Supervised Clustering with Support Vector Machines

Thomas Finley TOMF@CS.CORNELL.EDU
Thorsten Joachims TJ@CS.CORNELL.EDU
Department of Computer Science, Cornell University, Ithaca, NY 14853 USA

Abstract

Supervised clustering is the problem of training a clustering algorithm to produce desirable clusterings: given sets of items and complete clusterings over these sets, we learn how to cluster future sets of items. Example applications include noun-phrase coreference clustering, and clustering news articles by whether they refer to the same topic. In this paper we present an SVM algorithm that trains a clustering algorithm by adapting the item-pair similarity measure. The algorithm may optimize a variety of different clustering functions to a variety of clustering performance measures. We empirically evaluate the algorithm for noun-phrase and news article clustering.

1. Introduction

Clustering algorithms accept a set of items and produce a partitioning of that set. Two items in the same partition should be more similar than two items not in the same partition. However, sometimes what "similar" means is unclear. When clustering news articles, a user could want articles clustered either by topic, or by author, or by language, etc. A clustering algorithm may not produce desirable clusterings without additional information from the user. One way to provide this information is through manual adjustment of the clustering algorithm or similarity measure. However, manual adjustment of similarity can be difficult if articles are described by many attributes of indeterminate relevance. While users often cannot easily specify the similarity measure, they can often provide examples of what constitutes the "correct" clustering of a set. We take the approach of learning to cluster from user provided example clusterings.

Appearing in *Proceedings of the 22nd International Conference on Machine Learning*, Bonn, Germany, 2005.

In this paper, we present an SVM algorithm for supervised clustering. This algorithm learns an item-pair similarity measure to optimize performance of correlation clustering (Bansal et al., 2002) on a variety of performance measures. Since clustering is NP-hard, we present and empirically evaluate approximation methods to use when learning.

2. Supervised Clustering Task

Clustering is sometimes applied to multiple sets of items, with each set being clustered separately. For example, in the noun-phrase coreference task, a single document's noun-phrases are clustered by which noun-phrases refer to the same entity (MUC-6, 1995), and in news article clustering, a single day's worth of news articles are clustered by topic. In our method, users provide complete clusterings of a few of these sets to express their preferences, e.g., provide a few complete clusterings of several documents' noun-phrases, or several days' news articles. From these training examples, we learn to cluster future sets of items.

In this setting, the learning algorithm receives a set S of n training examples $(\mathbf{x}_1, \mathbf{y}_1), \ldots, (\mathbf{x}_n, \mathbf{y}_n) \in \mathcal{X} \times \mathcal{Y}$, all drawn i.i.d. from a distribution $P(X, Y)$. $\mathcal{X}$ is the set of all possible sets of items and $\mathcal{Y}$ is the set of all possible clusterings (partitionings) of these sets. For any $(\mathbf{x}, \mathbf{y})$, $\mathbf{x} = \{x_1, x_2, \ldots, x_m\}$ is a set of m items, and $\mathbf{y} = \{y_1, y_2, \ldots, y_c\}$ with $y_i \subseteq \mathbf{x}$ is the partitioning of $\mathbf{x}$ into c clusters. The goal is to learn a clustering function $h : \mathcal{X} \rightarrow \mathcal{Y}$ that can accurately cluster new sets of items.

Given a loss function that compares two clusterings $\Delta : \mathcal{Y} \times \mathcal{Y} \rightarrow \mathbb{R}$, the training error for a clustering function h on an example $(\mathbf{x}, \mathbf{y})$ is $\Delta(h(\mathbf{x}), \mathbf{y})$. The goal is to find h to minimize risk $Err_P(h) = \int_{\mathcal{X} \times \mathcal{Y}} \Delta(h(\mathbf{x}), \mathbf{y})\, dP(\mathbf{x}, \mathbf{y})$, approximated by empirical risk $Err_S(h) = \frac{1}{n} \sum_{i=1}^{n} \Delta(h(\mathbf{x}_i), \mathbf{y}_i)$.

This work shares many similarities with the semi-supervised clustering, which attempts to form desirable clusterings by taking user information into ac-

count, typically of the form "these items do/do not belong together." Some supervised clustering methods modify a clustering algorithm so it satisfies constraints (Aggarwal et al., 1999; Wagstaff et al., 2001). Others, including ours, learn a similarity measure that will tend to produce desired clusterings (Bilenko et al., 2004; De Bie et al., 2003; Lanckriet et al., 2004). Some methods do both (Basu et al., 2004).

Modifying the similarity measure has some intuitive appeal: if you want news articles clustered by topic, a great clustering method using author similarity will probably produce worse results than a mediocre clustering method using topic similarity.

3. Supervised Clustering vs. Pairwise Classification

To learn a similarity measure for clustering, one approach is to use a binary classifier. Take all pairs of items in all training sets, describe each pair in terms of a feature vector, and let positive examples be those pairs in the same cluster and negative examples be those pairs in different clusters. When you want to cluster a new set of items, run all pairs through the classifier. The output values are the pairwise similarity values; positive and negative outputs indicate a pair should or should not be in the same cluster, respectively. Then, cluster based on these output similarities. We discuss three problems with this approach.

