Supervised k-Means Clustering

Thomas Finley, Thorsten Joachims
Department of Computer Science
Cornell University
Ithaca, NY, USA
{tomf,tj}@cs.cornell.edu

Abstract

The k-means clustering algorithm is one of the most widely used, effective, and best understood clustering methods. However, successful use of k-means requires a carefully chosen distance measure that reflects the properties of the clustering task. Since designing this distance measure by hand is often difficult, we provide methods for training k-means using supervised data. Given training data in the form of sets of items with their desired partitioning, we provide a structural SVM method that learns a distance measure so that k-means produces the desired clusterings[1]*. We propose two variants of the methods – one based on a spectral relaxation and one based on the traditional k-means algorithm – that are both computationally efficient. For each variant, we provide a theoretical characterization of its accuracy in solving the training problem. We also provide an empirical clustering quality and runtime analysis of these learning methods on varied high-dimensional datasets.*

1. Introduction

Clustering is an important data mining task employed in dataset exploration and in other settings where one wishes to partition sets into related groups. Among the algorithms typically used for clustering, k-means and its variants are arguably the most widely used and effective. Successful use of k-means, however, requires a carefully chosen similarity measure that must be constructed to fit the task at hand. For example, in Noun-Phrase Co-Reference Resolution (see e.g., [15]), one must select a similarity measure so that for a given set of noun phrases occuring in a document, those that refer to the same entity in the world are indeed clustered into the same cluster. As another example, in news story clustering [10], one might want to select a similarity measure to cluster articles which are about the same story, as opposed to other criteria. Unfortunately, hand-tuning the similarity measure for specific tasks as these is difficult, since it is unclear how changes in the similarity measure relate to the behavior of the k-means algorithm.

In this paper we propose a supervised learning approach to finding a similarity measure so that k-means provides the desired clusterings for the task at hand. Given training examples of item sets with their correct clusterings, the goal is to learn a similarity measure so that future sets of items are clustered in a similar fashion. In particular, we provide a structural support vector machine (SSVM) algorithm for this supervised k-means learning problem, capable of directly optimizing a parameterized similarity measure to maximize cluster accuracy. We show theoretically and empirically that the algorithm is efficient, and that it provides improved clustering accuracy compared to non-learning methods, as well as compared to more conventional approaches to this supervised clustering problem.

2. Related Work

Supervised clustering is the task of automatically adapting a clustering algorithm with the aid of a training set $\mathcal{S} = \{(\mathbf{x}_1, \mathbf{y}_1), (\mathbf{x}_2, \mathbf{y}_2), \ldots, (\mathbf{x}_n, \mathbf{y}_n)\}$ consisting of n examples of item sets $\mathbf{x}_i$ and complete partitionings of these item sets $\mathbf{y}_i$. Some past applications of supervised clustering include image segmentation [1], news article clustering, noun-phrase co-reference [10], and streaming email batch clustering [11]. These works all learn a parameterized item-pair similarity score from complete partitions of item sets. The process is similar to metric or kernel learning, except the similarity measure is optimized for clustering performance. Methods of [10, 11] provide a structural SVM based supervised clustering for correlation clustering [2]. This paper's method has a similar learning framework, but for the case of practitioners that wish to parameterize k-means. Parameterizing k-means is more difficult than correlation clustering since its relaxation (spectral clustering)

[1] Notationally, a clustering contains multiple clusters, in the same manner that a partitioning contains partitions.

is non-linear [8]. The [1] method learns similarity measures for spectral clustering, and requires a special optimization procedure tightly coupled to a relaxed version of spectral clustering, unable to optimize for discrete k-means clusterers. This paper's method, in contrast, may parameterize any k-means type algorithm.

A related field is semi-supervised clustering, where the typical approach is also to learn a parameterized similarity measure [3, 4, 6, 14]. However, this learning problem is markedly different from supervised clustering. In semi-supervised clustering, the user has a single large dataset to cluster, with incomplete information about the clustering, usually in the form of pairwise constraints about cluster membership. This difference leads to very different algorithms in the two settings.

It is important to recognize that supervised clustering and multiclass classification are completely different. Unlike supervised clustering, multiclass classification by itself is ill-suited to deal with new classes it has not seen before; in noun-phrase coreference and news story clustering, one would want to have partitions for new entities and new news topics, respectively, which is impossible with multiclass classification.

3. Parameterized k-Means

In this section we shall introduce the k-means clustering algorithm, and then describe increasingly complex parameterizations of k-means that allows us to adjust the clusterings k-means produces through supervised learning.

The k-means clustering algorithm is classically described as taking an input set $\mathbf{x}$ of m items, $x_1, x_2, \ldots, x_m$, where each item x_i has some corresponding vector $\psi_i \in \mathbb{R}^N$. A clustering algorithm computes some clustering $\mathbf{y}$ of $\mathbf{x}$ with k clusters so as to minimize intracluster Euclidean distance over these ψ_i, i.e.,

$$\operatorname*{argmin}_{\mathbf{y}} \sum_{c \in \mathbf{y}} \sum_{x_i \in c} \left\| \psi_i - \frac{\sum_{x_j \in c} \psi_j}{|c|} \right\|_2^2. \tag{1}$$

Algebraic manipulation reveals this minimization is equivalent to finding $\mathbf{y}$ to maximize

$$\operatorname*{argmax}_{\mathbf{y}} \sum_{c \in \mathbf{y}} \frac{1}{|c|} \sum_{i,j \in c} \langle \psi_i, \psi_j \rangle \tag{2}$$

in a form often called kernel k-means [8].

To avoid confusion, note that by k-means we refer to the problem of minimizing (1), and *emphatically not* to any one particular instantiation of search procedure that attempts to solve this problem, e.g., batch k-means, point-iterative k-means, or spectral clustering algorithms.

How can we parameterize this (2) objective function to provide a family of similarity measures for learning? A simple but powerful parameterization is to provide some linear weighting $\mathbf{w} \in \mathbb{R}^N$ to distort the ψ_i dimensions:

$$\operatorname*{argmax}_{\mathbf{y}} \sum_{c \in \mathbf{y}} \frac{1}{|c|} \sum_{i,j \in c} \psi_i^T \operatorname{diag}(\mathbf{w}) \psi_j. \tag{3}$$

We can alternately phrase (3) as

$$\operatorname*{argmax}_{\mathbf{y}} \sum_{c \in \mathbf{y}} \frac{1}{|c|} \sum_{i,j \in c} \langle \mathbf{w}, \psi_i \circ \psi_j \rangle. \tag{4}$$

Here, $\circ$ is the componentwise vector product. By changing weights in $\mathbf{w}$, we affect what clustering $\mathbf{y}$ of $\mathbf{x}$ is optimal under this parameterized k-means objective (4).

