P}_+^{(i)}$
    $\mathbf{C}_{u',v'} \leftarrow \operatorname{argmin}_{\mathbf{C}_{u,v} \in \mathcal{P}_+^{(i)}} \Psi^\alpha(\mathbf{C}_{u,v})$
    for $\mathbf{C}_{u,v} \in \mathcal{P}_-^{(i)}$ do
      $J \leftarrow J + \max\left\{0, \Psi^\alpha(\mathbf{C}_{u',v'}) - \Psi^\alpha(\mathbf{C}_{u,v})\right\}$
  $\theta \leftarrow \theta - \gamma_1 \frac{\partial J}{\partial \theta}$
  $\alpha \leftarrow \alpha - \gamma_2 \frac{\partial J}{\partial \alpha}$
```

we never perform an *impure merger* during training. HAC rounds terminate when no pure mergers remain after which parameters are updated w.r.t the loss in Eq. 4 in order to make pure mergers more preferable than competing impure mergers. Pseudocode appears in Algorithm 2.

Empirically, we find that training is more robust when the loss function is augmented with a fixed threshold $\tau \in \mathbb{R}$, and margin $\mu \in \mathbb{R}$. That is, a pure merger incurs a loss only if the two clusters participating in the merge have dissimilarity greater than $\tau - \mu$; an impure merger incurs a loss only if the two clusters have dissimilarity less than $\tau + \mu$.

In order to train the dissimilarity function for a particular member of EXPLINK family, gradient updates to θ, parameters of the dissimilarity function, are performed to minimize loss in Eq. 4 while keeping α fixed. Similarly, the most suitable member of EXPLINK family for a given dissimilarity function is chosen by training α to minimize loss in Eq. 4 while keeping the dissimilarity function parameters fixed.

As anecdotal evidence, we note that the linkage function from ExpLink family used to cluster the data in Figure 2e was learned using this training procedure while by minimizing the loss on the same data.

5. Experiments

We experiment with the cross-product of four linkage functions–single (SL), average (Avg), complete (Comp) and ExpLink (Exp)–and the following eight algorithms for learning pairwise dissimilarity function (f_θ):

- **AllPairs** (**AP**): uses all within- and across-cluster data point pairs to train f_θ.
- **Triplet** (**Trp**): sample a point $x_i \in \mathbf{X}$. Then, sample points x_i^+ and x_i^- to form within- and across-cluster pairs respectively. We generate $N = 100 \cdot |\mathbf{X}|$ such samples.
- **BestEdges** (**Bst**): for each $x \in \mathbf{X}$, generate within- and across cluster pairs by finding the most and least similar data points to it respectively, according to f_θ.
- **MST**: proposed training method for SL (§3.3).
- **Exp-**, **Exp0**, **Exp+** proposed training procedure for ExpLink with fixed α (§4.3). **Exp-** refers to $\alpha = -\infty$, **Exp0** to $\alpha = 0$, and **Exp+** to $\alpha = \infty$.
- **Exp**α: proposed training procedure for ExpLink with joint learning of α and f_θ (§4.3).

Our experiment is designed to determine if the best clustering algorithm depends on the dataset or if there is a universally top performing algorithm. Additionally, our experiment tests whether the performance of a linkage function depends on the training algorithm. We randomly divide each dataset into 50 train/dev/test splits. For each linkage and training algorithm pair, for each split, we learn a dissimilarity function on the training set, tune on the development set and cluster the test set. We record the performance on each split. To compare two training algorithms for the same linkage, we compute their mean performance over all splits. When using Exp as the clustering algorithm with the dissimilarity function trained using a training method other than **Exp**α, the most appropriate linkage from ExpLink is chosen *after* learning the dissimilarity function by minimizing loss in Equation 4 on training data with respect to α.

We conduct experiments with the following four datasets:

- **UMIST Face Data (Faces)** (Graham & Allinson, 1998) : 564 gray-scale images with 20 ground-truth clusters. The pre-cropped images are downsampled to 56×46. We use PCA to reduce the data to 20 dimensions. 7 clusters are used for training, 6 for dev, and 7 for test set.
- **Noun Phrase Coreference (NP Coref)** (Hasler et al., 2006): 104 documents, each contains clusters of coreferent noun phrases (NPs). Each pair of NPs is a described by 102-dim vector (Stoyanov et al., 2009). We use 62 documents for training, 10 for dev, and 32 for test.
