Multi-View Hierarchical Semi-supervised Learning by Optimal Assignment of Sets of Labels to Instances

Bhavana Dalvi
School of Computer Science
Carnegie Mellon University
Pittsburgh, PA 15213
bbd@cs.cmu.edu

William W. Cohen
School of Computer Science
Carnegie Mellon University
Pittsburgh, PA 15213
wcohen@cs.cmu.edu

ABSTRACT

In multiclass semi-supervised learning, sometimes the information about datapoints is present in multiple views. In this paper we propose an optimization based method to tackle semi-supervised learning in the presence of multiple views. Our techniques make use of linear programming and mixed integer linear programming formulations along with the EM framework to find consistent class assignments given the scores in each data view. We compare our techniques against existing baselines on a number of multi-view datasets. Our proposed techniques give state-of-the-art performance in terms of F1 score, outperforming a well-studied SSL method based on co-training. Further, we show that our techniques can be easily extended to multi-view learning in the presence of hierarchical class constraints. These extensions improve the macro-averaged F1 score on a hierarchical multi-view dataset.

1. INTRODUCTION

In multiclass semi-supervised learning, sometimes the information about datapoints is present in multiple views. For instance consider the Web document classification: a learning algorithm can use two views, the text within the document and the anchor texts of its inbound hyper-links. Similarly, in an information extraction task to populate a Knowledge Base(KB) like NELL [8], each noun-phrase to be classified has different sets of features or data views associated with it; e.g., text contexts that appeared with it, its occurrences in HTML table-columns, morphological features, and so on. A common approach is to concatenate the feature vectors for each view but this is not always optimal. Multi-view learning [37, 2, 31, 30, 12] addresses this problem by introducing a different function to model each view, and jointly optimizing all the functions to exploit the redundant views and improve learning performance.

Multiple views are only one issue arising in complex real-world learning tasks. For instance, in the above mentioned KB population task, labels assigned to each noun-phrase need to be consistent with the hierarchical class constraints posed by the KB ontology. There has already been a lot of research in the individual areas of multi-view learning[2, 31, 30, 12], semi-supervised learning[32, 8, 14], and learning in the presence of class hierarchies[24, 22, 29, 13]. However, the problem of unifying these aspects into a single framework is relatively less explored.

In this paper we focus on the problem of hierarchical multi-view semi-supervised learning. We propose an optimization based formulation for this problem in the EM framework which extends the popular spherical K-Means algorithm. The main idea behind our approach is to use different binary labels to model membership in each view, and each level of a hierarchy. We then use linear programming and mixed integer programming formulations to optimally assign sets of labels to an instance. Instead of collectively optimizing labels for all datapoints, we solve an integer linear program for every datapoint, deciding the optimal label assignments for the datapoint. We compare our methods against a state-of-the-art co-training approach, applied to semi-supervised spherical K-Means algorithm [1].

Our experiments show that our proposed methods work as well as or better than standard multi-view baselines like multiplication, addition or concatenation of scores produced by the two views, and outperform a well-studied co-training based baseline, when applied in the standard multi-view k-way classification setting. We further show that our methods are easily extensible to hierarchical multi-view classification scenarios i.e., problems where a datapoint can belong to multiple classes at different levels of a hierarchy.

Contributions. Past approaches consider multi-view and hierarchical learning scenarios separately. We propose a unified optimization based framework for multi-class multi-view hierarchical semi-supervised learning. We show that using this framework we can derive multiple existing as well as new flat multi-view learning methods.

With an experimental evaluation on 9 different multi-view datasets, we showed that our techniques give state-of-the-art performance when compared to existing multi-view learning methods including the co-training based algorithm proposed by Bickel and Scheffer [1], on the problem of flat multi-view semi-supervised learning. We also found that using hierarchical variants that use class hierarchy in the learning process improve over their flat counterparts.

This shows the potential of such linear optimization based formulations for complex learning scenarios that cover multi-view and hierarchical classification in a single framework. Finally, we have made the hierarchical multi-view NELL_mv dataset available [1]Instead of collectively optimizing labels for all datapoints, we solve an integer linear program for every datapoint, deciding the optimal label assignments for the datapoint. We have published the

[1]The dataset can be downloaded from `http://rtw.ml.cmu.edu/wk/WebSets/multiView_wsdm_2015_online/index.html`

text context and HTML table features, class hierarchy, hierarchical labels for all entities, and seed train-test partitions of NELL_mv dataset used in this paper.

Outline. In Section 2 we propose our optimization framework for multi-view semi-supervised learning. We then discuss various multi-view methods that can be instantiated using this framework. Section 3 presents extensions of our methods to multi-view problems with hierarchical class constraints. Datasets and experimental evaluation of proposed methods are presented in Section 4. Finally, we discuss related work in Section 5, followed by conclusions.

2. MULTI-VIEW SEMI-SUPERVISED LEARNING

The spherical K-Means algorithm [16], i.e., the K-Means algorithm with cosine similarity, is a popular method for clustering high-dimensional data such as text. In this section we will first review this existing algorithm, we then present our proposed extension called "multi-view spherical K-Means".

2.1 Background and Notations

Let us start with the spherical K-Means optimization problem, which makes use of cosine similarity between datapoint and centroids to do the clustering. In this algorithm [16], each document, as well as each cluster centroid, is represented as a high-dimensional unit-length vector. Let $X = x_1, \ldots x_N$ be the datapoints, $C_1, \ldots C_K$ be the clusters, y_{ij} be an indicator variable specifying whether datapoint x_j belongs to cluster C_i, and μ_i denotes the centroid for cluster C_i. The task is to find optimal values of label assignments y_{ij} and cluster centroids μ_i, so as to optimize the objective function given in Equation 1.

$$\max_{y_{ij},\mu_i} \left(\sum_{j=1}^{N} \sum_{i=1}^{K} y_{ij} * cos(x_j, \mu_i) \right)$$
$$\text{s.t.} \sum_{i=1}^{K} y_{ij} = 1, \forall j = 1 \ldots N, y_{ij} \in \{0, 1\} \quad (1)$$

Since the variables y_{ij} and μ_i are dependent on each other, the above function can be optimized using the iterative EM like algorithm such that the E step finds the best of values of y_{ij} given μ_i's and then M step takes these y_{ij}'s to recompute the optimal μ_i's. Instead of probabilistically assigning a datapoint to a number of clusters, we choose hard assignments [9] i.e., we restrict y_{ij} as binary integers, hence this is the hard-EM algorithm. Fixing the values of μ_i's, the objective function is maximized when $y_{ij} = 1$ for $i = \textit{argmax}_i\ cos(x_j, \mu_i)$ and 0 otherwise. Let us now compute the μ_i that maximizes the objective given the values for y_{ij}'s. ($\langle a, b \rangle$ denotes dot product of vectors a and b.)

$$\max_{\mu_i} \left(\sum_{j=1}^{N} y_{ij} * cos(x_j, \mu_i) \right) = \max_{\mu_i} \left\langle \sum_{j=1}^{N} y_{ij} \frac{x_j}{||x_j||}, \frac{\mu_i}{||\mu_i||} \right\rangle \quad (2)$$

The Cauchy-Schwartz inequality [19] states that $|\langle x, y \rangle| \leq ||x|| * ||y||$, and the two sides are equal only if x and y are linearly dependent. To maximize Equation 2, we can set $\mu_i = \sum_j y_{ij} \frac{x_j}{||x_j||}$. If we normalize x_j and μ_i to be unit vectors, then the equations can be simplified further. Given μ_i's, $y_{ij} = 1$ for $i = \textit{argmax}_i \langle x_j, \mu_i \rangle$ and 0 otherwise. Similarly, given y_{ij}'s, $\mu_i = \sum_j y_{ij} * x_j$. Henceforth, we assume that x_j and μ_i are normalized to be unit vectors. Hence, $cos(x_j^{(2)}, \mu_i^{(2)}) = \langle x_j^{(2)}, \mu_i^{(2)} \rangle$. Finally, the spherical K-Means algorithm, repeats these two steps iteratively till convergence.