3.1. Kernel Learning Parameterizations

Though formulation of (4) is simple, it is a somewhat limited parameterization insofar as it requires that points explicitly exist in a vector space. To begin to generalize this, suppose instead of $\psi_i \circ \psi_j$, that any pair x_i, x_j in $\mathbf{x}$ has a corresponding pairwise vector $\psi_{ij} \in \mathbb{R}^N$.

$$\operatorname*{argmax}_{\mathbf{y}} \sum_{c \in \mathbf{y}} \frac{1}{|c|} \sum_{i,j \in c} \langle \mathbf{w}, \psi_{ij} \rangle. \tag{5}$$

If we then define a matrix $K \in \mathbb{R}^{m \times m}$ with entries

$$K_{ij} = \langle \mathbf{w}, \psi_{ij} \rangle \tag{6}$$

we can view (5) as

$$\operatorname*{argmax}_{\mathbf{y}} \sum_{c \in \mathbf{y}} \frac{1}{|c|} \sum_{i,j \in c} K_{ij}. \tag{7}$$

For simplicity, we assume for any K the associated $\mathbf{x}$ and $\mathbf{w}$ are obvious in context.

Work in kernel k-means clustering often specifies that K is symmetric positive semi-definite, i.e., $K \succeq 0$ [8]. Why? The items in the set $\mathbf{x}$ have representations in some (implicit) vector space if and only if $K \succeq 0$ [14]. This is relevant to our setting, since the proof of convergence for batch k-means clustering depends on the existence of this space, and may not converge without it [14].

How can we ensure $K \succeq 0$? Consider an alternate definition of K. For a given $\mathbf{x}$, let $K^{(\ell)} \in \mathbb{R}^{m \times m}$ be the matrix of the ℓ^{th} pairwise feature in pairwise ψ_{ij}, i.e., $K_{ij}^{(\ell)} = \langle \mathbf{e}_\ell, \psi_{ij} \rangle$. We may then define K as $K = \sum_{\ell=1}^{N} \mathbf{w}_\ell K^{(\ell)}$. Restricting $\mathbf{w} \geq 0$ and all $K^{(\ell)} \succeq 0$ will imply $K \succeq 0$, since non-negative linear combinations of symmetric positive semi-definite (SPSD) matrices are likewise SPSD. This style of parameterization has strong connections to the field of kernel learning [14].

Enforcing $\mathbf{w} \geq 0$ is the responsibility of the training procedure, but the constraint on the features in the pairwise ψ_{ij} is the responsibility of the practitioner providing these vectors. Fortunately, this is usually not difficult to satisfy. For example, the very common case with pairwise vectors $\psi_{ij} = \psi_i \circ \psi_j$ seen in (5) satisfies the constraint. More generally, features in ψ_{ij} whose values comes from a kernel function evaluation over $x_i, x_j \in \mathbf{x}$ satisfy the constraint.

3.2. Similarity Learning Parameterizations

The restrictions to enforce $K \succeq 0$ pose practical disadvantages. First, for the user providing ψ_{ij} pairwise feature vectors, ensuring that every single feature is a kernel may be difficult in some settings. Second, enforcing positivity constraints on $\mathbf{w}$ is bothersome insofar as it may complicate the parameter learning procedure, and it is even unhelpful: it is plausible that some pairwise features are *negatively* correlated with common cluster membership. To take a canonical example, if one is clustering web pages, certain link relationships among pages are often strong indicators that pages are of different types [13]. With some effort, tricks may be employed to overcome some of these difficulties (for example, doubling features with positive and negative versions of the features to allow negative correlations, and diagonal offsets large enough to ensure $K \succeq 0$), but this is troublesome and often confusing.

To avoid these problems, the alternative to Section 3.1's restrictions is to simply lift them, i.e., accept any ψ_{ij} pairwise vectors and parameterization $\mathbf{w}$. The cost of this greater simplicity and flexibility is that the resulting K is often no longer SPSD. Though "kernel k-means" becomes a bit of a misnomer in this case, we retain its use, as an established term for this representation. This is not a major problem, but it does restrict us to clustering algorithms robust to $K \not\succeq 0$.

3.3. Nonlinear Parameterizations

The preceding discussion has considered $\mathbf{w}$ to be a real vector $\mathbf{w} \in \mathbb{R}^N$, but it may also be considered a nonlinear parameterization vector. We may view $\mathbf{w}$ as a linear combination of pairwise vectors seen in training, i.e., $\mathbf{w} = \sum_{\hat{i},\hat{j}} \alpha_{\hat{i}\hat{j}} \psi_{\hat{i}\hat{j}}$. In this case, our parameterized pairwise similarity score becomes

$$K_{ij} = \langle \mathbf{w}, \psi_{ij} \rangle = \sum_{\hat{i},\hat{j}} \alpha_{\hat{i},\hat{j}} \left\langle \psi_{\hat{i}\hat{j}}, \psi_{ij} \right\rangle \tag{8}$$

and we may replace the inner product $\left\langle \psi_{\hat{i}\hat{j}}, \psi_{ij} \right\rangle$ with some kernel function $\kappa(\psi_{\hat{i}\hat{j}}, \psi_{ij})$. This allows parameterizations to capture complex non-linear interrelationships among pairwise features.

4. Supervised k-means with SSVMs

With k-means parameterization defined as above, how do we actually learn a parameterization? We provide a supervised approach based on structural support vector machines, taking as input a training set

$$\mathcal{S} = \{(\mathbf{x}_1, \mathbf{y}_1), (\mathbf{x}_2, \mathbf{y}_2), \ldots, (\mathbf{x}_n, \mathbf{y}_n)\}.$$

Each $\mathbf{x}_i \in \mathcal{X}$ is a set of items and $\mathbf{y}_i \in \mathcal{Y}$ a complete partitioning of that set. For example, $\mathcal{S}$ could have $\mathbf{x}_i$ as noun-phrases in a document and $\mathbf{y}_i$ as the partitioning into co-referent sets, or $\mathbf{x}_i$ as a pixel image with $\mathbf{y}_i$ as the segmentation of the image into coherent regions, etc. The output of the learning algorithm is a $\mathbf{w}$-parameterized hypothesis $h : \mathcal{X} \to \mathcal{Y}$, where the clustering algorithm in h uses the $\mathbf{w}$ parameterized similarity measure when clustering inputs $\mathbf{x}$. Intuitively, the goal is to learn some $\mathbf{w}$ so that each $h(\mathbf{x}_i)$ is close to $\mathbf{y}_i$ on the training set, and so that h predicts the desired clustering also for unseen sets of items $\mathbf{x}$.

4.1. Structural SVMs

Structural SVMs are a general method for learning hypotheses with complex structured output spaces [19]. From a training set $\mathcal{S} = ((\mathbf{x}_1, \mathbf{y}_1), \ldots, (\mathbf{x}_n, \mathbf{y}_n))$, a structural SVM learns a hypothesis $h : \mathcal{X} \to \mathcal{Y}$ mapping inputs $\mathbf{x} \in \mathcal{X}$ to outputs $\mathbf{y} \in \mathcal{Y}$, trading off model complexity and empirical risk. A hypothesis takes the form

$$h(\mathbf{x}) = \operatorname*{argmax}_{\mathbf{y} \in \mathcal{Y}} f(\mathbf{x}, \mathbf{y}), \tag{9}$$

maximizing a discriminant function $f : \mathcal{X} \times \mathcal{Y} \to \mathbb{R}$ with

$$f(\mathbf{x}, \mathbf{y}) = \langle \mathbf{w}, \mathbf{\Psi}(\mathbf{x}, \mathbf{y}) \rangle . \tag{10}$$

The $\mathbf{\Psi}$ combined feature vector function relates inputs and outputs, and $\mathbf{w}$ is the model parameterization learned from $\mathcal{S}$. The quality of a hypothesis is evaluated using a loss function $\Delta : \mathcal{Y} \times \mathcal{Y} \to \mathbb{R}$ describing the extent to which two outputs differ. The $\mathbf{\Psi}$ and Δ functions are task dependent.