- **Rexa** (Culotta et al., 2007): 1459 bibliographic records of authors divided into 8 blocks w.r.t. unique first initial and last name. Each pair of records within a block is represented using 14-dim vector. We use 3 blocks for training, 2 for dev and 3 for test.
- **AMINER** (Wang et al., 2011): 6730 publication records of authors divided into 100 blocks. Each pair of publications within a block is represented by a 8-dim vector (Wang et al., 2011). We use 60 blocks for training, 10 for dev, and 30 for test.

For Faces, we learn Mahalanobis distance matrix $M \succeq 0$. To do so, we learn a matrix A s.t. $M = A^T A$. We use threshold, $\tau = 100$, and margin, $\mu = 10$, when computing the loss. For other datasets, we use feature vectors representing data point pairs and train an average perceptron for the dissimilarity function with threshold, $\tau = 0$, and margin, $\mu = 2$. Code for experiments is available at: `https://github.com/iesl/expLinkage`.

5.1. Hierarchical Clustering Evaluation

Dendrogram Purity The output of hierarchical clustering algorithms is a cluster tree, rather than a flat clustering. Following approach used in Heller & Ghahramani (2005), we evaluate cluster trees using dendrogram purity, which is a holistic measure of the tree quality. Given $\mathcal{T}$, a hierarchical clustering of $\mathbf{X} = \{x_i\}_{i=1}^m$, with ground-truth clusters $\mathcal{C}^\star$, the dendrogram purity of $\mathcal{T}$ is:

$$\texttt{DP}(\mathcal{T}) = \frac{1}{|\mathcal{W}^\star|} \sum_{x_i, x_j \in \mathcal{W}^\star} \texttt{pur}(\texttt{lvs}(\texttt{LCA}(x_i, x_j)), \mathcal{C}^\star(x_i))$$

where $\mathcal{C}^\star(x_i)$ gives the ground-truth cluster of the point x_i, $\mathcal{W}^\star$ is the set of unordered pairs of points belonging to the same ground-truth cluster, $\texttt{LCA}(x_i, x_j)$ is the lowest common ancestor of x_i and x_j in $\mathcal{T}$, $\texttt{lvs}(z) \subset \mathbf{X}$ is the set of leaves for any internal node z in $\mathcal{T}$, and $\texttt{pur}(S_1, S_2) = |S_1 \cap S_2|/|S_1|$.

In words, to compute dendrogram purity of a tree $\mathcal{T}$ with respect to ground-truth clusters $\mathcal{C}^\star$, iterate over all pairs of points (x_i, x_j) which belong to the same ground-truth cluster, find the smallest subtree containing x_i and x_j, and measure fraction of leaves in that subtree which are in the same ground-truth cluster as x_i and x_j.

Table 3a shows mean dendrogram purity of hierarchical clustering for each training method/linkage function pair on four datasets, averaged over the 50 randomly generated train/dev/test splits. In the table, each row represents a training algorithm, and each column a linkage function and dataset. Bold values in each column indicates the best training method for linkage function and dataset corresponding to the column. Values with a single underline indicates that the difference in performance w.r.t the best training method in the column is statistically significant with p-value < 0.05. A double underline is used to indicate statistical significance with p-value < 0.01, where statistical significance is

Obj	Rexa				AMINER				NP Coref				Faces			
	SL	AVG	COMP	EXP	SL	AVG	COMP	EXP	SL	AVG	COMP	EXP	SL	AVG	COMP	EXP
BST	86.9	84.0	75.5	87.8	93.2	93.6	**82.0**	93.5	60.5	57.5	46.8	52.8	93.7	77.4	70.9	93.7
MST	**87.3**	85.1	75.5	88.4	92.7	93.1	81.5	93.2	59.1	54.9	47.6	53.6	**95.3**	81.7	76.2	**95.4**
EXP-	**87.3**	85.6	76.2	88.6	87.6	84.9	77.8	85.3	**63.3**	62.0	55.8	**64.3**	94.5	79.9	74.9	94.6
AP	81.7	84.6	80.1	82.3	92.8	93.4	81.8	93.4	58.7	58.1	50.9	55.3	91.3	85.6	81.7	86.3
TRP	85.6	88.1	**82.4**	89.1	92.0	93.2	81.1	92.9	59.3	61.6	56.6	62.2	91.0	**85.8**	**82.2**	85.8
EXP0	85.5	**88.9**	79.1	**89.5**	**93.4**	**93.9**	81.7	**94.1**	61.0	**62.8**	**57.4**	63.5	91.0	84.8	80.6	90.6