2.2 Multi-View K-Means

We extend the spherical K-Means algorithm [16] for multi-view setting and present a general optimization based framework for Multi-View K-Means. Later, we also describe various ways of combining evidence from the different views to instantiate multiple variants of this general algorithm.

In the multi-view scenario, a datapoint x_j has a feature representation in each view denoted by $x_j^{(1)}$ in view 1 and $x_j^{(2)}$ in view 2. Once the datapoints are clustered in k clusters, each centroid can also be represented differently in each view; i.e., centroid μ_i can be represented as $\mu_i^{(1)}$ in view 1 and $\mu_i^{(2)}$ in view 2. There is one score for every datapoint in every view; i.e., we will have scores $s_{ij}^{(1)}$ and $s_{ij}^{(2)}$ for datapoint x_j and centroid μ_i in view 1 and view 2 respectively to be defined below. There is a single label assignment vector per datapoint that combines scores from different views, which we represent as matrix Y, i.e. $y_{ij} = 1$ indicates that datapoint x_j belongs to cluster μ_i.

Let us define an *effective score* s_{ij} of datapoint x_j belonging to cluster μ_i as an overall measure of similarity of x_j to μ_i under both views. Then the optimization problem described in Equation 1 can be re-written as:

$$\max_{y_{ij}} \left(\sum_{j=1}^{N} \sum_{i=1}^{K} y_{ij} * s_{ij} \right)$$
$$\text{s.t.} \sum_{i=1}^{K} y_{ij} = 1, \forall j = 1 \ldots N, y_{ij} \in \{0, 1\} \quad (3)$$

In the multi-view scenario, the score s_{ij} can be defined as a function of $s_{ij}^{(1)}$ and $s_{ij}^{(2)}$. This can be done in various ways: e.g., a linear combination of scores, multiplication of scores, taking max or min over scores and so on.

Algorithm 1 describes a generic semi-supervised multi-view K-Means algorithm. Next, we will see various ways in which the cluster assignment step (line 8) and the centroid computation step (line 10) in this algorithm can be done, leading to new multi-view SSL algorithms. Recall that instead of probabilistically assigning a datapoint to a number of clusters, we choose hard assignments i.e., we restrict $Y_t^u(x)$ (in Line 8) as binary integers, hence this is the hard-EM algorithm (or classification EM) [9]. Even though finding an optimal set of hard assignments using mixed integer programs is relatively more expensive than soft assignments; we are solving it per datapoint, instead of collectively optimizing labels for all datapoints. Hence we can divide the datapoints into multiple shards and parallelize the computation in Line 8 of Algorithm 1. Since we are using the spherical K-Means algorithm in a semi-supervised setting, henceforth we will use "cluster" and class" interchangeably.

2.3 Cluster Assignment using Scores from Two Data Views

In this section, we go through various ways in which the cluster assignment and centroid computations steps (lines 8, 10) in Algorithm 1 can be done. We will start with an assumption that each datapoint belongs to exactly one cluster. (In case of classification, this can be viewed as a class constraint that says "all classes $C_1 \ldots C_k$ are mutually exclusive".) We also assume that the cluster assignment for a datapoint remains same in all data views. Later we will relax these assumptions one by one and present a way to solve these problem variants.

SUMSCORE Method

Here we define the effective score as an addition of scores in different views: $s_{ij} = s_{ij}^{(1)} + s_{ij}^{(2)}$. Let us say $s_{ij}^{(1)} = \langle x_j^{(1)}, \mu_i^{(1)} \rangle$ and

Algorithm 1 Semi-supervised Multi-View K-Means Algorithm

function Multi-view-K-Means ($X^{l(1)}$, $X^{l(2)}$, Y^l, $X^{u(1)}$, $X^{u(2)}$, k): $\mu^{(1)}$, $\mu^{(2)}$, Y^u
Input: $X^{l(1)}$, $X^{l(2)}$ labeled data points in view 1 and view 2 resp. with L_2 norm = 1; Y^l labels(or cluster assignments) for datapoints X^l; $X^{u(1)}$, $X^{u(2)}$ unlabeled datapoints in view 1 and view 2 resp. with L_2 norm = 1 (same feature space as X^l); k number of clusters to which the datapoints belong.
Output: $\mu^{(1)}$, $\mu^{(2)}$ cluster centroids in view 1 and view 2 resp. with L_2 norm = 1, $\mu^{(1)} = \{\mu_1^{(1)} \dots \mu_k^{(1)}\}$; Y^u labels(or cluster assignments) for unlabeled data points X^u
{Initialize model parameters using labeled data}
$t = 0$, $\mu_0^{(1)} = argmax_\mu\ L(X^{l(1)}, Y^l|\mu)$,
$\mu_0^{(2)} = argmax_\mu\ L(X^{l(2)}, Y^l|\mu)$,
{here, $\mu_0^{(1)}$, $\mu_0^{(2)}$ are cluster centroids at iteration 0 in view 1,2 resp.}
while cluster assignments not converged, $t = t + 1$ **do**
{E step: (Iteration t) Make cluster predictions for the unlabeled datapoints}
for $x \in X^u$ **do**
$s_j^{(1)} = \langle x^{(1)}, \mu_{j,t-1}^{(1)} \rangle$,
$s_j^{(2)} = \langle x^{(1)}, \mu_{j,t-1}^{(2)} \rangle$, for all labels $1 \le j \le k$
Compute cluster assignments $Y_t^u(x)$ given scores $s_{1\dots k}^{(1)}$,$s_{1\dots k}^{(2)}$.
end for
{M step : Recompute model parameters using current assignments for X^u}
Compute cluster centroids $\mu_t^{(1)}$, $\mu_t^{(2)}$ given $Y_t^u(x)$.
end while
end function

$s_{ij}^{(2)} = \langle x_j^{(2)}, \mu_i^{(2)} \rangle$.