Structural SVMs find a $\mathbf{w}$ that balances model complexity and empirical risk $R^{\Delta}_{\mathcal{S}}(h) = \frac{1}{n} \sum_{i=1}^{n} \Delta(\mathbf{y}_i, h(\mathbf{x}_i))$ by solving this quadratic program (QP) [19]:

Optimization Problem 1 (STRUCTURAL SVM)

$$\min_{\mathbf{w}, \xi \geq \mathbf{0}} \frac{1}{2}\|\mathbf{w}\|^2 + \frac{C}{n} \sum_{i=1}^{n} \xi_i \tag{11}$$

$$\forall i, \forall \mathbf{y} \in \mathcal{Y} \setminus \mathbf{y}_i : \langle \mathbf{w}, \mathbf{\Psi}(\mathbf{x}_i, \mathbf{y}_i) \rangle \geq \langle \mathbf{w}, \mathbf{\Psi}(\mathbf{x}_i, \mathbf{y}) \rangle + \Delta(\mathbf{y}_i, \mathbf{y}) - \xi_i . \tag{12}$$

Introducing constraints for all possible outputs is typically intractable. However, it has been shown that the cutting plane technique in Algorithm 1 can be used to efficiently to solve OP 1 to arbitrary precision ϵ. This algorithm

Algorithm 1 Cutting plane algorithm to solve OP 1.

1: Input: $(\mathbf{x}_1, \mathbf{y}_1), \ldots, (\mathbf{x}_n, \mathbf{y}_n)$, C, ϵ
2: $S_i \leftarrow \emptyset$ for all $i = 1, \ldots, n$
3: **repeat**
4: **for** $i = 1, \ldots, n$ **do**
5: $H(\mathbf{y}) \equiv \Delta(\mathbf{y}_i, \mathbf{y}) + \langle \mathbf{w}, \mathbf{\Psi}(\mathbf{x}_i, \mathbf{y}) \rangle - \langle \mathbf{w}, \mathbf{\Psi}(\mathbf{x}_i, \mathbf{y}_i) \rangle$
6: compute $\hat{\mathbf{y}} = \operatorname{argmax}_{\mathbf{y} \in \mathcal{Y}} H(\mathbf{y})$
7: compute $\xi_i = \max\{0, \max_{\mathbf{y} \in S_i} H(\mathbf{y})\}$
8: **if** $H(\hat{\mathbf{y}}) > \xi_i + \epsilon$ **then**
9: $S_i \leftarrow S_i \cup \{\hat{\mathbf{y}}\}$
10: $\mathbf{w} \leftarrow$ optimize primal over $\bigcup_i S_i$
11: **end if**
12: **end for**
13: **until** no S_i has changed during iteration

iteratively finds the most violated constraint with a *separation oracle* (line 6), adds it to a working set $\bigcup_i S_i$ if violated by more than desired precision ϵ (line 9), and resolves the QP to find a new parameterization $\mathbf{w}$ (line 10). Algorithm 1 terminates when no new constraint is found, that is, when all constraints in OP 1 are satisfied within ϵ. We will discuss the computational complexity and accuracy of this algorithm for supervised k-means learning in Section 5.

To use structural SVMs to learn parameterizations for k-means clustering, we must (1) state our clustering procedure $h(\mathbf{x})$ in terms of $h(\mathbf{x}) = \operatorname{argmax}_{\mathbf{y}} \langle \mathbf{w}, \mathbf{\Psi}(\mathbf{x}, \mathbf{y}) \rangle$, (2) provide a loss function $\Delta(\mathbf{y}, \hat{\mathbf{y}})$, and (3) provide the separation oracle $\operatorname{argmax}_{\mathbf{y} \in \mathcal{Y}} \langle \mathbf{w}, \mathbf{\Psi}(\mathbf{x}_i, \mathbf{y}) \rangle + \Delta(\mathbf{y}_i, \mathbf{y})$. These are explained in the following three sections.

4.2. Combined Feature Function $\mathbf{\Psi}$

To express $h(\mathbf{x})$ as $h(\mathbf{x}) = \operatorname{argmax}_{\mathbf{y}} \langle \mathbf{w}, \mathbf{\Psi}(\mathbf{x}, \mathbf{y}) \rangle$, we work from (7) and (6):

$$\begin{aligned} h(\mathbf{x}) &= \operatorname*{argmax}_{\mathbf{y} \in \mathcal{Y}} \sum_{c \in \mathbf{y}} \frac{1}{|c|} \sum_{i,j \in c} K_{ij} \\ &\equiv \operatorname*{argmax}_{\mathbf{y} \in \mathcal{Y}} \sum_{c \in \mathbf{y}} \frac{1}{|c|} \sum_{i,j \in c} \langle \mathbf{w}, \psi_{ij} \rangle \\ &\equiv \operatorname*{argmax}_{\mathbf{y} \in \mathcal{Y}} \left\langle \mathbf{w}, \sum_{c \in \mathbf{y}} \frac{1}{|c|} \sum_{i,j \in c} \psi_{ij} \right\rangle. \end{aligned}$$

So, $\mathbf{\Psi}(\mathbf{x}, \mathbf{y})$ is

$$\mathbf{\Psi}(\mathbf{x}, \mathbf{y}) = \sum_{c \in \mathbf{y}} \frac{1}{|c|} \sum_{i,j \in c} \psi_{ij} \tag{13}$$

for the most general parameterization of k-means.