EXP+	83.9	87.3	81.8	88.1	92.4	92.7	81.0	90.7	61.9	**62.8**	56.6	62.5	90.4	84.1	80.5	83.4
EXPα	87.1	86.9	76.4	89.1	**93.4**	**93.9**	81.7	**94.1**	61.2	**62.8**	57.3	63.4	94.2	79.2	73.9	94.5

(a) Dendrogram Purity for all training methods

Obj	Rexa				AMINER				NP Coref				Faces			
	SL	AVG	COMP	EXP	SL	AVG	COMP	EXP	SL	AVG	COMP	EXP	SL	AVG	COMP	EXP
BST	75.0	55.4	38.9	74.2	**79.3**	**78.7**	**43.2**	**81.0**	50.0	44.3	37.1	39.9	**77.6**	60.2	54.6	**78.8**
MST	**75.9**	56.9	41.0	73.4	79.1	77.9	42.1	**81.0**	48.2	43.3	37.5	42.6	76.9	63.9	59.2	77.7
EXP-	74.0	60.8	43.4	76.7	75.1	66.9	43.1	68.3	50.2	44.5	38.0	47.6	73.1	60.8	58.9	74.5
AP	59.8	58.0	49.4	56.9	75.9	76.4	42.9	75.3	46.4	46.7	41.1	43.9	70.2	68.6	65.9	69.4
TRP	66.9	61.2	**56.5**	71.7	73.9	75.3	41.8	77.2	39.7	45.9	38.9	46.0	70.6	**68.8**	**66.2**	69.4
EXP0	70.5	**69.8**	47.0	72.3	76.3	69.8	42.6	74.8	45.8	**47.8**	**41.9**	**48.1**	69.1	68.1	64.6	74.5
EXP+	67.5	65.0	50.0	69.5	74.2	69.3	**43.2**	59.7	**50.8**	47.7	41.2	47.3	67.7	66.4	63.9	68.0
EXPα	74.7	63.5	42.8	**78.1**	75.7	69.5	42.6	75.4	46.0	47.6	41.5	47.9	75.0	62.1	57.6	75.5

(b) Pairwise F1 Score for all training methods

Figure 3. Performance of each training method-linkage pair. Each row corresponds to a training algorithm and each column corresponds to a linkage function and dataset. Each value represents the mean performance of a training algorithm for a linkage over 50 train/dev/test splits. Bold numbers indicate the best performing training method for a particular linkage. Single underline indicates that the value is statistically significantly worse than the best performing method with $p < 0.05$ and double underline indicate $p < 0.01$.

measured using resampled paired-t test (Dietterich, 1998).

Top Performers. For SL, we find that training with a corresponding method like **MST**, **BST** or **EXP-** is best on three out of four datasets[1] [2]. For AVG, training with **EXP0** is best except on the Faces dataset, where training with **TRP** does marginally better than training with **EXP0**. For COMP, we find no recurring pattern except that it achieves much lower dendrogram purity than any of the other linkages. Finally, for EXP, joint training with **EXPα** outperforms other training methods and standard linkages on AMINER, NP Coref and joint training with **EXPα** is only marginally outperformed on Rexa and Faces when choosing a linkage function from EXPLINK *after* training the dissimilarity function. Overall, these results suggest matching the training algorithm and linkage function to achieve the best performance.

All Pairs. Table 3a also reveals that the training method most commonly used in practice, **AP**, is rarely a good choice. For SL and EXP, **AP** is always worse than the top performer by a statistically significant margin–on Rexa, NP Coref and Faces, **AP** causes a 4-8% drop in dendrogram purity for SL and EXP. Despite AVG being the closest match for **AP** among the standard linkages, **AP** is always worse than the top performer for AVG.

Joint Training. A particularly notable result is that jointly learning a linkage from the EXPLINK family and a corresponding dissimilarity function is best or closely tracks the best performer on all datasets. Again, on Faces, training with a method that matches with SL gives best results. Joint training does not require the practitioner to choose a linkage function *a priori*, which is the desired setting for real-world data. Our results suggest that joint training is at least as effective as choosing the best of the standard linkage function/training algorithm pairs. Additionally, joint training is as effective or competitive with choosing a linkage from the EXPLINK family *after* training the dissimilarity function.