First, some supervised clustering tasks are associated with a performance measure, e.g., the model-theoretic MITRE score for MUC noun-phrase coreference (Vilain et al., 1995). A pairwise classifier may not optimize for the correct measure in the likely event that perfectly learning the "same/different cluster" concept is impossible with the given features.

Second, in clustering applications often the number of pairs in a cluster is relatively small, e.g., only 1.8% of pairs in the MUC-6 training set represent items in the same cluster. The training imbalance could lead to understatement of pairwise similarity.

Third, and most important, a pairwise classifier that assumes pairs are i.i.d. cannot take advantage of dependencies between item pairs. Consider this small document: "$(\text{Bush})_{np_1}$ ate some tacos. ... $(\text{President Bush})_{np_2}$ likes tacos. ... $(\text{He})_{np_3}$ said tacos are good food for himself, the $(\text{president})_{np_4}$." We want np_1, np_2, np_3, np_4 to be in the same cluster. With pairwise features as in (Soon et al., 2001) or (Ng & Cardie, 2002), it is probably easy to learn that (np_1, np_2), (np_2, np_3), and (np_2, np_4) are coreferent and should have positive similarity, but difficult to learn that pairs (np_1, np_3), (np_1, np_4), and (np_3, np_4) are coreferent. However, a clustering algorithm would still come up with the correct clustering even if the hard pairs were kept as unknowns (perhaps with similarity kept slightly less than 0) and only the easy pairs were learned. A learner could exploit these transitive dependencies to learn more effectively, since insisting that we learn the difficult relationships may diminish the hypothesis's overall effectiveness.

To overcome these problems, some methods employ heuristics to train the classifier only on selected item pairs. The hope is that the heuristic will compensate for these weaknesses. Work in (Cohen & Richman, 2001) adapts the canopy technique for clustering (McCallum et al., 2000) to supervised clustering. Given an existing similarity measure, the pairwise classifier trains only on item-pairs whose similarity is within a certain interval. An approach specific to noun-phrase coreference appears in (Ng & Cardie, 2002). Each noun-phrase x_b and its closest preceding non-pronominal coreferent noun-phrase x_a form a positive training pair, while all non-coreferent noun-phrases in between x_a and x_b are paired with x_b as a negative training example. This approach is excellent for the NP coreference task, but was built with expert domain knowledge and is not applicable to other tasks.

Other approaches avoid these types of heuristics and multiple step approaches entirely, instead learn to cluster directly, optimizing actual clustering performance instead of an inexact model of clustering performance. This removes the need to heuristically model the transitive dependencies between item pairs. We are aware of two existing methods. One is generative, modeling the probabilities of clusters under certain assumptions about the independence of attributes, the type of clustering function we use, and the type of loss that is used (Kamishima & Motoyoshi, 2003). The other is based on CRFs, and can be used with a variety of clustering functions, does not require the independence of attributes, but cannot optimize clusters with respect to a custom loss function (McCallum & Wellner, 2003). Our work is more closely related to the latter technique, except ours is motivated by a maximum margin approach rather than CRFs. This makes it straightforward to optimize with respect to a particular loss. Further, a maximum margin formulation does not require approximating a normalizing constant (i.e., a sum over all clusterings) like a CRF.

4. Supervised Clustering with SVMs

This section describes our supervised clustering algorithm. We define our model, summarize the structural

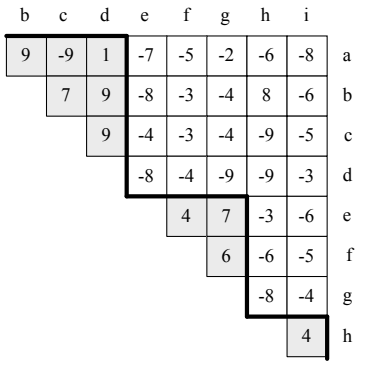

Figure 1. Correlation clustering on a matrix of similarities for items x_a through x_i, where shaded boxes indicate that a pair is considered to be in the same cluster.

SVM algorithm (Tsochantaridis et al., 2004), and then describe how to adapt the algorithm to clustering.

4.1. Model

In our supervised clustering method, we hold the clustering algorithm constant and modify the similarity measure so that the clustering algorithm produces desirable clusterings. Our similarity measure $Sim_{\mathbf{w}}$, parameterized by $\mathbf{w}$, maps pairs of items to a real number indicating how similar the pair is; positive values indicate the pair is alike, negative values, unalike. Each pair of different items $x_a, x_b \in \mathbf{x}$ has a feature vector $\phi(x_a, x_b) \equiv \phi_{a,b}$ to describe the pair. The similarity measure is $Sim_{\mathbf{w}}(x_a, x_b) = \mathbf{w}^T \phi_{a,b}$.