In this work, we also want to represent and learn from "relaxed" clusterings, such as those that appear in methods like spectral clustering. More specifically, we shall provide a matrix representation of clusterings. Consider this alternate representation of clusterings $\mathbf{y}$: for each partitioning $\mathbf{y}$ of m items into k clusters, let $\mathbf{Y} \in \mathbb{R}^{m \times k}$ be an equivalent alternate matrix representation of the clustering. Each column in $\mathbf{Y}$ corresponds to some cluster $c \in \mathbf{y}$, where each element i in the column is $|c|^{-0.5}$ if $i \in c$, and is 0 otherwise. For example, the following two clustering representations are equivalent:

$$\mathbf{y} = \{\{1, 3\}, \{2, 4, 5\}\} \qquad \mathbf{Y} = \begin{bmatrix} \frac{1}{\sqrt{2}} & 0 \\ 0 & \frac{1}{\sqrt{3}} \\ \frac{1}{\sqrt{2}} & 0 \\ 0 & \frac{1}{\sqrt{3}} \\ 0 & \frac{1}{\sqrt{3}} \end{bmatrix}.$$

More formally, any matrix $\mathbf{Y}$ corresponding to a discrete clustering $\mathbf{y}$ will obey three basic constraints. First is column orthonormality: for any columns $\mathbf{Y}_{:,i}$ or $\mathbf{Y}_{:,j}$ from $\mathbf{Y}$, $\|\mathbf{Y}_{:,i}\|_2 = 1$, $\mathbf{Y}_{:,i}^T \mathbf{Y}_{:,j} = 0$, i.e., $\mathbf{Y}^T \mathbf{Y} = I$. Second, each column's nonzero entries are equal: for any pair of column $\mathbf{Y}_{:,i}$'s entries $\mathbf{Y}_{j,i} \neq 0$ and $\mathbf{Y}_{\ell,i} \neq 0$, $\mathbf{Y}_{j,i} = \mathbf{Y}_{\ell,i}$. Third is that there are no negative entries: any entry $\mathbf{Y}_{j,i} \geq 0$.

With this new representation $\mathbf{Y}$, we may rephrase (7) as:

$$\operatorname*{argmax}_{\mathbf{Y}} \operatorname{trace}(\mathbf{Y}^T K \mathbf{Y}). \tag{14}$$

We phrase the objective in terms of (10) to get $\mathbf{\Psi}(\mathbf{x}, \mathbf{Y})$:

$$\begin{aligned} h(\mathbf{x}) &= \operatorname*{argmax}_{\mathbf{Y}} \operatorname{trace}(\mathbf{Y}^T K \mathbf{Y}) \\ &\equiv \operatorname*{argmax}_{\mathbf{Y}} \left\langle \mathbf{w}, \sum_{i=1}^{m} \sum_{j=1}^{m} \left(\mathbf{Y}_{i,:}^T \mathbf{Y}_{j,:} \right) \psi_{ij} \right\rangle. \end{aligned}$$

So, $\mathbf{\Psi}(\mathbf{x}, \mathbf{Y})$ is

$$\mathbf{\Psi}(\mathbf{x}, \mathbf{Y}) = \sum_{i=1}^{m} \sum_{j=1}^{i-1} \left(\mathbf{Y}_{i,:}^T \mathbf{Y}_{j,:} \right) \psi_{ij}. \tag{15}$$

Note that (15) generalizes (13) insofar as the two are equal for any $\mathbf{Y}$ corresponding to $\mathbf{y}$, and (15) is defined for any spectral output $\mathbf{Y}$.

As an aside, that $\mathbf{\Psi}(\mathbf{x}, \mathbf{Y})$ is quadratic in the entries of $\mathbf{Y}$ brings up a subtle but important distinction regarding the generality of structural SVMs versus alternative formulations of OP 1, like max-margin Markov nets (M^3N) [18], associative Markov nets, and their variants [17]. These alternatives require that "inference" (in this case, k-means clustering) be phrased as either a Markov random field or linear program, respectively. One could begin to express the quadratic nature of $\mathbf{Y}$ as pairwise cliques in an M^3N, or linearize clustering by optimizing $\mathbf{Z} = \mathbf{Y}\mathbf{Y}^T$ for associative networks. However, these methods would be incapable of feasibly capturing $\mathbf{Y}$ orthonormality, or the $rank(\mathbf{Z}) = k$ constraint on $\mathbf{Z}$. In contrast, the restriction of the structure

and number of columns of $\mathbf{Y}$, the nonlinearity of $\mathbf{Y}$ in $\mathbf{\Psi}$, and the nonlinearity of the clustering procedure are all incidental and naturally expressed in structural SVMs since the structure of $\mathbf{\Psi}(\mathbf{x}, \mathbf{y})$ is unrestricted.

4.3. Loss Function Δ

The Δ loss function for the dissimilarity between two clusterings we use in this work is

$$\Delta(\mathbf{Y}, \hat{\mathbf{Y}}) = 100 \cdot \left(1 - \frac{1}{k} \operatorname{trace}(\mathbf{Y}^T \hat{\mathbf{Y}} \hat{\mathbf{Y}}^T \mathbf{Y})\right) \quad (16)$$

$$= 100 \cdot \left(1 - \frac{1}{k} \|\mathbf{Y}^T \hat{\mathbf{Y}}\|_F^2\right). \quad (17)$$

For $\mathbf{Y}$ corresponding to a discrete partitioning $\mathbf{y}$, (16) is

$$\Delta(\mathbf{y}, \hat{\mathbf{y}}) = 100 \cdot \left(1 - \tfrac{1}{k} \sum_{c \in \mathbf{y}} \sum_{\hat{c} \in \mathbf{y}} \tfrac{|c \cap \hat{c}|^2}{|c| \cdot |\hat{c}|}\right). \quad (18)$$

This loss Δ has attractive qualities. It is symmetric and invariant to column rearrangements. Also, as seen in (18), Δ essentially counts agreement among pairs of items in clusters which is normalized by the size of the clusters in question. This is favorable relative to alternate loss functions based on the Rand index [16] used in previous supervised clustering work [10]: where this normalization is absent, loss becomes heavily biased against mistakes in larger clusters. Finally, though any judgment about the appropriateness of a loss function must necessarily be subjective, this Δ appears to give qualitatively sensible judgments about the similarity of two clusterings.

4.4. Separation Oracle and Prediction

For the separation oracle $\operatorname{argmax}_{\mathbf{y} \in \mathcal{Y}} \langle \mathbf{w}, \mathbf{\Psi}(\mathbf{x}_i, \mathbf{y}) \rangle + \Delta(\mathbf{y}_i, \mathbf{y})$, the form of Δ is well suited to constructing the separation oracle: one can employ a clustering algorithm as the separation oracle and cluster over the matrix $(K - \frac{1}{k}\mathbf{Y}\mathbf{Y}^T)$ in place of K in the (7) objective.

However, finding the actual clustering $\mathbf{y}$ that globally maximizes (7) either for prediction or computing the most violated constraint is an NP-hard problem. This has led to the adoption of many varied approximate algorithms to maximize this objective function. The survey in [8] characterizes many of the popular clustering algorithms that approximate the maximization of the discriminant function (7). We use three methods from that paper that are all robust to $K \not\succeq 0$. We denote these differing methods as Iterative, Spectral, and Discrete. In prediction, one could use other clustering methods if one conformed to SPSD restrictions on K as defined in Section 3.1, including batch k-means, normalized cut algorithms, etc. In the separation oracle, however, we must use these robust methods: even with $K \succeq 0$, it is quite possible that $(K - \frac{1}{k}\mathbf{Y}\mathbf{Y}^T) \not\succeq 0$.

4.4.1 Iterative Point-Incremental Clustering

Iterative is point-incremental k-means [7]. We use point-incremental (i.e., recomputing cluster centers with each point reassignment) and not standard batch (i.e., recomputing cluster centers after a pass over all points) k-means since K easily becomes non-SPSD without positivity constraints on $\mathbf{w}$'s elements, breaking batch k-means' convergence guarantees.