5.2. Flat Clustering Evaluation

Pairwise F1. We evaluate flat clusterings using pairwise F-Measure (F1) (Manning et al., 2010). Let $\mathcal{W}^\star$ be pairs of points that belong to the same ground-truth cluster, and $\hat{\mathcal{W}}$ be pairs of points in that belong the same predicted cluster. A true positive is defined as a pair of points that belong to both $\mathcal{W}^\star$ and $\hat{\mathcal{W}}$, a true negative belongs to $\mathcal{W}^\star$ but not to $\hat{\mathcal{W}}$. Similarly, define false positive, and false negatives. Then, compute pairwise F1 as the harmonic mean of pairwise precision and recall.

Selecting a flat clustering with a threshold. To extract a flat clustering from a tree $\mathcal{T}$ given a threshold value ξ, we select the clusters represented by the roots of subtrees for

[1] **MST** and **BST** have minor differences: every positive example generated by **BST** is also generated by **MST**, but **MST** may also include examples to guarantee that the positive training examples are the edges in an minimum spanning tree over the data.

[2] **EXP-** differs from **MST** in that **MST** considers only the best across cluster edge for every point, while **EXP-** might consider more than one across cluster for every point.

which all linkages between siblings in the subtree are less than ξ and the linkage of the subtree root with its sibling is greater than ξ. We select a threshold value that leads to the clustering with maximum pairwise F1 score on dev set. Note that choosing a single partition adds another opportunity to introduce error for all methods (i.e. if the threshold returns a poor partition from a tree with high dendrogram purity).

Table 3b shows mean pairwise-F1 scores for flat clusterings extracted from the trees for 50 random train/dev/test splits (similar to Table 3a). Some trends exists that are similar to those observed with respect to dendrogram purity, albeit with more exceptions. Generally, a linkage function achieves best performance when dissimilarity is learned using a matching training algorithm. SL is dominant on Faces, as is the case with dendrogram purity. The magnitude of relative differences between the methods are larger, for example: on Rexa, when clustering with EXP, joint training leads to more than 20 points higher F1 than **AP** training, and on AMINER, best performing method is better than **EXP+** by 20 points F1. While these results are informative, they depend largely on the method for selecting the tree consistent partition. Methods that select better tree-consistent partitions (than the threshold method) are likely to exist.

6. Related Work

Several approaches have been proposed to learn a distance metric to optimize performance of classification methods such as k-NN and clustering methods such as k-means (Xing et al., 2003; Goldberger et al., 2005; Globerson & Roweis, 2006; Kunapuli & Shavlik, 2012; Weinberger & Saul, 2009; Ashtiani & Ben-David, 2015). These approaches operate under the semi-supervised setting where the metric is learned using a small fraction of within- and across-cluster pairs, or using clustering of a small fraction of data points and the learned distance metric is used on the *same* set of clusters. Balcan & Blum (2008); Awasthi & Zadeh (2010) operate in the setting where the goal is to learn the desired clustering with help of an oracle with the goal of minimizing total number of queries to the oracle. Hierarchical clustering algorithms have also received much attention in trying to incorporate constraints/labels on a small fraction of points (Zheng & Li, 2011; Li et al., 2011; Xiao et al., 2016; Chatziafratis et al., 2018). Our work differs from these in that we operate in the supervised setting where we use *all* labels in the training dataset to learn a pairwise dissimilarity and linkage function for HAC, that generalizes to *different* test set of clusters.

For the task of noun-phrase coreference, several heuristic approaches for generating training examples have been proposed (Soon et al., 2001; Ng & Cardie, 2002a;b). Recent approaches for coreference resolution use latent tree models, where each noun-phrase is linked to its closest/best proceeding noun-phrase (Lassalle & Denis, 2015; Chang et al., 2013; Durrett & Klein, 2013). Our training procedure for SL differs from these approaches in that we do not make use of the word order when constructing an MST over ground-truth clusters, and since these approaches rely on the word order in raw text, they are not applicable to other supervised clustering tasks such as author coreference. The most related to our training procedure for SL is Yu & Joachims (2009), in which the authors learn a latent tree model using SVMs for noun-phrase coreference. Whereas their loss function penalizes the merger of two different ground-truth clusters, ours also penalizes of splitting ground-truth clusters. Unlike our work, they only use SL while our training procedure optimizes over the EXPLINK family.