For our clustering method, we use *correlation clustering* (Bansal et al., 2002). The correlation clustering of a set of items $\mathbf{x}$ is the clustering $\mathbf{y}$ maximizing the sum of similarities for item pairs in the same cluster.

$$\operatorname*{argmax}_{\mathbf{y}} \sum_{y \in \mathbf{y}} \sum_{x_a, x_b \in y} Sim_{\mathbf{w}}(x_a, x_b) \tag{1}$$

As shown in Figure 1, pairs considered dissimilar can appear in the same cluster if the net effect of including them is positive (e.g., x_a and x_c), and pairs considered similar cannot be in the same cluster if the net effect of including them is negative (e.g., x_b and x_h).

We can rewrite the objective function as follows:

$$\sum_{y \in \mathbf{y}} \sum_{x_a, x_b \in y} Sim_{\mathbf{w}}(x_a, x_b) \tag{2}$$

$$= \sum_{y \in \mathbf{y}} \sum_{x_a, x_b \in y} \mathbf{w}^T \phi(x_a, x_b) \tag{3}$$

$$= \mathbf{w}^T \left(\sum_{y \in \mathbf{y}} \sum_{x_a, x_b \in y} \phi(x_a, x_b) \right) \tag{4}$$

The objective function is a linear product of $\mathbf{w}$, and a sum of ϕ vectors. Though we use correlation clustering, any clustering method with an objective function expressible as a linear product of $\mathbf{w}$ is acceptable.

4.2. Learning Algorithm

The structural SVM algorithm provides a general framework for learning with complex structured output spaces (Tsochantaridis et al., 2004). The method we present in the sequel is implemented in the software SVM^{struct} available from `http://svmlight.joachims.org/svm_struct.html`. We describe how to apply this method to supervised clustering. We refer to our method as SVM^{cluster} (SVM supervised clustering). The structural SVM algorithm solves this quadratic program:

$$\min_{\mathbf{w}, \xi} \frac{1}{2} \|\mathbf{w}\|^2 + C \sum_{i=1}^{n} \xi_i, \text{ s.t. } \forall i : \xi_i \geq 0, \tag{5}$$

$$\forall i, \forall \mathbf{y} \in \mathcal{Y} \setminus \mathbf{y}_i : \mathbf{w}^T \Psi(\mathbf{x}_i, \mathbf{y}_i) \geq \mathbf{w}^T \Psi(\mathbf{x}_i, \mathbf{y}) + \Delta(\mathbf{y}_i, \mathbf{y}) - \xi_i \tag{6}$$

Here, Expression 5 contains the typical SVM quadratic objective and slack constraints. Inequality 6 expresses the set of constraints that allows us to learn the desired hypothesis. This particular QP is called the SVM$_1^{\Delta m}$ program, i.e., slack norm is 1, and loss acts as the margin, like in (Taskar et al., 2003). Other similar QPs are described in (Tsochantaridis et al., 2004), but we use SVM$_1^{\Delta m}$ since it is more compatible with the correlation clustering algorithm we use in our experiments.

$\Delta(\mathbf{y}, \hat{\mathbf{y}})$ indicates a real valued loss between a true cluster $\mathbf{y}$ and a predicted cluster $\hat{\mathbf{y}}$. $\Delta(\mathbf{y}, \hat{\mathbf{y}}) = 0$ if $\mathbf{y} = \hat{\mathbf{y}}$, and Δ rises as the two clusters become more dissimilar. In our experimental section we use two loss functions Δ: a loss based on the MITRE precision and recall score for noun-phrase coreference, and a "pairwise" loss that counts the number of pairwise cluster relationships the clusterings disagree on. More details of these loss functions appear in Section 6.

The $\Psi(\mathbf{x}, \mathbf{y})$ function returns a combined feature representation of an input $\mathbf{x}$ and output $\mathbf{y}$. In the case of learning for correlation clustering,

$$\Psi(\mathbf{x}_i, \mathbf{y}) = \frac{1}{|\mathbf{x}_i|^2} \sum_{y \in \mathbf{y}} \sum_{x_a, x_b \in y} \phi(x_a, x_b) \tag{7}$$

Since $\mathbf{w}^T \Psi(\mathbf{x}_i, \mathbf{y})$ is the correlation clustering objective, for every training example $(\mathbf{x}_i, \mathbf{y}_i)$, and every possible wrong clustering $\mathbf{y}$, SVM^{cluster} finds the vector $\mathbf{w}$ to make the value of the objective for the *correct* clustering be greater than the value of the objective for this *incorrect* clustering by at least a margin of

the loss between $\mathbf{y}_i$ and $\mathbf{y}$. Note that $\sum_{i=1}^{n} \xi_i$ upper bounds the training loss.