The algorithm works by randomly assigning all m items to k clusters, and then iterating over all points, reassigning them to the cluster with the "closest" cluster center. Unlike typical batch k-means clustering which waits until a pass is completed before updating cluster centers, point-iterative k-means updates the centers upon each point reassignment. Compared to batch k-means, point-iterative k-means does not depend upon $K \succeq 0$ and tends to produce clusterings with lower intracluster distance [7].

4.4.2 Spectral Clustering

Spectral is a straightforward eigenanalysis of K to produce a "relaxed" clustering in the matrix representation $\mathbf{Y}$ described in Section 4.2. If we relax of Section 4.2's constraints on $\mathbf{Y}$ *except* for having orthonormal columns, then this optimization problem

$$\operatorname*{argmax}_{\mathbf{Y}} \operatorname{trace}(\mathbf{Y}^T K \mathbf{Y})$$

over this multi-vector Rayleigh quotient may be maximized by assigning $\mathbf{Y}$'s columns as the eigenvectors corresponding to the k-largest eigenvalues of K. This eigenvector matrix is a relaxed "clustering" in that we have relaxed the requirements for the special structure of $\mathbf{Y}$ listed in Section 4.2 that ensured it corresponded to some discrete clustering $\mathbf{y}$.

4.4.3 Discretized Spectral Clustering

Discrete is a discretized spectral method via Bach and Jordan post-processing [1], and is a combination of the previous methods: once we have our eigenvector matrix $\bar{\mathbf{Y}}$, we cluster $\bar{K} = \bar{\mathbf{Y}}\bar{\mathbf{Y}}^T$ with point-incremental k-means to find a discrete $\mathbf{y}$.

5. Theoretical Analysis

Structural SVMs have three major important theoretical characteristics, including polynomial time termination in the number of iterations of Algorithm 1, correctness insofar as Algorithm 1 solves OP 1, and that $\frac{1}{n}\sum_{i=1}^{n} \xi_i$ upper bounds empirical risk [19]. We will now discuss how far they hold for supervised k-means algorithms.

There is one subtle but important point that arises from using approximations in the separation oracle: the known performance guarantees for Algorithm 1 are known to apply only to the case where the separation oracle $\text{argmax}_{\mathbf{y}\in\mathcal{Y}} H(\mathbf{y})$ is calculated exactly [19]. In Section 4.4 we constructed our separation oracle from a clustering algorithm, but because clustering algorithms are approximations, this may not find the globally optimal $\mathbf{y}$. What can we still guarantee about our supervised k-means algorithms?

Consider the space of possible clusterings $\mathcal{Y}$ for training example $(\mathbf{x}_i, \mathbf{y}_i)$. During training, the ideal clusterer separation oracle would find the true maximizing clustering $\mathbf{y}^* = \text{argmax}_{\mathbf{y}\in\mathcal{Y}} \langle \mathbf{w}, \mathbf{\Psi}(\mathbf{x}_i, \mathbf{y})\rangle + \Delta(\mathbf{y}_i, \mathbf{y})$. (To reiterate, under this ideal case, Algorithm 1 is guaranteed to solve OP 1.) However, this ideal is unrealizable. So what happens when we use one of our approximations?

Let us first consider polynomial time termination. The polynomial time termination guarantee still holds, since the proof does not depend on the quality of the solution, but rather on the idea that any constraint violated by more than ϵ must increase the objective by some minimum amount [19].

Correctness and empirical risk are less easy to deal with. The separation oracles can be divided into two broad categories according to what they do solve, depending on whether they use the discrete clusterers Iterative/Discrete, or relaxed Spectral.

The methods Iterative and Discrete may return some suboptimal clustering, i.e., some clustering $\hat{\mathbf{y}}$ such that

$$\langle \mathbf{w}, \mathbf{\Psi}(\mathbf{x}_i, \hat{\mathbf{y}})\rangle + \Delta(\mathbf{y}_i, \hat{\mathbf{y}}) < \langle \mathbf{w}, \mathbf{\Psi}(\mathbf{x}_i, \mathbf{y}^*)\rangle + \Delta(\mathbf{y}_i, \mathbf{y}^*).$$

In such a suboptimal case, constraints violated by more than ϵ in OP 1 may go undetected by Algorithm 1, leading to termination with a solution infeasible in OP 1. In other words, the problem becomes underconstrained.

The method Spectral is a very different animal. Rather than searching $\mathcal{Y}$ for local maxima, it instead searches some relaxed $\overline{\mathcal{Y}}$ space which it can efficiently search for a global maximum. In this case, $\overline{\mathcal{Y}}$ is the space of all indicator matrices $\mathbf{Y}$ where the special structure of entries described in Section 4.4.2 is abandoned, save for the requirement of orthonormal columns. More to the point, $\mathcal{Y} \subset \overline{\mathcal{Y}}$, and because the separation oracle searches over $\overline{\mathcal{Y}}$, at the end of Algorithm 1 we not only shall have all constraints in OP 1 respected, but additional constraints from outputs $\mathbf{Y} \in (\overline{\mathcal{Y}} - \mathcal{Y})$. The solution is feasible but probably suboptimal in OP 1. The problem becomes overconstrained.

Either underconstrained or overconstrained learning has its unique costs. With underconstrained learning, since constraints in OP 1 may be violated, slack no longer bounds empirical risk, thus eroding one of the basic principles of SVM learning. On the other hand, with overconstrained learning, Algorithm 1 solves a problem which accounts for outputs that would never arise from a discrete clustering algorithm, thus unnecessarily ruling out parameterizations $\mathbf{w}$ which may yield superior performance. It is unclear theoretically whether either way is better, so our experiments shall provide an empirical evaluation of both underconstrained and overconstrained learning.

6. Empirical Analysis

We implemented supervised k-means clustering with the *SVM*^{python} structural SVM package [9]. The module's code, instructions and examples of use, as well as the datasets that we used in our experiments, are accessible from:

`http://www.cs.cornell.edu/~tomf/projects/supervisedkmeans/`.

To empirically analyze our methods, we compare it to conventionally trained and untrained clusterers, and also provide comparisons of our methods using underconstrained and overconstrained learning on real and synthetic datasets. Parameterizations $\mathbf{w}$ and pairwise vectors ψ_{ij} are unconstrained as outlined in Section 3.2, i.e., not requiring $K \succeq 0$.

In all experiments, pairwise feature vectors ψ_{ij} are composed from "node" features vectors $\bar{\psi}_i, \bar{\psi}_j \in \mathbb{R}^{N_n}$ and an explicitly provided pairwise feature vector $\bar{\psi}_{ij} \in \mathbb{R}^{N_p}$ such that

$$\psi_{ij} = \begin{bmatrix} \bar{\psi}_i \circ \bar{\psi}_j \\ \bar{\psi}_{ij} \end{bmatrix}.$$

Pairwise feature vectors ψ_{ij} are in $\mathbb{R}^N$ where $N = N_n + N_p$, and correspondingly we have $\mathbf{w} \in \mathbb{R}^N$. Some datasets evaluated have no node or explicit pairwise features, i.e., sometimes $N_n = 0$ or $N_p = 0$.