$$\max_{\substack{y_{ij},\\ \mu_i^{(1)},\mu_i^{(2)}}} \left(\sum_{j=1}^{N} \sum_{i=1}^{K} y_{ij} * \left(\langle x_j^{(1)}, \mu_i^{(1)} \rangle + \langle x_j^{(2)}, \mu_i^{(2)} \rangle \right) \right),$$

$$\text{s.t.} \sum_{i=1}^{K} y_{ij} = 1, \forall j = 1 \dots N,\ y_{ij} \in \{0,1\} \forall i, j$$

In this setting the optimization objective can be written as follows:

$$\max_{\substack{y_{ij},\\ \mu_i^{(1)},\mu_i^{(2)}}} \left(\left\langle \sum_{j=1}^{N} y_{ij} * x_j^{(1)}, \mu_i^{(1)} \right\rangle + \left\langle \sum_{j=1}^{N} y_{ij} * x_j^{(2)}, \mu_i^{(2)} \right\rangle \right) \quad (4)$$

For the E step, keeping μ_i's fixed, the closed form solution to compute the y_{ij}'s is as follows: $y_{ij} = 1$ for $i = argmax_i\ (\langle x_j^{(1)}, \mu_i^{(1)} \rangle + \langle x_j^{(2)}, \mu_i^{(2)} \rangle)$ and 0 otherwise. In the M step, once y_{ij}'s are fixed, the optimal μ_i's are those for which this objective function is maximized i.e. the two terms being summed are individually maximized. Each term is identical to Equation 2, and hence will have same solution. Given y_{ij}, we can compute $\mu_i^{(1)}$ and $\mu_i^{(2)}$ as follows: $\mu_i^{(1)} = \sum_j y_{ij} * x_j^{(1)}$, and $\mu_i^{(2)} = \sum_j y_{ij} * x_j^{(2)}$. Henceforth, we will refer to this method as 'SUMSCORE' method.

PRODSCORE Method

Here we define the effective score as a product of scores in different views: $s_{ij} = s_{ij}^{(1)} * s_{ij}^{(2)}$. Keeping μ_i's fixed, we can compute y_{ij}'s as follows: $y_{ij} = 1$ for $i = argmax_i\ (\langle x_j^{(1)}, \mu_i^{(1)} \rangle * \langle x_j^{(2)}, \mu_i^{(2)} \rangle)$ and 0 otherwise. Given y_{ij}, $\mu_i^{(1)}$ and $\mu_i^{(2)}$ can be computed as follows:

$$\max_{\mu_i^{(1)},\mu_i^{(2)}} \left(\sum_{j=1}^{N} y_{ij} * \left(\left\langle x_j^{(1)}, \mu_i^{(1)} \right\rangle * \left\langle x_j^{(2)}, \mu_i^{(2)} \right\rangle \right) \right) \quad (5)$$

Note that in Equation 5, unlike Equation 4, $\mu_i^{(1)}$ and $\mu_i^{(2)}$ cannot be independently optimized. However we can still follow an iterative procedure, that in the E step assigns $y_{ij} = 1$ for $i = argmax_i\ (\langle x_j^{(1)}, \mu_i^{(1)} \rangle * \langle x_j^{(2)}, \mu_i^{(2)} \rangle)$ and 0 otherwise; and in the M step, recomputes the centroids in different views as $\mu_i^{(1)} = \sum_j y_{ij} * x_j^{(1)}$, and $\mu_i^{(2)} = \sum_j y_{ij} * x_j^{(2)}$. Henceforth, we will refer to this method as the 'PRODSCORE' method.

Next we propose three other variants of the multi-view K-Means algorithm, which differ from the PRODSCORE and SUMSCORE methods in terms of their label assignment strategy in the E step of the EM algorithm. The PRODSCORE and SUMSCORE methods we just saw consider the mutual exclusion constraints as hard constraints. Next, let us see what happens if we convert them into soft constraints. The modified optimization problem is represented in Equation 6 (Figure 1 (a)). $f(y_{ij}, s_{ij}^{(1)}, s_{ij}^{(2)})$ is a function of label vector, and score vectors in the two views. Note that in this case multiple bits in y_i can be 1, and finding the best possible bit vector of y_i's can lead to evaluating 2^k possible assignments.

(a)
$$\underset{y_{ij},\mu_i^{(1)},\mu_i^{(2)}}{\text{maximize}} \left(\sum_{i,j} \left(f(y_{ij}, s_{ij}^{(1)}, s_{ij}^{(2)}) - \left(\text{Penalty for} \left(\sum_{i=1}^{K} y_{ij} \neq 1 \right) \right) \right) \right) \quad (6)$$
subject to, $y_{ij} \in \{0,1\}$

(b)
$$\begin{aligned} \underset{y_i, w_i^{(1)}, w_i^{(2)}, m_1, m_2}{\text{minimize}} \quad & \left(\sum_{\text{i}} \text{w}_\text{i}^{(1)} + \sum_{\text{i}} \text{w}_\text{i}^{(2)} + \text{m}_1 + \text{m}_2 \right) \\ \text{subject to,} \quad & y_i - s_i^{(1)} \le w_i^{(1)},\ \forall i = 1 \dots k \\ & s_i^{(1)} - y_i \le w_i^{(1)},\ \forall i = 1 \dots k \\ & y_i - s_i^{(2)} \le w_i^{(2)},\ \forall i = 1 \dots k \\ & s_i^{(2)} - y_i \le w_i^{(2)},\ \forall i = 1 \dots k \\ & \sum_i y_i \ge 1 - m_1, \\ & \sum_i y_i \le 1 + m_2, \\ & y_i \in \{0,1\},\ \forall i \textbf{.....} y_i \textbf{'s are integers} \\ & m_1 \ge 0, m_2 \ge 0, w_i^{(1)} \ge 0, w_i^{(2)} \ge 0\ \ \forall i \end{aligned} \quad (7)$$

Figure 1: (a) Linear programming optimization for optimal label assignment (b)Mixed integer program for NS method.

Near Score (NS) Method

We now formulate a simple mixed integer programming(MIP) formulation called NS to find best assignment y_i for a datapoint x, given the μ_i's are fixed, i.e. the scores $s_i^{(1)}$ and $s_i^{(2)}$ are known. The E step solves the MIP is given in Equation 7 (Figure 1 (b)), for every datapoint x, hence the index j vanishes, and y_i denotes the assignment of x to cluster μ_i. Here, $w_i^{(1)}$, $w_i^{(2)}$ denotes the slack terms for each class trying to match y_i with $s_i^{(1)}$, $s_i^{(2)}$; and m_1 (m_2) denotes a penalty term for y_i's summing to at least one (at most one).