The quadratic program (5-6) introduces a constraint for every possible wrong clustering of the set. Unfortunately, the number of wrong clusterings scales more than exponentially with the number of items. The approach in the structural SVM algorithm is to start with no constraints, and iteratively find the most violated constraint. This algorithm for $\text{SVM}_1^{\Delta m}$ is:

Input: $(\mathbf{x}_1, \mathbf{y}_1), \ldots, (\mathbf{x}_n, \mathbf{y}_n)$, C, ϵ
$S_i \leftarrow \emptyset$ for all $i = 1, \ldots, n$
repeat
  for $i = 1, \ldots, n$ **do**
    $H(\mathbf{y}) \equiv \Delta(\mathbf{y}_i, \mathbf{y}) + \mathbf{w}^T \Psi(\mathbf{x}_i, \mathbf{y}) - \mathbf{w}^T \Psi(\mathbf{x}_i, \mathbf{y}_i)$
    compute $\hat{\mathbf{y}} = \text{argmax}_{\mathbf{y} \in \mathcal{Y}} H(\mathbf{y})$
    compute $\xi_i = \max\{0, \max_{\mathbf{y} \in S_i} H(\mathbf{y})\}$
    if $H(\hat{\mathbf{y}}) > \xi_i + \epsilon$ **then**
      $S_i \leftarrow S_i \cup \{\hat{\mathbf{y}}\}$
      $\mathbf{w} \leftarrow$ optimize primal over $S = \bigcup_i S_I$
    end if
  end for
until no S_i has changed during iteration

By solving $\text{argmax}_{\mathbf{y}} H(\mathbf{y})$, the algorithm finds the clustering $\hat{\mathbf{y}}$ associated with the most violated constraint for $(\mathbf{x}_i, \mathbf{y}_i)$. Since H is the minimum necessary slack for $\hat{\mathbf{y}}$ under the current $\mathbf{w}$, if $H(\hat{\mathbf{y}}) > \xi_i + \epsilon$, the constraint is violated by more than ϵ, so we introduce the constraint and re-optimize. The algorithm repeats this process until no new constraints are introduced. (Tsochantaridis et al., 2004) proves the convergence and (Joachims, 2003) proves the correctness of the algorithm. We state the theorems but omit the proofs.

Theorem 1 *Let* $\bar{\Delta} = \max_{(\mathbf{x}_i, \mathbf{y}_i) \in S} (\max_{\mathbf{y}} \Delta(\mathbf{y}_i, \mathbf{y}))$ *and* $\bar{R} = \max_{(\mathbf{x}_i, \mathbf{y}_i) \in S} (\max_{\mathbf{y}} \|\Psi(\mathbf{x}_i, \mathbf{y}_i) - \Psi(\mathbf{x}_i, \mathbf{y})\|)$ *for a training sample* S*. Then, the structural SVM algorithm converges after introducing at most* $\max\left\{\frac{2n\bar{\Delta}}{\epsilon}, \frac{8Cn\bar{\Delta}\bar{R}^2}{\epsilon^2}\right\}$ *constraints.*

Theorem 2 *The algorithm returns an approximation with an objective less than or equal to QP (5-6)'s objective. All constraints are fulfilled within* ϵ*.*

For simplicitly, the previous discussion considered only the linear case where the pairwise similarity is the inner product of the pairwise feature vector ϕ with $\mathbf{w}$. However, in the dual $\mathbf{w}$ is some linear combination of all ϕ, so nonlinear mappings are possible through kernels as with a regular SVM. However, since deriving each pairwise similarity requires a kernel evaluation of the ϕ with every component of $\mathbf{w}$, the use of kernels in this particular problem appears to be impractical except for clustering over very small sets of items.

5. Approximate Inference

In this section we describe the difficulty of finding the most violated constraint in $\text{argmax}_{\mathbf{y}} H(\mathbf{y})$ and suggest methods for approximately finding the most violated constraint with two clustering methods.

Consider the cost function H.

$$H(\mathbf{y}) \equiv \Delta(\mathbf{y}_i, \mathbf{y}) + \mathbf{w}^T \Psi(\mathbf{x}_i, \mathbf{y}) - \mathbf{w}^T \Psi(\mathbf{x}_i, \mathbf{y}_i) \quad (8)$$

The last term is a constant, and so can be ignored since it does not change the maximum. The cost function is a loss Δ between the true labeling $\mathbf{y}_i$ and prediction $\mathbf{y}$ plus the correlation clustering objective function. Finding the $\mathbf{y}$ to maximize the correlation clustering objective function is NP-complete (Bansal et al., 2002), and the addition of the loss is unlikely to help tractability, so finding $\text{argmax}_{\mathbf{y}} H(\mathbf{y})$ is intractable. Fortunately algorithms exist for *approximately* maximizing these clustering objectives, and $\text{argmax}_{\mathbf{y}} H(\mathbf{y})$. These approximations will not solve $\text{argmax}_{\mathbf{y}} H(\mathbf{y})$ exactly, but are possibly close enough that $\text{SVM}^{cluster}$ still learns something reasonable. Applying a similar margin maximizing framework to perform collective classifications, (Taskar et al., 2004) inferred approximated constraints with a linear relaxation. Approximate inference may work for clustering as well.