We shall provide a web page with our software for download, as well as the datasets shown in this paper.

6.1. Datasets

We used three general "families" of datasets in our empirical analysis, from which we drew one or more specific evaluation datasets. The datasets are listed in Table 1.

6.1.1 News Dataset

News is a dataset related to the news article clustering dataset of [10]. The sets of items and partitioning was collected through trawling Google News for one day and extracting the text of news articles from the linked news sites. During any particular day, there are many different topics or stories. The set of articles in a particular story for a day form each of the example clusters. Each area (Google News has seven major areas: Business, Entertainment, Health, Nation, Sports, Technology, World) serves as an example clustering, with the stories forming the clusters, and the articles

Table 1. Dataset statistics, including number of example clusterings n, number of clusters k in each example clustering, average number of points m in the clusterings, node features N_n, and pairwise features N_p. (The SSVM learns $N = N_n + N_p$ weights in $\mathbf{w}$.)

DATASET	n	k	AVG. m	N_n	N_p
WEBKB-L	4	6	1041	50397	100796
WEBKB-N	4	6	1041	41131	0
NEWS 8-1	7	10	150	0	30
NEWS 8-2	7	10	150	0	30
NEWS 8-4	7	10	150	0	30
SYNTH	5	5	100	0	750

as the cluster elements. The data is expressed as a pairwise feature vector between articles, where each feature is the cosine similarity of TFIDF weighted token vectors, where these token vectors are unigrams, bigrams, and trigrams of text in the title, article text, and quoted sections of the article text, in both original and Porter stemmed versions of the features, for 30 features in all. We sampled from three days (August 1, 2, and 4 of 2004) to get three datasets (News 8-1, News 8-2, and News 8-4).

6.1.2 WebKB Dataset

WebKB consists of web pages retrieved from the computer science departments of four universities, labeled as being a course web page, faculty page, student page, etc [5]. It is often used in classification and multiclass classification tasks that seek to exploit the link structure among the web documents. In our experiment, we effectively turned this into two closely related datasets.

One of these datasets contains only node features (WebKB-N) as TFIDF-scaled unigram word count vectors. There are no pairwise features.

The other dataset (WebKB-L) contains these word count features and additional features relating to the relationships among these documents, and also critically a pairwise feature vector with two regions, corresponding to documents where one document links to another, and another where both are linked from the same document (co-citation). If documents are linked or co-cited, the respective region in the pairwise feature vector will contain the componentwise product of the node features, plus a single 1 indicator feature. If they are not linked or co-cited, the corresponding region is zeroed.

Though this is naturally a classification rather than a clustering problem, as we know what classes will occur in our test data a priori, it nonetheless serves as an appropriate test bed for our supervised clustering algorithms as well.

Table 2. Range of C values tested during the LOO search for training hyperparameters. All powers of ten between and including these endpoints were considered.

DATASET	LOW C	HIGH C	DATASET	LOW C	HIGH C
WEBKB-L	$1{\cdot}10^{-1}$	$1{\cdot}10^{4}$	NEWS	$1{\cdot}10^{0}$	$1{\cdot}10^{5}$
WEBKB-N	$1{\cdot}10^{0}$	$1{\cdot}10^{5}$	SYNTH	$1{\cdot}10^{-2}$	$1{\cdot}10^{3}$

6.1.3 Synth Dataset

Synth is a synthetic dataset meant to emphasize the importance of some features being harmful and others helpful, in the face of significant noise. It was generated in this way: there are 5 clusters, each with 20 points. Between every pair of the 100 points is a pairwise feature vector. This pairwise feature vector is comprised of 15 "regions" (one for each possible cluster pair), each region with 50 features (so 750 pairwise features total). For a pair of points in clusters i and j, the feature "region" corresponding to i, j will have 5 of the 50 features active. Also, noise is introduced for each pairwise feature vector[2]: instead of consistently indexing the region (i, j), it will 20% of the time replace i with a random cluster (so 16% of the time it will differ from i), and the same for j. So, only about 70.5% of pairwise vectors have the "correct" index. Only one dataset was generated.

6.2. Experimental Setup

To evaluate performance, we trained k-means parameterizations on our dataset. For each dataset of n clustering examples, we ran n experiments, where each clustering was taken in turn as the single example "test set" with the $n - 1$ remaining clusterings as the training set. For each experiment, LOO cross validation was used on the $n - 1$ size training set to choose the two training hyperparameters: C (values drawn from a sample of powers of 10 seen in Table 2), and which classifier to use as the final predictor (Iterative, Spectral, or Discrete).

The parameterizations were trained with Iterative and Spectral separation oracle supervised k-means trainers. In addition to these supervised k-means clustering methods, we have two baselines.

Pair is a model training method based on binary classifiers by taking all pairwise feature vectors, considering whether the associated pair is in the same cluster, and treating it as a binary classification problem trained for accuracy with an SVM. During classification, entries in the similarity matrix K are outputs of the learned binary classifier. This

[2]Without noise, learned clusterers produced perfect clusterings. While useful as a sanity check, it makes for uninteresting comparisons.

style of supervised clustering using binary classifiers has been successfully used in work on noun-phrase coreference resolution [15]. The resulting training method differs from supervised k-means clustering insofar as the clustering procedure and desired Δ are not considered in training, but it will still try to increase or decrease the similarity of pairs in or out of the same cluster, respectively. Hyperparameters (C and clusterer in prediction) were selected in an identical fashion to supervised k-means clustering.

None is a second baseline, which consists of Iterative classification with all equal weights, i.e., no training at all.

6.3. Clustering Accuracy

Table 3 details the loss figures resulting from training the clusterer with the Iterative and Spectral separation oracle (columns Iterative and Spect), training the clusterer with the pairwise binary classifier (column Pair), and with no training (column None). While loss Δ values can reach 100, a more reasonable upper bound is $\frac{k-1}{k} \cdot 100$, the loss resulting from putting all points together in 1 cluster.

6.3.1 Supervised Clustering vs. Pairwise/Untrained

How do efforts to do any supervised k-means clustering compare against the more naive pairwise binary training? On the WebKB-L, WebKB-N and News datasets, the performance gains from structural SVM training in Δ figures are quite dramatic, and both Iterative and Spectral trained supervised k-means clustering methods outperform these baselines on these datasets every time.

The relationship on Synth is somewhat different: while there are differences, the pairwise trained model even "wins" once (testing on cluster 3), and the extent to which each either class of supervised k-means clustering models wins is not conclusively better statistically speaking. Why does this happen? One important power of supervised clustering methods is their ability to exploit cluster structure: two items $i, j \in \mathbf{x}$ with low similarity K_{ij} can still be in the same cluster owing to the effect of other items in $\mathbf{x}$. In contrast, the baseline pairwise classifier treats all judgments on pairwise ϕ_{ij} independently. However, since all ϕ_{ij} are generated independently in the synthetic dataset and there is no long range dependency structure to exploit, pairwise classification for training $\mathbf{w}$ works fine.