It is called NS method because the y_i's are assigned so that they are close to scores $s_i^{(1)}$, $s_i^{(2)}$. Equation 7 is a MIP because y_i's are

(a)

$$\underset{y_i^{(1)},y_i^{(2)},d_i,\zeta^1,\zeta^2,\delta^1,\delta^2}{\text{maximize}} \quad \left(\alpha_1 * \left(\sum_{\mathrm{i}} \mathrm{y}_{\mathrm{i}}^{(1)} * \mathrm{s}_{\mathrm{i}}^{(1)} + \mathrm{y}_{\mathrm{i}}^{(2)} * \mathrm{s}_{\mathrm{i}}^{(2)}\right) - \alpha_2 * \left(\sum_{\mathrm{i}} \mathrm{d}_{\mathrm{i}}\right) - \alpha_3 * \left(\zeta^1 + \zeta^2 + \delta^1 + \delta^2\right)\right)$$

$$\begin{aligned}
\text{subject to,} \quad & d_i = |y_i^{(1)} - y_i^{(2)}|, \forall i = 1 \ldots k \\
& \sum_i y_i^{(1)} \leq 1 + \delta^1, \quad \sum_i y_i^{(2)} \leq 1 + \delta^2, \quad \forall i = 1 \ldots k \\
& \sum_i y_i^{(1)} \geq 1 - \zeta^1, \quad \sum_i y_i^{(2)} \geq 1 - \zeta^2, \quad \forall i = 1 \ldots k \\
& \zeta^1, \zeta^2, \delta^1, \delta^2 \geq 0, \quad y_i \in \{0,1\} \quad \forall i
\end{aligned} \tag{8}$$

(b)

$$\underset{y_i^{(1)},y_i^{(2)},y_i^{(3)},d_i^{(12)},d_i^{(23)},d_i^{(13)},\zeta^1,\zeta^2,\zeta^2,\delta^1,\delta^2,\delta^3}{\text{maximize}} \quad \left(\alpha_1 * \left(\sum_{\mathrm{i}} \mathrm{y}_{\mathrm{i}}^{(1)} * \mathrm{s}_{\mathrm{i}}^{(1)} + \mathrm{y}_{\mathrm{i}}^{(2)} * \mathrm{s}_{\mathrm{i}}^{(2)} + \mathrm{y}_{\mathrm{i}}^{(3)} * \mathrm{s}_{\mathrm{i}}^{(3)}\right) - \alpha_2 * \sum_{\mathrm{i}} \left(\mathrm{d}_{\mathrm{i}}^{(12)} + \mathrm{d}_{\mathrm{i}}^{(23)} + \mathrm{d}_{\mathrm{i}}^{(13)}\right) - \alpha_3 * \left(\zeta^1 + \zeta^2 + \delta^1 + \delta^2\right)\right)$$

$$\begin{aligned}
\text{subject to,} \quad & d_i^{(12)} = |y_i^{(1)} - y_i^{(2)}|, \quad d_i^{(23)} = |y_i^{(2)} - y_i^{(3)}|, \quad d_i^{(13)} = |y_i^{(1)} - y_i^{(3)}|, \quad \forall i = 1 \ldots k \\
& \sum_i y_i^{(1)} \leq 1 + \delta^1, \quad \sum_i y_i^{(2)} \leq 1 + \delta^2, \quad \sum_i y_i^{(3)} \leq 1 + \delta^3, \quad \forall i = 1 \ldots k \\
& \sum_i y_i^{(1)} \geq 1 - \zeta^1, \quad \sum_i y_i^{(2)} \geq 1 - \zeta^2, \quad \sum_i y_i^{(3)} \geq 1 - \zeta^3, \quad \forall i = 1 \ldots k \\
& \zeta^1, \zeta^2, \zeta^3, \delta^1, \delta^2, \delta^3 \geq 0, \quad y_i \in \{0,1\} \, \forall i
\end{aligned} \tag{9}$$

Figure 2: Mixed integer program for MAXAGREE method with (a) two views and (b) three views.

binary integers, but the slack and penalty terms can be real numbers.

MAXAGREE Method

The 'NS' method assigns the same label vector for each datapoint in both views. However for some datapoints, view 1 and view 2 might not agree on labels in the initial EM iterations; here we relax this constraint. This new method, called MAXAGREE (maximize agreement) allows different views to assign a datapoint to different clusters, with a penalty on cluster assignments being different. In particular, label predictions are done by solving a mixed integer linear program on scores produced in the two views and choosing a label per datapoint per data view, with a penalty for assigning different labels for a single datapoint in different views. Equation 8 (Figure 2 (a)) shows the mixed integer linear program to be solved for each datapoint.

For each datapoint $s_i^{(1)}$ and $s_i^{(2)}$ are scores of x belonging to cluster i according to data view 1 and view 2 resp. $y_i^{(1)}$ and $y_i^{(2)}$ represent cluster assignments in view 1 and view 2 resp. d_i is the penalty on $y_i^{(1)}$ and $y_i^{(2)}$ being different. ζ^1 and ζ^2 are the penalty terms for the constraint that each datapoint should be assigned to at least one cluster in each view. Similarly, δ^1 and δ^2 are the penalty terms for the constraint that each datapoint should be assigned to at most one cluster in each view. α_1, α_2 and α_3 are constants that define relative importance of terms in the objective function.

COTRAIN Method

We also experiment with a well-studied co-training based method 'COTRAIN' as one of the baselines. This method is a multi-view spherical K-Means algorithm that is proposed by Bickel and Scheffer [1]. In particular, it is a co-training variant of the spherical K-Means algorithm, in which predictions made for view 1 in the E step are used to recompute centroids of view 2 in the M step and visa versa.

2.4 Incorporating more than two views

Note that the optimization based methods discussed in this section are presented for two views. However, all methods instantiated using our optimization framework are easily extensible to additional number of views. Say, a datapoint x_j in the third view is represented as $x_j^{(3)}$; and the cluster centroid μ_i in the third view is represented as $\mu_i^{(3)}$. Let us see example adaptations for two of the proposed methods: SUMSCORE and MAXAGREE.

SUMSCORE method with three views

SUMSCORE method will add scores from all three views. In the E step, keeping μ_i's fixed, the closed form solution to compute the y_{ij}'s is as follows: $y_{ij} = 1$ for $i = argmax_i$ $(\langle x_j^{(1)}, \mu_i^{(1)} \rangle + \langle x_j^{(2)}, \mu_i^{(2)} + \langle x_j^{(3)}, \mu_i^{(3)} \rangle)$ and 0 otherwise. In the M step, once y_{ij}'s are fixed, the optimal μ_i's are computed as follows: $\mu_i^{(1)} = \sum_j y_{ij} * x_j^{(1)}$, $\mu_i^{(2)} = \sum_j y_{ij} * x_j^{(2)}$, and $\mu_i^{(3)} = \sum_j y_{ij} * x_j^{(3)}$.

MAXAGREE method with three views

Equation 9 (Figure 2 (b)) shows the modified optimization problem for the MAXAGREE method. It tries to maximize pairwise agreement between all three views indicated by variables $d_i^{(12)}$, $d_i^{(23)}$ and $d_i^{(13)}$. Two new penalty terms ζ^3 and δ^3 are introduced. ζ^3 constrains each datapoint should be assigned to at least one cluster in each view. Similarly, δ^3 are the penalty terms for the constraint that each datapoint should be assigned to at most one cluster in each view.

Although our methods are easily extensible to additional number of views, it is not the focus of this paper; hence we limit all experiments in this paper to two data views.