How are the termination and the correctness of the structural SVM algorithm affected if one uses approximate maximization of $H(\mathbf{y})$? The proof of polynomial time termination in Theorem 1 still holds. The proof does not depend upon finding $\text{argmax}_{\mathbf{y}} H(\mathbf{y})$ exactly, but rather that new introduced constraints are violated by more than ϵ, and so cause the quadratic objective to increase by a minimum amount. However, the proof of correctness for Theorem 2 no longer holds. Without finding $\text{argmax}_{\mathbf{y}} H(\mathbf{y})$ exactly, either violated constraints may remain undetected, or the objective may be raised. We consider two approximations: a simple greedy approach C_G, and a real relaxation of correlation clustering C_R (Demaine & Immorlica, 2003). We consider how they impact the correctness of the algorithm in the sequel, and later in Section 7 empirically evaluate their performance.

5.1. Greedy Approximation, C_G

To greedily approximate $\text{argmax}_{\mathbf{y}} H(\mathbf{y})$, start with an initial partitioning $\mathbf{y}$ with every item of $\mathbf{x}$ in its own cluster. Repeatedly find and merge the two clusters $y_i, y_j \in \mathbf{y}$ that would maximally increase $H(\mathbf{y})$. Halt and return $\mathbf{y}$ when no merge increases $H(\mathbf{y})$.

Corollary 3 *The greedy approximation* C_G *leads to an underconstrained program with respect to QP (5-6),*

with an objective value not greater than (5-6)'s objective.

Suppose the true $\text{argmax}_{\mathbf{y}} H(\mathbf{y})$ is $\hat{\mathbf{y}}$, but the approximate $\text{argmax}_{\mathbf{y}} H(\mathbf{y})$ found with this greedy approximation is $\mathbf{y}^*$, so that $H(\hat{\mathbf{y}}) \geq H(\mathbf{y}^*)$. Some constraints from the full QP (5-6) violated by more than ϵ might not be found and introduced. This leads to an underconstrained program that may find a solution not allowed by QP (5-6). Since the underconstrained program's feasible region contains the solution to (5-6), the objective cannot be greater than (5-6)'s objective.

5.2. Relaxation Approximation, C_R

An alternative to simple greedy approximation is a real relaxation approximation, either in the form of a linear (Demaine & Immorlica, 2003) or semidefinite program (Swamy, 2004). We use a linear program equivalent to (Demaine & Immorlica, 2003). In the LP, each pair of items $x_a, x_b \in \mathbf{x}$ has a corresponding variable $e_{a,b}$ indicating the degree to which x_a and x_b are in the same cluster. For the collection of the $e_{a,b}$ variables $\mathbf{e}$, the LP is:

$$\max_{\mathbf{e}} \sum_{e_{a,b} \in \mathbf{e}} e_{a,b} \cdot (\mathbf{w}^T \phi_{a,b}) \tag{9}$$

$$\text{s.t.} \quad e_{a,b} \in [0,1],\ e_{a,b} = e_{b,a},\ e_{a,b} + e_{b,c} \geq e_{a,c} \tag{10}$$

For some types of Δ losses, we can also include the loss function Δ in the LP objective function so that solving this LP solves the relaxed $\text{argmax}_{\mathbf{y}} H(\mathbf{y})$. We can use the relaxed solution in the constraints by extending Equation 7 to incorporate a relaxed solution $\mathbf{e}$ instead of the discrete solution $\mathbf{y}$.

$$\Psi(\mathbf{x}, \mathbf{e}) = \frac{1}{\mathbf{x}} \sum_{e_{a,b} \in \mathbf{e}} e_{a,b} \cdot \phi_{a,b} \tag{11}$$

Note that Equation 11 is equivalent to Equation 7 if all $e_{a,b}$ are integral.

Corollary 4 *The relaxed approximation C_R leads to an overconstrained program with respect to QP (5-6), with an objective not less than than (5-6)'s objective.*

The feasible region of the LP relaxation contains the integer solution to $\text{argmax}_{\mathbf{y}} H(\mathbf{y})$. This means the relaxed solution $\mathbf{e}$ forms an upper bound on $\text{argmax}_{\mathbf{y}} H(\mathbf{y})$, i.e., $H(\hat{\mathbf{y}}) \leq H(\mathbf{e})$. If $\hat{\mathbf{y}}$'s corresponding constraint would be introduced, $\mathbf{e}$'s corresponding constraint must also be introduced. So, at the end of the iterations no constraint in the QP (5-6) is significantly violated, and as additional constraints not in the (5-6) may have been introduced, the QP is potentially overconstrained with respect to (5-6). Since the extra constraints may exclude (5-6)'s solution from the feasible region, the objective cannot be less than QP (5-6)'s objective.

For evaluation on the test set, we employ C_R^*, a discretized version of C_R. C_R^* forces a relaxed solution $\mathbf{e}$ into discrete clusters with a simple ball-growing technique: Start with an initial partitioning $\mathbf{y}$ that has every item in $\mathbf{x}$ in its own cluster. Iterate over all $x_a \in \mathbf{x}$. If x_a is currently in a singleton cluster in $\mathbf{y}$, select it. Then, for all other $x_b \in \mathbf{x}$ still in a singleton cluster, put x_b in x_a's cluster if $e_{a,b} > 0.7$.