Culotta et al. (2007) present an approach for generating training examples using errors produced during HAC. Culotta et al. (2007) use a loss that only considers the impure agglomeration that appears in the earliest round of HAC whereas our work uses a loss function based on multiple rounds of a modified version of HAC that considers pure agglomerations. Also, Culotta et al. (2007) learn a function that scores sets and while we learn pairwise dissimilarities.

There is also related work on *unsupervised* hierarchical clustering objectives which describes the quality of a tree structured clustering of the data and, unlike the supervised objectives described in this paper, assume that a similarity or dissimilarity function is given (Dasgupta, 2016; Roy & Pokutta, 2016; Cohen-Addad et al., 2017; Moseley & Wang, 2017; Charikar & Chatziafratis, 2017; Cohen-Addad et al., 2018; Charikar et al., 2019).

7. Conclusion

In this paper, we examine the dependence between training methods and HAC variants in the supervised clustering setting. Using the popular HAC with SL algorithm as an example, we show that mismatch between training and clustering objectives leads to poor performance. Then, we present a training algorithm suited specifically for HAC with SL that yields improved results. We introduce a new family of HAC linkage functions that smoothly interpolates between single, average and complete linkage—called the Exponential Linkage (EXPLINK) family—and provide a joint training algorithm that simultaneously learns an appropriate linkage in the EXPLINK family and a corresponding pairwise dissimilarity function. In experiments, we demonstrate that EXPLINK, coupled with our training algorithm, outperforms or is competitive with the best HAC variant–an important result for practitioners since the best HAC variant for a problem at hand is often unknown. Our experiments also underscore the notion that matching training and clustering objectives leads to superior test time performance in the supervised clustering setting.

Acknowledgments

We thank Ian Gemp and Javier Burroni for many helpful discussions and feedback. We also thank the anonymous reviewers for their constructive feedback. This work was supported in part by the Center for Data Science and the Center for Intelligent Information Retrieval, in part by the Chan Zuckerberg Initiative under the project Scientific Knowledge Base Construction, and in part by the National Science Foundation under Grant No. NSF-1763618. The work reported here was performed in part using high performance computing equipment obtained under a grant from the Collaborative R&D Fund managed by the Massachusetts Technology Collaborative. Any opinions, findings and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect those of the sponsor.

References

Alon, U., Barkai, N., Notterman, D. A., Gish, K., Ybarra, S., Mack, D., and Levine, A. J. Broad patterns of gene expression revealed by clustering analysis of tumor and normal colon tissues probed by oligonucleotide arrays. *Proceedings of the National Academy of Sciences (PNAS)*, 1999.

Ashtiani, H. and Ben-David, S. Representation learning for clustering: A statistical framework. *Uncertainty in Artificial Intelligence (UAI)*, 2015.

Awasthi, P. and Zadeh, R. B. Supervised clustering. *Advances in Neural Information Processing Systems (NeurIPS)*, 2010.

Balcan, M.-F. and Blum, A. Clustering with interactive feedback. *International Conference on Algorithmic Learning Theory (ALT)*, 2008.

Bilenko, M. and Mooney, R. J. Learning to combine trained distance metrics for duplicate detection in databases. *Tech. Report*, 2002.

Bilenko, M., Kamath, B., and Mooney, R. J. Adaptive blocking: Learning to scale up record linkage. *International Conference on Data Mining (ICDM)*, 2006.

Brown, P. F., Desouza, P. V., Mercer, R. L., Pietra, V. J. D., and Lai, J. C. Class-based n-gram models of natural language. *Computational linguistics*, 1992.

Chang, K.-W., Samdani, R., and Roth, D. A constrained latent variable model for coreference resolution. *Empirical Methods in Natural Language Processing (EMNLP)*, 2013.

Charikar, M. and Chatziafratis, V. Approximate hierarchical clustering via sparsest cut and spreading metrics. *Symposium on Discrete Algorithms (SODA)*, 2017.

Charikar, M., Chatziafratis, V., and Niazadeh, R. Hierarchical clustering better than average-linkage. *Symposium on Discrete Algorithms (SODA)*, 2019.