3. EXTENSIONS TO HIERARCHICAL MULTI-VIEW LEARNING

The methods we discussed in the previous section assume the flat classification scenario, i.e. all classes are mutually exclusive, hence there is a penalty on assigning multiple labels to a single datapoint. However, in real world datasets it is often the case that classes are arranged in a hierarchy. They have subset and mutual exclusion constraints among them i.e. pairs of classes might be constrained to be disjoint ($C_i \cap C_j = \phi$) or in a subset relation ($C_i \subseteq C_j$). E.g., in an information extraction task to populate a KB, the KB ontology might pose constraints that if an entity is classified as "*Mammal*" then it should also be classified as "*Animal*" (i.e., *Mammal* $\subseteq$ *Animal*), and further the same entity should not be classified as "*Reptile*" (i.e. *Mammal* $\cap$ *Reptile* $= \phi$).

In this section we will discuss natural extensions of our proposed multi-view approaches for hierarchical multi-view learning. The only change that needs to be made is that each datapoint can be assigned multiple labels, so that they satisfy the class constraints. Further, instead of making the constraints hard, we relax them using slack variables. These slack variables add a penalty to the objective function upon violating any of the class constraints. Let *Subset* be the set of all subset or inclusion constraints, and *Mutex* be the set of all mutual exclusion constraints. In other words, *Subset* = $\{\langle i, j\rangle : C_i \subseteq C_j\}$ and *Mutex* = $\{(i, j) : C_i \cap C_j = \phi\}$. Our only assumption is that we know subset and mutual exclusion constraints between classes under consideration. However, we do not assume that the classes are necessarily arranged in a tree structured hierarchy. Next, we will discuss three methods that do multi-view learning in the presence of such class constraints.

Hier-SUMSCORE Method

This method is an extension of the SUMSCORE method discussed in Section 2.3. For each datapoint, this method tries to maximize the sum of scores of selected labels, after penalizing for violation of class constraints. The scores from two views are combined through addition. There is a unique label assignment vector for each datapoint across two views computed by solving the mixed integer program given Equation 10 (Figure 3 (a)).

Hier-PRODSCORE Method

This method is an extension of the PRODSCORE method. The mixed integer program for this method is very similar to that of the Hier-SUMSCORE method except that in the objective function, scores from two views are combined using product instead of addition of scores. There is a unique label assignment vector for each datapoint across two views.

Hier-MAXAGREE Method

This is an immediate extension of the MAXAGREE method described in Section 2.3, to incorporate class hierarchy. Different views can assign a datapoint to different sets of clusters. There is a penalty on cluster assignments being different across views, and any violation of the class constraints. Equation 11 (Figure 3 (b)) shows the mixed integer linear program to be solved for each datapoint. For each datapoint $s_i^{(1)}$, $s_i^{(2)}$ represent scores of x belonging to cluster i; and $y_i^{(1)}$, $y_i^{(2)}$ represent cluster assignment of x in view 1 and view 2 respectively. d_i is the penalty on $y_i^{(1)}$ and $y_i^{(2)}$ being different. $\zeta_{ij}^{(1)}$ and $\zeta_{ij}^{(2)}$ are the penalty terms for *Subset* constraints. Similarly, $\delta_{ij}^{(1)}$ and $\delta_{ij}^{(2)}$ are the penalty terms for *Mutex* constraints. α_1, α_2 and α_3 are constants that define relative importance of terms in the objective function.

Note that the hierarchical methods discussed here can work with any class ontology and are not limited to tree structured ontologies.

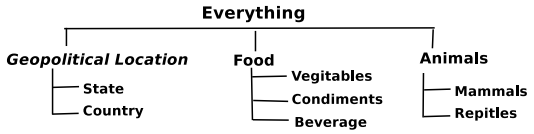

Figure 4: Class ontology used in the NELL_mv dataset.

4. EXPERIMENTAL RESULTS

Here, we first describe the datasets used in this paper. We then explain the experimental setup and evaluation criteria used. Finally, in Section 4.4 and 4.5 we will go through the experimental findings.

4.1 Datasets

We present experiments with 9 multi-view datasets, summarized in Table 1. The first six of them are publicly available, and the last three are created by us for exploring the multi-view learning task. First three datasets Cora, WebKB and CiteSeer$_1$ [21] consists of 2 data views for scientific publications. First view consists of 0/1-valued word vector derived from the document text, and the other view is citation links between these documents. Each document in the Cora, WebKB and CiteSeer$_1$ datasets has been classified into one of the seven, five and six classes respectively. Next two datasets, Wikipedia_linqs and Pubmed_linqs [21] also contain bag of words and the links between documents as the two views, but they are different from the earlier datasets in the sense that words features are tf/idf-weighted instead of being 0/1-valued. Wikipedia_linqs has 19 distinct categories, whereas the Pubmed_linqs dataset has 3 categories.

The UCI Leaves image dataset [28] contains sixteen samples of leaves of each of the one-hundred plant species. For each sample, a shape descriptor, and texture histogram (feature vectors of size 64 each) are given. The Citeseer$_2$ dataset contains text and author views for around 6K scientific articles, classified into 17 categories. The NELL_mv dataset was created using around 1800 entities in the NELL KB and data views being occurrences of those entities in ClueWeb09 text and HTML table data [7, 11]. The NELL_mv dataset contains hierarchical labels that can be arranged in an ontology shown in Figure 4. For flat multi-view learning purposes, we use labels at each level of hierarchy, while for hierarchical multi-view learning experiments we make use of the ontology to create class constraints. The NELL ontology considered here is tree-structured, however our proposed hierarchical methods are not limited to tree-structured ontologies.

General statistics like the number of datapoints, features, classes etc. about these datasets are summarized in Table 1. We can see that our datasets have varying number of datapoints and classes. They also cover different kinds of features like: binary text features, tfidf text features, link features, semi-structured data features and image features. We have made the hierarchical multi-view NELL_mv dataset available [2] for research community to help the future research in this field.

4.2 Experimental Setting

In addition to SUMSCORE and PRODSCORE baselines, we experimented with three other single view baselines. Methods 'V1' and 'V2' are single-view spherical K-Means methods that use only view 1 and view 2 of the data respectively. For the experiments in this paper we order the views for each dataset such that view