6. Loss Functions

Many learning tasks already have existing performance measures. For example, performance on noun-phrase coreference is often evaluated with the MITRE score (Vilain et al., 1995). While many learning methods optimize to some implicit performance measure, good performance on this learning measure may not translate into good performance on the desired measure. In this section we test whether SVM^{cluster}'s ability to optimize to a particular loss function is beneficial. We use SVM^{cluster} with two loss functions:

Δ_P (Pairwise Loss) is $\Delta_P(\mathbf{y}, \bar{\mathbf{y}}) = 100\frac{W}{T}$, where T is the total number of pairs of items in the set partitioned by $\mathbf{y}$ and $\bar{\mathbf{y}}$, and W is the total number of pairs where $\mathbf{y}$ and $\bar{\mathbf{y}}$ disagree about their cluster membership. This is the complement of the Rand index (Rand, 1971). When using the relaxation approximation to $\text{argmax}_{\mathbf{y}} H(\mathbf{y})$, we incorporate Δ_P into the linear objective Expression 9 by changing summands to $e_{a,b} \cdot (\mathbf{w}^T \phi_{a,b} + \frac{100}{T})$ for x_a, x_b in different clusters.

Δ_M (MITRE Loss) is $\Delta_M(\mathbf{y}, \bar{\mathbf{y}}) = 100\frac{2(1-R)(1-P)}{(1-R)+(1-P)}$ where R and P are the MITRE recall and precision scores respectively (Vilain et al., 1995). The MITRE measures R and P are too complex to describe briefly, but can be looked at in terms of the number of operations to transform $\mathbf{y}$ into $\bar{\mathbf{y}}$. Suppose we consider two operations: merge two clusters in $\mathbf{y}$ to form one cluster, or split one cluster in $\mathbf{y}$ to form two clusters. The recall and precision are proportional to how many merges and splits are needed to transform $\bar{\mathbf{y}}$ into $\mathbf{y}$.

7. Experiments

This section describes experiments to test the ability of SVM^{cluster} to exploit dependencies in data, to examine the importance of the loss function during optimization, and to examine the different approximations to $\text{argmax}_{\mathbf{y}} H(\mathbf{y})$. We evaluate SVM^{cluster}'s performance on noun-phrase clustering and news article clustering.

For the MUC-6 noun-phrase coreference task, there are 60 documents with their noun-phrases assigned to coreferent clusters. Each document had an average of 101 clusters, with an average of 1.48 noun-phrases per cluster; there are many single element clusters. The first 30 documents form the training set. The last 30 form the evaluation set. The pairwise feature vectors for pairs of noun-phrases are those used in (Ng & Cardie, 2002). Each feature vector contains 53 features, e.g., whether the noun-phrases appear to have the same gender, how many sentences apart they are, whether either one is the subject in a sentence, etc.

The news article clustering data set is a new data set we derived by trawling Google News. Google News *itself* works by clustering news articles, but presumably their clustering method is sufficiently sophisticated that teaching an unsophisticated clustering method how to cluster in the same fashion is interesting. For each day for 30 days, at most 10 topics from the "World" category were selected, and from each topic at most 15 articles were selected. The topics form our true reference clusters. We have various simple heuristics for extracting the article text, quoted article text, headline, and title. The first 15 days are the training set, and the last 15 days are the test set.

Each article has 30 TFIDF weighted vectors for unigrams, bigrams, and trigrams of the text appearing in the title, the headline according to two extraction methods, article text, and article text in quotations, and for all of these there are Porter stemmed and non-stemmed versions of the vectors. The pairwise feature vector $\phi_{a,b}$ for two articles $x_a, x_b \in \mathbf{x}$ are the 30 cosine similarities between these entities corresponding vectors in x_a and x_b, plus one feature which is always the constant 1. For example, feature 11 is the cosine similarity among TFIDF bigrams in unstemmed text.

With these data sets, we trained and tested several supervised clustering models. A model consists of the learned similarity weights $\mathbf{w}$. In all cases, the C regularization parameter was chosen from several values based on k-fold cross validation on the training set ($k = 10$ for NP-coreference, $k = 5$ for news article clustering). Significance tests between the results for two models use the paired two-tailed T-test. Performance is considered significantly different for p values less than 0.05. For our baseline, we use PCC (pairwise classification clustering), the naïve approach described in Section 3. PCC uses SVM^{light} as the pairwise classifier, and clusters with correlation clustering.

Table 1. Results for NP Coreference

	C_G	PCC	Default
Test with C_G, Δ_M	41.3	51.6	51.0
Test with C_G, Δ_P	2.89	3.15	3.59

Table 2. Results for News Articles

	C_G	C_R	PCC	Default
Test with C_G, Δ_P	2.36	2.43	2.45	9.45
Test with C_R^*, Δ_P	2.04	2.08	1.96	9.45

7.1. SVM^{cluster} Versus PCC

Section 3 outlines problems with a method like PCC. We supposed SVM^{cluster} would be able to handle transitive dependencies better than a simple pairwise classifier. How does SVM^{cluster} compare to PCC?