The untrained model does quite poorly in Synth, but this is expected since the dataset was generated specifically to contain large numbers of pairwise features correlated negatively with co-cluster membership.

6.3.2 Discrete Iterative vs. Relaxed Spectral

How does discrete Iterative compare against the relaxed Spectral when used as a separation oracle during training?

Table 3. Loss Δ on various datasets (lower is better). The left columns identify the dataset and the particular clustering used as the test dataset in the corresponding row.

Dataset	Test Clustering	Iter.	Spect	Pair	None
WebKB-L	Cornell	45.3	53.3	79.7	74.7
	Texas	59.8	56.7	78.9	72.8
	Washington	53.1	46.6	60.6	76.2
	Wisconsin	47.3	60.2	81.1	77.5
WebKB-N	Cornell	63.0	61.4	74.8	78.6
	Texas	69.9	56.8	75.5	78.7
	Washington	68.8	58.2	74.9	78.3
	Wisconsin	72.6	66.2	77.0	78.6
News 8-1	Business	23.7	20.6	45.2	49.5
	Entertainment	12.7	22.2	53.0	25.9
	Health	28.1	28.7	57.4	38.8
	Nation	3.8	3.8	40.2	14.6
	Sports	15.2	14.3	47.6	59.9
	Technology	35.9	30.4	51.7	37.3
	World	3.7	2.4	41.7	62.1
News 8-2	Business	3.6	4.6	34.1	63.8
	Entertainment	22.7	9.5	40.1	22.8
	Health	20.4	20.4	48.4	43.9
	Nation	24.6	23.7	47.4	60.6
	Sports	20.2	15.8	59.3	57.0
	Technology	16.1	13.8	48.3	41.3
	World	12.2	11.9	50.5	70.4
News 8-4	Business	19.7	14.9	42.7	33.5
	Entertainment	4.6	6.3	46.8	32.4
	Health	15.0	16.2	51.7	32.1
	Nation	19.4	20.3	41.2	30.0
	Sports	19.0	19.0	55.6	54.7
	Technology	5.8	11.6	46.4	37.6
	World	4.8	5.8	39.6	39.3
Synth	1	43.3	55.6	48.1	74.7
	2	53.4	58.7	54.7	74.7
	3	56.0	56.7	55.2	74.7
	4	39.3	59.5	43.9	74.7
	5	40.3	63.4	49.1	74.7

We use non-parametric tests like Fisher sign or Wilcoxon signed-rank tests. While the loss figures are not independent since they result from shared training sets, we accept these non-parametric tests as an imperfect measure that nevertheless gives some indication of difference.

Results of the comparison are seen in Table 4. These results reflect the feeling one might get glancing at Table 3: there is no clear winner in WebKB or News. The exception is the Synth synthetic data set, where the Iterative trained model appears to yield superior performance.

6.3.3 WebKB-N versus WebKB-L

The WebKB-L dataset differs from WebKB-N in that it contains pairwise features relevant to the hyperlink structure in the corpus, whereas WebKB-N are straightforward document vectors. Each of the 8 supervised k-means clustering WebKB-L trained models outperform their corresponding WebKB-N trained model. While 8 wins to 0 losses is statistically significant under a sign test, these loss Δ figures are not independent; nevertheless, the magnitude of the differ-

Table 4. Counts of the times within Table 3 the Iterative trained model won, tied, or lost versus the Spectral trained model respectively.

DATASET	WIN	TIE	LOSE	W	$n_{s/r}$	$P_{1\text{-TAIL}}$
WEBKB-L	2	0	2	4	4	>0.05
WEBKB-N	0	0	4	4	4	>0.05
NEWS	8	3	10	30	18	0.2611
SYNTH	5	0	0	15	5	0.05

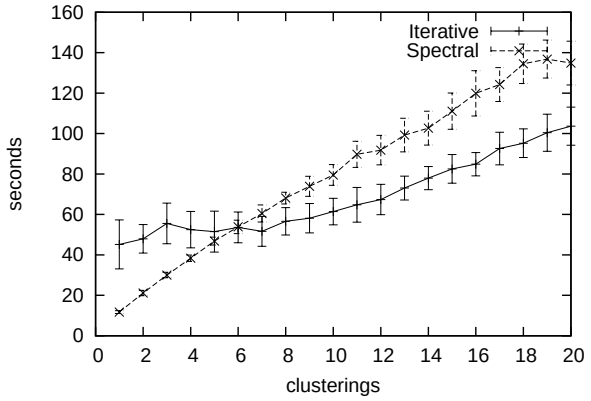

Figure 1. Train time vs. number of example clusterings n in the training set.

ences, always over 10 in the case of Iterative trained models, suggests a substantial gain. As the usefulness of exploiting hyperlink structure in WebKB is a feature of most papers featuring this dataset, it is important that our methods are able to handle definitions of these general pairwise features.

6.4. Computation Time

Clustering performance aside, how does training time depend on characteristics of the dataset? To answer this question empirically, we took the basic Synth dataset described in Section 6.1. The basic dataset has 5 clustering examples, 5 clusters, 750 features, and 100 points. To test the algorithms in a controlled way, we varied each of these characteristics (examples, clusters, features, points), and trained over 20 training sets to test the time it took to train a model. Results are reported for both Iterative and Spectral clustering. The regularization parameter $C = 10^4$ was constant in all training methods.

As we increase the number of training example clusterings in our training set, Figure 1 reveals a relationship linear for Spectral and approximately linear for Iterative. That training time is linear in the number of training examples is expected due to previous theoretical results [12].

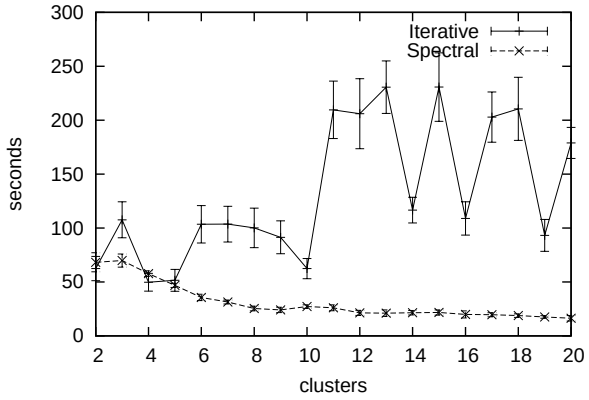

Figure 2. Train time vs. number of clusters k in each example.

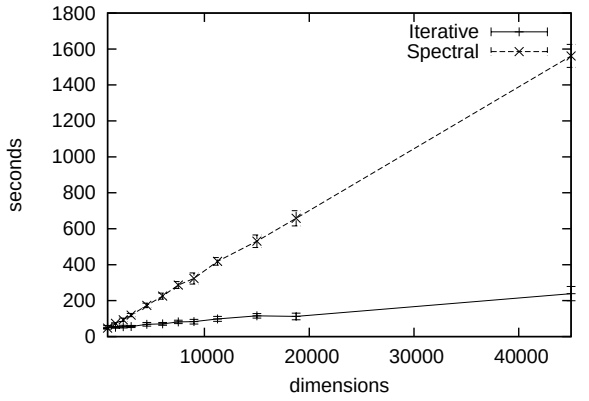

Figure 3. Train time vs. number of features N.