[2]The dataset can be downloaded from `http://rtw.ml.cmu.edu/wk/WebSets/multiView_wsdm_2015_online/index.html`

$$\text{(a)} \quad \underset{y_i,\zeta_{ij},\delta_{ij}}{\text{maximize}} \quad \left(\sum_{\text{i}} \text{y}_\text{i} * (\text{s}_\text{i}^{(1)} + \text{s}_\text{i}^{(2)}) - \sum_{\langle \text{i,j} \rangle \in \text{Subset}} \zeta_{\text{ij}} - \sum_{(\text{i,j}) \in \text{Mutex}} \delta_{\text{ij}} \right)$$
$$\text{subject to,} \quad y_j \geq y_i - \zeta_{ij}, \quad \forall \langle i,j \rangle \in \textit{Subset}$$
$$y_i + y_j \leq 1 + \delta_{ij}, \quad \forall (i,j) \in \textit{Mutex}$$
$$\zeta_{ij}, \delta_{ij} \geq 0, \quad y_i \in \{0,1\}\, \forall i,j \tag{10}$$

$$\text{(b)} \quad \underset{y_i^{(1)},y_i^{(2)},d_i,\zeta_{ij}^{(1)},\delta_{ij}^{(1)},\zeta_{ij}^{(2)},\delta_{ij}^{(2)}}{\text{maximize}} \quad \alpha_1 * \left(\sum_{\text{i}} \text{y}_\text{i}^{(1)} * \text{s}_\text{i}^{(1)} + \text{y}_\text{i}^{(2)} * \text{s}_\text{i}^{(2)} \right) - \alpha_2 * \sum_{\text{i}} \text{d}_\text{i}$$
$$-\alpha_3 * \left(\sum_{\langle i,j \rangle \in \textit{Subset}} (\zeta_{ij}^{(1)} + \zeta_{ij}^{(2)}) + \sum_{(i,j) \in \textit{Mutex}} (\delta_{ij}^{(1)} + \delta_{ij}^{(2)}) \right)$$
$$\text{subject to,} \quad d_i = |y_i^1 - y_i^2|, \quad \forall i = 1 \ldots k$$
$$y_j^{(1)} \geq y_i^{(1)} - \zeta_{ij}^{(1)}, \quad y_j^{(2)} \geq y_i^{(2)} - \zeta_{ij}^{(2)}, \quad \forall \langle i,j \rangle \in \textit{Subset}$$
$$y_i^{(1)} + y_j^{(1)} \leq 1 + \delta_{ij}^{(1)}, \quad y_i^{(2)} + y_j^{(2)} \leq 1 + \delta_{ij}^{(2)}, \quad \forall (i,j) \in \textit{Mutex}$$
$$\zeta_{ij}^{(1)}, \delta_{ij}^{(1)}, \zeta_{ij}^{(2)}, \delta_{ij}^{(2)} \geq 0, y_i \in \{0,1\}\, \forall i,j \tag{11}$$

Figure 3: Mixed integer program for (a) Hier-SUMSCORE method and (b) Hier-MAXAGREE method.

Dataset	N	K	View statistics			
			View 1		View 2	
			F	|Data|	F	|Data|
Cora	2708	7	1433	49.2K	2222	5,429
WebKB	877	5	1703	79.4K	735	1608
Citeseer$_1$	3312	6	3703	105.1K	2312	4722
Wikipedia_linqs	3363	19	4973	2.1M	2851	45.0K
Pubmed_linqs	19.7K	3	500	988.0K	19.7K	44.3K
UCI Leaves	1600	100	64	102.0K	64	102.4K
Citeseer$_2$	6601	17	26.7K	392.0K	10.8K	16.0K
NELL_mv	1855	7	3.4M	8.8M	1.1M	2.4M

Table 1: Statistics of all datasets used. Here, N: number of datapoints/documents in the dataset, K: number of classes, F: number of unique features in a particular view, and $|Data|$: total number of (datapoint, feature) occurrences in the view. (Please refer to Section 4.1.)

1 is on average better than view 2 in terms of F1 scores. Using the concatenation of two views as feature vectors yields Method 'V12'. Method 'COTRAIN' is the co-training based multi-view spherical K-Means algorithm proposed by Bickel and Scheffer [1]. We compared these baseline methods with multi-view based on our proposed optimization formulation: These methods include 'SUMSCORE', 'PRODSCORE', 'NS' and 'MAXAGREE' (Section 2.3).

We also experimented with a relaxation of 'NS' method, called Relaxed Near Score (RNS) that uses a linear programming solution by relaxing the integer constraint in Equation 7 i.e., letting y_i's be real numbers. Method 'RNS' solves a linear program on scores produced in the two views. We then choose the label that has been assigned the maximum score. The same set of labels are used to recompute centroids in both views. Although RNS and NS methods look very similar, they result in very a different performance. RNS assigns effective label scores near the average of scores in the two views (very similar to SUMSCORE method) and ranks the y_i's with this score. Our EM algorithm then chooses the bit with maximum score as class label. NS on the other hand sets all those bits 1, which improves the objective function (even after paying a penalty for setting multiple bits to one); hence it behaves like a thresholding method. In the output of NS, all the bits which are set to 1 are indistinguishable, and the EM algorithm picks one of them randomly as a class label. Thus the ranking output by the RNS is much more informative compared to the bits set by NS. In the experimental results presented in this section, we will see that NS often performs worse than RNS.

Further, we evaluate extensions of some of these flat multi-view methods to the hierarchical learning scenarios: 'Hier-SUMSCORE', 'Hier-PRODSCORE', and 'Hier-MAXAGREE' (Section 3). For experiments in this paper we set $\alpha_1 = 0.5$, $\alpha_2 = 0.1$, and $\alpha_3 = 1$ for both the 'MAXAGREE' and 'Hier-MAXAGREE' methods. Note that NS and RNS methods are not extended to hierarchical multi-view setting. NS method did not perform very well compared to MAXAGREE method in the flat multi-view experiments, hence we do not explore it further. Though RNS method performs very well on the flat multi-view task, it was not trivial to extend it to hierarchical setting. This is because it ranks all labels with real valued scores, and we choose a label that gets the maximum score for the flat multi-view task. However, for hierarchical learning such post-processing of label scores to choose the best subset of labels is not straightforward.

4.3 Evaluation Criteria

To evaluate the performance of various methods, we use macro and micro averaged F1 scores. The '*Macro averaged F1 score*' gives equal weight to performance on all classes, hence is better in cases where class distribution is skewed. On the other hand, the '*Micro averaged F1 score*' gives equal weight to performance on all datapoints, hence it is biased towards the majority class. For each dataset we experimented with different values of the training percentage. For each value of training percentage, we average the scores across 10 random train/test partitions. Note that we are running these experiments in the transductive setting i.e. our semi-supervised learning algorithm will get as input training examples with ideal labels as well as unlabeled test examples; and the task is to learn a model from labeled training and unlabeled test data to in turn label the test datapoints.