Table 1 shows a comparison on the noun-phrase task. The C_G column contains results of two models trained on SVM^{cluster} using the greedy C_G approximation, with the first optimized and tested with respect to the MITRE loss Δ_M, the second with respect to the pairwise loss Δ_P. Both tests used greedy C_G clustering on the test set with the learned similarity measure. The PCC column contains analogous results for PCC. The default column contains results for a model that either puts each item in its own cluster (for Δ_P), or all in one cluster (for Δ_M).

The SVM^{cluster} model performs significantly better. While the Δ_M performance could be explained as optimization to a loss which PCC cannot do, the Δ_P loss, as the proportion of pairwise relationships that are wrong, is analogous to pairwise accuracy, which is what PCC's classifier optimizes. Even under this configuration, SVM^{cluster} performs significantly better.

What happens for item sets without complex transitive dependencies between items? Consider the case where you view two noun-phrases in isolation, versus two news articles in isolation. While it is often very difficult to tell whether two noun-phrases co-refer by just looking at two noun-phrases taken out of context, it is usually quite easy to tell if two news articles are about the same topic just by viewing the two articles. For this reason, it seems less helpful to exploit dependencies in a task like news article clustering. In Table 2, we see the results of a comparison between SVM^{cluster} and PCC. The C_G and C_R columns refer to the clusterers used in the cost function approximation in SVM^{cluster}. The two rows show the performance of the learned similarity measure with different clustering methods. Though results seem mixed, the results among the different methods in each row are not statistically different from one another. These empirical results suggest that SVM^{cluster} is more effective than the nave PCC approach when the data contains tran-

Table 3. Training and testing on separate losses.

	Opt. to Δ_M	Opt. to Δ_P
Performance on Δ_M	41.3	42.8
Performance on Δ_P	4.06	2.89

sitive dependencies, and that both methods perform comparably when not.

7.2. Optimization to Loss

The SVM^{cluster} algorithm has the ability to optimize to specific loss functions. How important is it to use the correct loss function during training? We address this question in an experiment that evaluates how a model optimized for one loss function performs when evaluated under a different loss function.

Table 3 shows evaluation results on the NP-task for models optimized to different losses (corresponding to columns) and evaluated on different losses (corresponding to rows). The performances in the first row for the MITRE loss Δ_M are not significantly different for models optimized to Δ_M and Δ_P. Interestingly, when optimized under the pairwise loss Δ_P, there is a great difference; indeed, models optimized to Δ_M are not even significantly different from the default clustering shown in Table 1. We conclude that optimization to the appropriate loss function can make a significant and substantial difference in clustering accuracy.

7.3. Loss in $\text{argmax}_{\mathbf{y}} H(\mathbf{y})$

The cost function H includes a loss function, but when computing $\text{argmax}_{\mathbf{y}} H(\mathbf{y})$, sometimes including the loss function is difficult or impossible for computational reasons, e.g., including the MITRE score in the linear objective for correlation clustering. Can we sometimes get away with not including the loss in the $\text{argmax}_{\mathbf{y}} H(\mathbf{y})$? Note, we do still include the loss when introducing a new QP constraints; however, the method to choosing which constraint to introduce would no longer necessarily find the best constraint.

A comparison of SVM^{cluster} models that differed only in whether the loss is not or is included in the cost function is seen in Table 4. No two results in a row of this table differ significantly. This bodes well for situations where including the cost in the $\text{argmax}_{\mathbf{y}} H(\mathbf{y})$ approximation is difficult.

7.4. Greedy vs. Relaxation

For clustering, finding the exact $\text{argmax}_{\mathbf{y}} H(\mathbf{y})$ present in Line 6 of the algorithm requires solving an NP-hard problem, so we instead use greedy and relaxation approximations. How do these approximations compare?

Table 4. Comparison of performance when loss was not used in the $\text{argmax}_{\mathbf{y}} H(\mathbf{y})$, versus when it was included. NP-coreference experiments used C_G clustering. News experiments used Δ_P loss.

	w/ loss	w/o loss
NP-coreference, Δ_M	41.3	41.1
NP-coreference, Δ_P	2.89	2.81
News, train C_G, test C_G	2.36	2.42
News, train C_R, test C_R^*	2.08	2.16

Table 5. Comparison of performance when different clustering methods were used to approximate $\text{argmax}_{\mathbf{y}} H(\mathbf{y})$.

	Train C_G	Train C_R
Test C_G	2.36	2.43
Test C_R^*	2.04	2.08

The different clustering methods C_G and C_R are used in training models for the news article task. In Table 5 we compare models that differ only in which approximation was used during training. The test results are not significantly different. This comparison was run only on the news story task: the off the shelf linear solver used in the correlation clustering implementation could not handle some problem sizes in the noun-phrase MUC-6 task. These results provide no basis to prefer either the greedy underconstrained approximation or relaxation overconstrained approximation.