Figure 2 shows that increasing the number of clusters while holding other statistics constant leads to a steady decrease in training time for Spectral trained methods. This appears to be a symptom of the difficulty of learning this dataset: the number of points and dimensions is constant, but spread over an increasing number of clusters in each example. Consequently the best hypothesis that can be reasonably extracted from the provided data becomes weaker, and fewer iterations are required to converge. The Iterative method, on the other hand, often takes longer. Logs reveal this is due to one or two iterations where Iterative as separation oracle took a very long time to converge, explaining the unstable nature of the curve.

Figure 3 shows a linear relationship of number of features versus training time. This linear time relationship is unsurprising given that computing similarities and $\mathbf{\Psi}$ is linear in the number of features. Spectral is slower than Iterative both on account of the speed of the clustering algo-

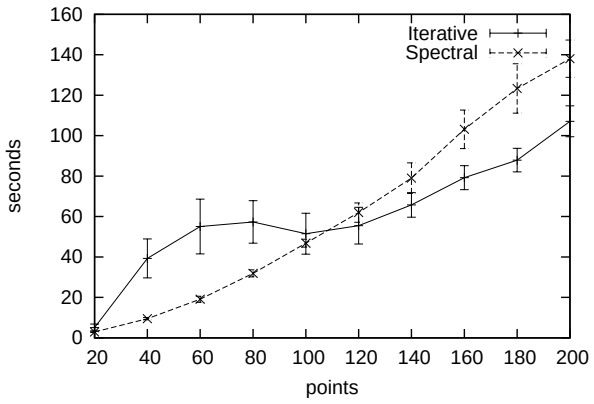

Figure 4. Train time vs. number of points within each cluster m/k.

rithm, as well as requiring more iterations of Algorithm 1.

Let's now examine how training time varies with the number of points in each cluster. Figure 4 shows Spectral time complexity as a straightforward polynomially increasing curve (due to the LAPACK `DSYEVR` eigenpair procedure working on steadily larger matrices). The Iterative trained classifier also tends to increase with number of points, with a hump on lower numbers of points arising from Iterative clustering often requiring more time for the clusterer to converge on smaller datasets, a tendency reversed as more points presumably smooth the search space.

One theme seen throughout is that the timing behavior of relaxed spectral training is very predictable relative to the discrete k-means training. Considering the somewhat unpredictable nature of local search versus largely deterministic matrix computations, it is unsurprising to see the latter's relative stability carry over into model training time.

7. Conclusions

We provided a means to parameterize the popular canonical k-means clustering algorithm based on learning a similarity measure between item pairs, and then provided a supervised k-means clustering method to learn these parameterizations using a structural SVM. The supervised k-means clustering method learns this similarity measure based on a training set of item sets and complete partitionings over those sets, choosing parameterizations optimized for good performance over the training set.

We then theoretically characterized the learning algorithm, drawing a distinction between the iterative local search k-means clustering method and the relaxed spectral relaxation, as leading to underconstrained and overconstrained supervised k-means clustering learners, respectively. Empirically, the supervised k-means clustering algorithms exhibited superior performance compared to naive pairwise learning or unsupervised k-means. The underconstrained and overconstrained supervised k-means clustering learners compared to each other exhibited different performance, though neither was clearly consistently superior to the other. We also characterized the runtime behavior of both the supervised k-means clustering learners through an empirical analysis on datasets with varying numbers of examples, clusters, features, and items to cluster. We find training time is linear or better in the number of example clusterings, clusters per example, and number of features.

8. Acknowledgments

This work was supported under NSF Award IIS-0713483 "Learning Structure to Structure Mapping," and through a gift from Yahoo! Inc.

References

[1] F. R. Bach and M. I. Jordan. Learning spectral clustering. In *NIPS*. MIT Press, 2003.

[2] N. Bansal, A. Blum, and S. Chawla. Correlation clustering. *Machine Learning*, 56(1-3):89–113, 2002.

[3] S. Basu, M. Bilenko, and R. J. Mooney. A probabilistic framework for semi-supervised clustering. In *ACM SIGKDD-2004*, pages 59–68, August 2004.

[4] M. Bilenko, S. Basu, and R. J. Mooney. Integrating constraints and metric learning in semi-supervised clustering. In *ICML*, New York, NY, USA, 2004. ACM Press.

[5] M. Craven, D. DiPasquo, D. Freitag, A. McCallum, T. Mitchell, K. Nigam, and S. Slattery. Learning to extract symbolic knowledge from the world wide web. In *AAAI/IAAI*, pages 509–516, Menlo Park, CA, USA, 1998.

[6] T. De Bie, M. Momma, and N. Cristianini. Efficiently learning the metric using side-information. In *ALT2003*, volume 2842, pages 175–189. Springer, 2003.

[7] I. S. Dhillon, Y. Guan, and J. Kogan. Iterative clustering of high dimensional text data augmented by local search. In *ICDM*, page 131. IEEE Computer Society, 2002.

[8] I. S. Dhillon, Y. Guan, and B. Kulis. A unified view of kernel k-means, spectral clustering and graph cuts. Technical Report TR-04-25, UT Austin Dept. of CS, 2005.

[9] T. Finley. SVM^{python}, 2007. Software at `http://www.cs.cornell.edu/~tomf/svmpython2/`.

[10] T. Finley and T. Joachims. Supervised clustering with support vector machines. In *ICML*, 2005.

[11] P. Haider, U. Brefeld, and T. Scheffer. Supervised clustering of streaming data for email batch detection. In *ICML*, pages 345–352, New York, NY, USA, 2007. ACM.

[12] T. Joachims. Training linear svms in linear time. In *KDD*, pages 217–226, New York, NY, USA, 2006. ACM.

[13] J. M. Kleinberg. Hubs, authorities, and communities. *ACM Comput. Surv.*, page 5.

[14] G. R. G. Lanckriet, N. Christianini, P. L. Bartlett, L. E. Ghaoui, and M. I. Jordan. Learning the kernel matrix with semi-definite programming. In *ICML*, pages 323–330, 2002.

[15] V. Ng and C. Cardie. Improving machine learning approaches to coreference resolution. In *ACL-02*, pages 104–111, 2002.

[16] W. M. Rand. Objective criteria for the evaluation of clustering methods. *Journal of the American Statistical Association*, 66(366):846–850, 1971.

[17] B. Taskar, V. Chatalbashev, and D. Koller. Learning associative Markov networks. In *ICML*, page 102, New York, NY, USA, 2004. ACM.

[18] B. Taskar, C. Guestrin, and D. Koller. Max-margin Markov networks. In *NIPS 16*. 2003.

[19] I. Tsochantaridis, T. Hofmann, T. Joachims, and Y. Altun. Support vector machine learning for interdependent and structured output spaces. In *ICML*, 2004.