For the flat multi-view experiments presented in Section 4.4, we run experiments on all 8 datasets, with 2 values of training percentage {10, 30}, with 10 random train-test partitions for each combination. For the NELL_mv dataset we do separate evaluation at second and third level of the class hierarchy. Hence for each method we get 18 values of average F1 scores for each of the 'Macro' and

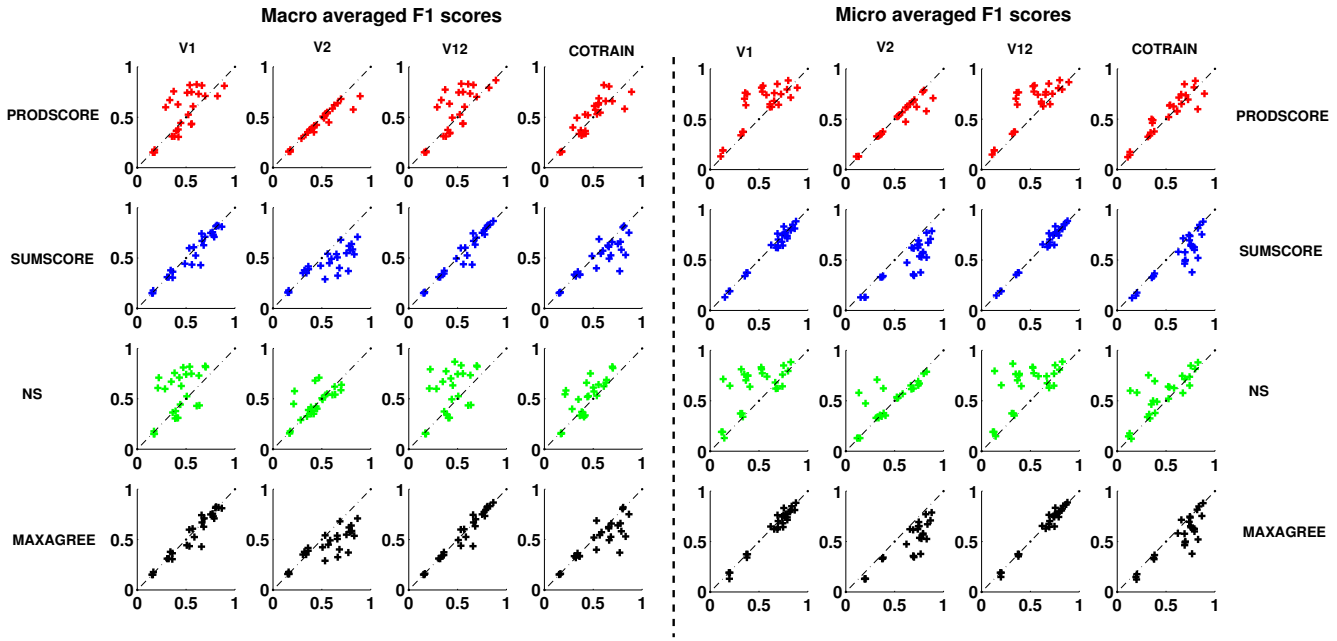

Figure 5: Scatter plots of F1 scores of Flat Multi-view methods on all datasets. Proposed methods (PRODSCORE, SUMSCORE, NS, MAXAGREE) are on the x axis compared to baselines (V1, V2, V12, COTRAIN) on the y axis, hence points below the 'x=y' dotted line mean that the proposed method performed better. (Please refer to Section 4.4.)

'Micro' averaging methods.

Next, we compute the *Average rank* of methods for these 26 test-cases. The lower the average rank, the better the method's performance. Further we visualize these results using *scatter plots* that show the F1 score of each of the proposed method vs. baseline method. Proposed methods are shown on the x axis compared to baselines on the y axis, hence points below the 'x=y' dotted line mean that proposed methods performed better. Both *average rank* and *scatter plots* help us measure the effectiveness of these methods over a number of datasets.

4.4 Results: Flat Multi-view Learning

Table 2 compares the macro-averaged F1 scores of proposed optimization based methods with baseline methods on all 8 multi-view datasets with 10% and 30% training data. V12 method came out as a strong baseline. We can see that our proposed methods SUMSCORE, MAXAGREE and RNS give comparable or better performance w.r.t all baseline methods, and clearly outperform the COTRAIN and V1(best of two views) methods on most datasets. As we have already discussed in Section 4.2, the ranking of labels per datapoint outputted by RNS is similar to that of SUMSCORE method and is much more informative compared to the bits set by NS. This explains the observed performance trends. We can also see that MAXAGREE method performs as well as and sometimes better than the SUMSCORE method. MAXAGREE allows different label vectors to be assigned in different views and hence is the most expressive out of the proposed method.

Figure 5 shows the scatter plot of the Macro and Micro averaged F1 scores of our proposed methods (PRODSCORE, SUMSCORE, NS, and MAXAGREE) vs. baselines (V1, V2, V12, and COTRAIN). Points being below x=y line denotes that our proposed method (plotted on x axis) performed better that the baseline method (plotted on y axis). These plots reassure the fact that SUMSCORE and MAXAGREE methods are performing comparable to or better than all baseline methods. MAXAGREE method outperforms COTRAIN in 15 out of 18 cases, and outperforms V12 in 7 out of 18 cases. SUMSCORE method outperforms COTRAIN in 15 out of 18 cases, and outperforms V12 in 6 out of 18 cases. MAXAGREE method outperforms SUMSCORE method for one datapoint. PRODSCORE and NS methods are not always outperforming all baselines.

Thus we observed that our optimization based label assignment strategies give state-of-the-art or comparable performance when compared to existing flat multi-view techniques, including the well studied co-training based baseline COTRAIN. Results also showed that MAXAGREE, RNS and SUMSCORE methods performed very well on all multi-view datasets. In the next section we will see extensions of our flat multi-view approaches for hierarchical multi-view learning problems.

4.5 Results: Hierarchical Multi-view Learning

We ran experiments on the NELL_mv dataset with training percentage varying from 5% to 30%. We generated 10 random train/test partitions for each training percentage value. Both flat and hierarchical methods got exactly the same train and test instances as input. Hierarchical methods had access to the entire hierarchy of labels and classified each datapoint w.r.t all classes in the hierarchy following the class constraints. Flat methods on the other hand learnt models only for leaf classes of the hierarchy. We compared average F1 scores of all methods on the leaf classes.

Table 3 shows the macro-averaged F1 results for all methods varying the training percentage. We can see that all the proposed hierarchical methods Hier-SUMSCORE, Hier-PRODSCORE and Hier-MAXAGREE improve over their flat counterparts in terms of macro-averaged F1 scores for all values of the training percentage. Column marked as Δ shows the percentage relative improvement of the hierarchical method over its flat counterpart. We can see that

Dataset	Training percentage	Baseline methods				Proposed optimization-based methods				
		V1	V2	V12	COTRAIN	PRODSCORE	SUMSCORE	NS	MAXAGREE	RNS
Cora	10	62.7	40.2	63.4	52.7	41.8	**68.0**+	45.2	**68.0**+	***68.0***+
	30	73.1	62.0	73.5	66.1	62.9	**77.9**+	63.2	77.9+	***78.0***+
WebKB	10	59.9	29.0	**60.0**	39.9	29.2	53.3	28.5	53.3	54.2
	30	74.3	50.6	**74.6**	61.2	51.1	65.9	50.2	65.9	66.7
Citeseer_1	10	**67.1**	32.6	**67.1**	49.4	33.4	66.3	34.9	66.3	66.7
	30	75.0	54.9	75.1	64.1	55.6	**75.2**+	55.5	**75.2**+	***75.6***+
Wikipedia_linqs	10	**60.8**	45.0	60.3	54.5	57.2	57.8	21.2	56.7	55.7
	30	**71.1**	68.2	70.2	66.0	69.6	69.3	43.2	68.6	66.0
Pubmed_linqs	10	75.7	37.1	**77.3**	37.0	37.3	77.2	38.3	77.2	***77.3***+
	30	81.9	53.7	**83.2**	53.1	53.8	**83.2**+	53.7	**83.2**+	***83.2***+
UCI Leaves	10	71.0	57.7	79.3	58.3	**82.2**+	78.8	22.1	78.8	***85.3***+
	30	81.2	71.0	86.7	75.1	**89.2**+	86.6	47.2	86.6	***91.5***+
Citeseer_2	10	15.2	17.7	15.2	15.7	**17.8**+	16.1	17.7+	16.1	16.2
	30	31.0	37.8	31.1	32.1	38.1+	32.3	**40.6**+	32.3	32.4
NELL_mv level=2	10	82.5	58.6	**82.9**	82.0	60.9	81.5	69.6	80.6	80.7
	30	81.5	63.9	**82.0**	80.6	66.2	80.5	70.1	80.1	80.2
NELL_mv level=3	10	44.3	42.4	49.6	**52.1**	44.6	48.8	44.1	50.8	***52.7***+
	30	52.6	49.2	52.6	57.3	52.3	**59.9**+	50.7	58.6+	***60.6***+