Chatziafratis, V., Niazadeh, R., and Charikar, M. Hierarchical clustering with structural constraints. *International Conference on Machine Learning (ICML)*, 2018.

Cohen-Addad, V., Kanade, V., and Mallmann-Trenn, F. Hierarchical clustering beyond the worst-case. *Advances in Neural Information Processing Systems (NeurIPS)*, 2017.

Cohen-Addad, V., Kanade, V., Mallmann-Trenn, F., and Mathieu, C. Hierarchical clustering: Objective functions and algorithms. *Symposium on Discrete Algorithms (SODA)*, 2018.

Culotta, A., Kanani, P., Hall, R., Wick, M., and McCallum, A. Author disambiguation using error-driven machine learning with a ranking loss function. *Workshop on Information Integration on the Web (IIWeb)*, 2007.

Dasgupta, S. A cost function for similarity-based hierarchical clustering. *Symposium on Theory of Computing (STOC)*, 2016.

Dietterich, T. G. Approximate statistical tests for comparing supervised classification learning algorithms. *Neural computation*, 1998.

Diez, I., Bonifazi, P., Escudero, I., Mateos, B., Muñoz, M. A., Stramaglia, S., and Cortes, J. M. A novel brain partition highlights the modular skeleton shared by structure and function. *Scientific Reports*, 2015.

Durrett, G. and Klein, D. Easy victories and uphill battles in coreference resolution. *Empirical Methods in Natural Language Processing (EMNLP)*, 2013.

Eisen, M. B., Spellman, P. T., Brown, P. O., and Botstein, D. Cluster analysis and display of genome-wide expression patterns. *Proceedings of the National Academy of Sciences (PNAS)*, 1998.

Ester, M., Kriegel, H.-P., Sander, J., and Xu, X. A density-based algorithm for discovering clusters a density-based algorithm for discovering clusters in large spatial databases with noise. *Knowledge Discovery and Data Mining (KDD)*, 1996.

Finley, T. and Joachims, T. Supervised clustering with support vector machines. *International Conference on Machine Learning (ICML)*, 2005.

Gan, Q., Wei, W. C., and Johnstone, D. A faster estimation method for the probability of informed trading using hierarchical agglomerative clustering. *Quantitative Finance*, 2015.

Globerson, A. and Roweis, S. T. Metric learning by collapsing classes. *Advances in Neural Information Processing Systems (NeurIPS)*, 2006.

Goldberger, J., Hinton, G. E., Roweis, S. T., and Salakhutdinov, R. R. Neighbourhood components analysis. *Advances in Neural Information Processing Systems (NeurIPS)*, 2005.

Gower, J. C. and Ross, G. J. Minimum spanning trees and single linkage cluster analysis. *Journal of the Royal Statistical Society: Series C (Applied Statistics)*, 1969.

Graham, D. B. and Allinson, N. M. Characterising virtual eigensignatures for general purpose face recognition. 1998.

Guha, A., Iyyer, M., Bouman, D., and Boyd-Graber, J. Removing the training wheels: A coreference dataset that entertains humans and challenges computers. *North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT)*, 2015.

Hasler, L., Orasan, C., and Naumann, K. Nps for events: Experiments in coreference annotation. *International Conference on Language Resources and Evaluation (LREC)*, 2006.

Heller, K. A. and Ghahramani, Z. Bayesian hierarchical clustering. *International Conference on Machine Learning (ICML)*, 2005.

Huang, J., Ertekin, S., and Giles, C. L. Efficient name disambiguation for large-scale databases. *European conference on principles of data mining and knowledge discovery (PKDD)*, 2006.

Ieva, C., Gotlieb, A., Kaci, S., and Lazaar, N. Discovering program topoi via hierarchical agglomerative clustering. *IEEE Transactions on Reliability*, 2018.

Janowitz, M. An order theoretic model for cluster analysis. *SIAM Journal on Applied Mathematics (SIAP)*, 1978.

Kenyon-Dean, K., Cheung, J. C. K., and Precup, D. Resolving event coreference with supervised representation learning and clustering-oriented regularization. *Joint Conference on Lexical and Computational Semantics (*SEM)*, 2018.

Kim, K., Khabsa, M., and Giles, C. L. Random forest dbscan clustering for uspto inventor name disambiguation and conflation. *IJCAI Workshop on Scholarly Big Data: AI Perspectives, Challenges, and Ideas*, 2016.