7.5. Efficiency of SVM^{cluster}

When run on the NP-coreference problem, before SVM^{cluster} converged, about 1000 constraints were introduced into an SVM QP reoptimized 50 times. The overhead of clustering these small sets is small relative to the time spend reoptimizing the QP; using greedy C_G clustering, only one percent of the time spent reoptimizing the QPs was spent clustering. Of all the reported experiments, the longest SVM^{cluster} ever took to converge was between 3 and 4 hours, with under one hour as a more typical time. Due to PCC's simplicity one might suspect superior performance; however, with slightly under half a million noun-phrase pairs in the training set, training PCC's classifier required half a week with half a million constraints.

8. Conclusions

We formulated a supervised clustering method SVM^{cluster} based on an SVM framework for learning structured outputs. The algorithm accepts a series of "training clusters," a series of sets of items and clusterings over that set. The method learns a similarity measure between item pairs to cluster future sets of items in the same fashion as the training clusters.

The learning algorithm's correctness depends on an ability to iteratively find and introduce the most violated constraint. Since finding the most violated constraint is intractable for clustering, we use existing clustering methods to help find an approximation. We experimentally evaluate two approximations: one based on greedy clustering, and one based on a linear programming relaxation. Both produce comparable results. Further, we find that a simplified formulation that excludes the loss from $\text{argmax}_{\mathbf{y}} H(\mathbf{y})$ does not lead to a loss in accuracy. Overall, the results suggest that SVM^{cluster}'s ability to optimize to a custom loss function and exploit transitive dependencies in data does improve performance compared to a naïve classification approach.

Acknowledgments

This work was supported under NSF Award IIS-0412894 and through the KD-D grant.

References

Aggarwal, C. C., Gates, S. C., & Yu, P. S. (1999). On the merits of building categorization systems by supervised clustering. *ACM SIGKDD-1999* (pp. 352–356). San Diego, California, United States: ACM Press.

Bansal, N., Blum, A., & Chawla, S. (2002). Correlation clustering. *Machine Learning*, *56*, 89–113.

Basu, S., Bilenko, M., & Mooney, R. J. (2004). A probabilistic framework for semi-supervised clustering. *ACM SIGKDD-2004* (pp. 59–68). Seattle, WA.

Bilenko, M., Basu, S., & Mooney, R. J. (2004). Integrating constraints and metric learning in semi-supervised clustering. *ICML*. New York, NY, USA: ACM Press.

Cohen, W., & Richman, J. (2001). Learning to match and cluster entity names. *ACM SIGIR workshop on Mathematical/Formal Methods in IR*.

De Bie, T., Momma, M., & Cristianini, N. (2003). Efficiently learning the metric using side-information. *ALT2003* (pp. 175–189). Sapporo, Japan: Springer.

Demaine, E., & Immorlica, N. (2003). Correlation clustering with partial information. *RANDOM-APPROX 2003* (pp. 1–13). Princeton, New Jersey.

Joachims, T. (2003). *Learning to align sequences: A maximum-margin approach* (Technical Report).

Kamishima, T., & Motoyoshi, F. (2003). Learning from cluster examples. *Mach. Learn.*, *53*, 199–233.

Lanckriet, G. R. G., Cristianini, N., Bartlett, P., Ghaoui, L. E., & Jordan, M. I. (2004). Learning the kernel matrix with semidefinite programming. *J. Mach. Learn. Res.*, *5*, 27–72.

McCallum, A., Nigam, K., & Ungar, L. H. (2000). Efficient clustering of high-dimensional data sets with application to reference matching. *Knowledge Discovery and Data Mining* (pp. 169–178).

McCallum, A., & Wellner, B. (2003). Toward conditional models of identity uncertainty with application to proper noun coreference. *IIWeb* (pp. 79–84).

MUC-6 (1995). *Proceedings of the Sixth Message Understanding Conference (MUC-6)*. San Francisco, California: Morgan Kaufmann.

Ng, V., & Cardie, C. (2002). Improving machine learning approaches to coreference resolution. *ACL-02* (pp. 104–111).

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

Soon, W. M., Ng, H. T., & Lim, D. C. Y. (2001). A machine learning approach to coreference resolution of noun phrases. *Computational Linguistics*, *27*, 521–544.

Swamy, C. (2004). Correlation clustering: maximizing agreements via semidefinite programming. *ACM-SIAM SODA* (pp. 526–527). New Orleans, Louisiana: Society for Industrial and Applied Mathematics.

Taskar, B., Chatalbashev, V., & Koller, D. (2004). Learning associative markov networks. *ICML*. Banff, Alberta, Canada: ACM Press.

Taskar, B., Guestrin, C., & Koller, D. (2003). Max-margin markov networks. *NIPS*.

Tsochantaridis, I., Hofmann, T., Joachims, T., & Altun, Y. (2004). Support vector machine learning for interdependent and structured output spaces. *ICML*.

Vilain, M., Burger, J., Aberdeen, J., Connolly, D., & Hirschman, L. (1995). A model-theoretic coreference scoring scheme. *MUC-6* (pp. 45–52). San Francisco, California: Morgan Kaufmann.

Wagstaff, K., Cardie, C., Rogers, S., & Schroedl, S. (2001). Constrained k-means clustering with background knowledge. *ICML*.