Kunapuli, G. and Shavlik, J. Mirror descent for metric learning: A unified approach. *Joint European Conference on Machine Learning and Knowledge Discovery in Databases (ECML PKDD)*, 2012.

Lassalle, E. and Denis, P. Joint anaphoricity detection and coreference resolution with constrained latent structures. *Conference on Artificial Intelligence (AAAI)*, 2015.

Lee, H., Recasens, M., Chang, A., Surdeanu, M., and Jurafsky, D. Joint entity and event coreference resolution across documents. *Empirical Methods in Natural Language Processing (EMNLP)*, 2012.

Levin, M., Krawczyk, S., Bethard, S., and Jurafsky, D. Citation-based bootstrapping for large-scale author disambiguation. *Journal of the American Society for Information Science and Technology (JASIST)*, 2012.

Li, J., Shao, B., Li, T., and Ogihara, M. Hierarchical co-clustering: a new way to organize the music data. *IEEE Transactions on Multimedia*, 2011.

Liu, Y., Liu, J., Li, Z., Tang, J., and Lu, H. Weakly-supervised dual clustering for image semantic segmentation. *Computer Vision and Pattern Recognition (CVPR)*, 2013.

Manning, C., Raghavan, P., and Schütze, H. Introduction to information retrieval. *Natural Language Engineering*, 2010.

Moseley, B. and Wang, J. Approximation bounds for hierarchical clustering: Average linkage, bisecting k-means, and local search. *Advances in Neural Information Processing Systems (NeurIPS)*, 2017.

Ng, V. and Cardie, C. Combining sample selection and error-driven pruning for machine learning of coreference rules. *Association for Computational Linguistics (ACL)*, 2002a.

Ng, V. and Cardie, C. Improving machine learning approaches to coreference resolution. *Association for Computational Linguistics (ACL)*, 2002b.

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., et al. Scikit-learn: Machine learning in python. *Journal of Machine Learning Research (JMLR)*, 2011.

Roy, A. and Pokutta, S. Hierarchical clustering via spreading metrics. *Advances in Neural Information Processing Systems (NeurIPS)*, 2016.

Seo, J. and Shneiderman, B. Interactively exploring hierarchical clustering results [gene identification]. *Computer*, 2002.

Soon, W. M., Ng, H. T., and Lim, D. C. Y. A machine learning approach to coreference resolution of noun phrases. *Computational linguistics*, 2001.

Stoyanov, V., Gilbert, N., Cardie, C., and Riloff, E. Conundrums in noun phrase coreference resolution: Making sense of the state-of-the-art. *Association for Computational Linguistics (ACL)*, 2009.

Tie, J., Chen, W., Sun, C., Mao, T., and Xing, G. The application of agglomerative hierarchical spatial clustering algorithm in tea blending. *Cluster Computing*, 2018.

Wang, X., Tang, J., Cheng, H., and Philip, S. Y. Adana: Active name disambiguation. *International Conference on Data Mining (ICDM)*, 2011.

Weinberger, K. Q. and Saul, L. K. Distance metric learning for large margin nearest neighbor classification. *Journal of Machine Learning Research (JMLR)*, 2009.

Xiao, W., Yang, Y., Wang, H., Li, T., and Xing, H. Semi-supervised hierarchical clustering ensemble and its application. *Neurocomputing*, 2016.

Xing, E. P., Jordan, M. I., Russell, S. J., and Ng, A. Y. Distance metric learning with application to clustering with side-information. *Advances in Neural Information Processing Systems (NeurIPS)*, 2003.

Xu, Z., Xuan, J., Liu, J., and Cui, X. Michac: Defect prediction via feature selection based on maximal information coefficient with hierarchical agglomerative clustering. *International Conference on Software Analysis, Evolution, and Reengineering (SANER)*, 2016.

Yim, O. and Ramdeen, K. T. Hierarchical cluster analysis: comparison of three linkage measures and application to psychological data. *The Quantitative Methods for Psychology*, 2015.

Yu, C.-N. J. and Joachims, T. Learning structural svms with latent variables. *International Conference on Machine Learning (ICML)*, 2009.

Zheng, L. and Li, T. Semi-supervised hierarchical clustering. *International Conference on Data Mining (ICDM)*, 2011.