Table 2: Comparison of proposed optimization based multi-view methods w.r.t baselines on all 8 datasets in terms of macro-averaged F1 scores. RNS method is presented in the last column. Best F1 scores(excluding RNS method) in each row are bold-faced, and wherever RNS performed better the scores are both bold-faced and italicized. (+) in front of scores for proposed methods indicate that for that dataset the proposed method performed better than or equal to the best baseline score for the dataset. (Please refer to Section 4.4.)

Training Percentage	Baseline methods		Flat vs. hierarchical multi-view methods								
	V12	COTRAIN	SUMSCORE	Hier-SUMSCORE		PRODSCORE	Hier-PRODSCORE		MAXAGREE	Hier-MAXAGREE	
				F1	Δ		F1	Δ		F1	Δ
5	42.26	45.77	43.33	47.20	+8.9%	43.91	44.44	+1.2%	42.11	**47.30**	+12.3%
10	45.40	49.36	46.81	51.91	+10.9%	50.79	51.20	+0.8%	45.65	**51.94**	+13.8%
15	47.99	52.28	49.18	55.32	+12.5%	54.27	55.01	+1.4%	48.27	**55.40**	+14.8%
20	50.45	54.12	52.00	59.23	+13.9%	57.31	59.21	+3.3%	51.19	**59.28**	+15.8%
25	51.35	55.52	52.70	61.07	+15.9%	59.97	**62.71**	+4.6%	51.98	61.13	+17.6%
30	52.43	56.21	53.97	63.86	+18.3%	61.87	**66.00**	+6.7%	53.23	63.95	+20.1%

Table 3: Comparison of hierarchical vs. flat multi-view methods in terms of % Macro averaged F1 on the NELL_mv dataset. Column Δ lists the percentage relative improvement of the hierarchical methods over their flat counterparts. (Please refer to Section 4.5.)

all methods benefit from hierarchy and the the maximum percentage improvements are 18.3% for SUMSCORE, 6.7% for PRODSCORE and 20.1% for the MAXAGREE method. Overall, Hier-MAXAGREE method performs best when the amount of training data is very small, followed by Hier-SUMSCORE and Hier-PRODSCORE methods. Further Hier-PRODSCORE works best for higher values of training percentages.

Figure 6 compares the learning curves of hierarchical and flat versions of MAXAGREE and PRODSCORE methods vs. baselines. We can see all methods are improving with more training data, however learning curves of Hier-MAXAGREE and Hier-PRODSCORE are better than baselines V12 and COTRAIN. This validates the effectiveness of our proposed methods.

Figure 7(a) shows the scatter plot of hierarchical vs. flat methods (SUMSCORE, PRODSCORE and MAXAGREE) in terms of both macro and micro averaged F1 scores. Each method has 6 datapoints corresponding to the 6 different training percentages as shown in Table 3. We can see that hierarchical methods always outperform flat methods in terms of macro-averaged F1, but they might be worse in terms of micro-averaged F1 scores. Figure 7(b) shows that the histogram of NELL_mv leaf class frequencies is skewed. It has been observed that skewed category distribution often leads to less reliable micro-averaged performance [10] (since it is biased towards the most popular class(es)). This can justify the surprising trend in Figure 7(a) that for 6 out of 18 datapoints, flat method outperforms hierarchical method in terms of micro-averaged F1 score. We found that all of these 6 datapoints come from PRODSCORE method. Hence Hier-MAXAGREE and Hier-SUMSCORE outperform their flat counterparts in terms of both macro and micro averaged F1 scores.

Finally we compare the flat and hierarchical multi-view methods in terms of average runtime of experiments presented in this section. Figure 7(c) shows the bar-chart of average run-times of methods. In our MATLAB implementation, the running time of proposed multi-view methods RNS, Hier-PRODSCORE, Hier- SUMSCORE and Hier-MAXAGREE are longer than traditional PRODSCORE, SUMSCORE methods, but not unreasonably so. However, flat multi-view methods NS and MAXAGREE are relatively more expensive, due to the combinatorial number of possible label assignments they need to evaluate while solving mixed integer linear programs. We believe that adding the hierarchical constraints and ignoring unnecessary variables(an implementation trick), reduces the number of possible candidate assignments to evaluate for Hier-MAXAGREE, making it more efficient than MAXAGREE.

Results in this section, proved the superiority of hierarchical multi-view techniques based on our proposed optimization frame-

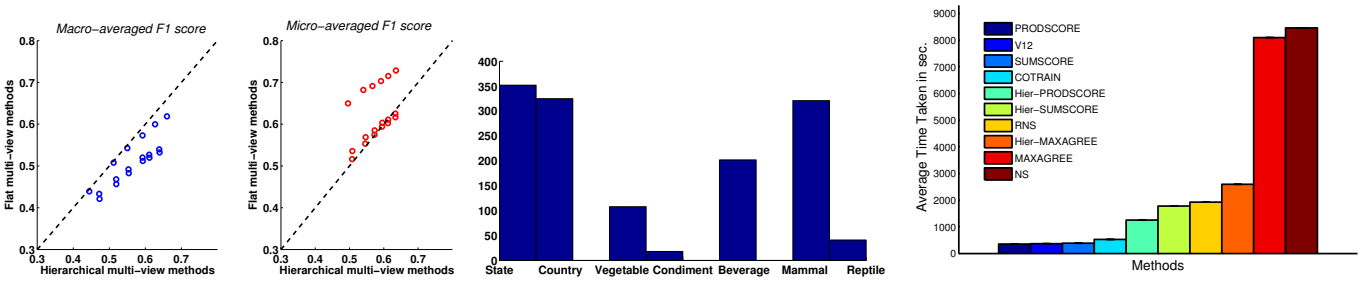

Figure 7: Results on NELL_mv dataset: (a) Scatter plot of hierarchical vs. corresponding flat multi-view methods (b) Class frequency histogram of leaf classes, (c) Average run-times of methods. (Please refer to Section 4.5.)

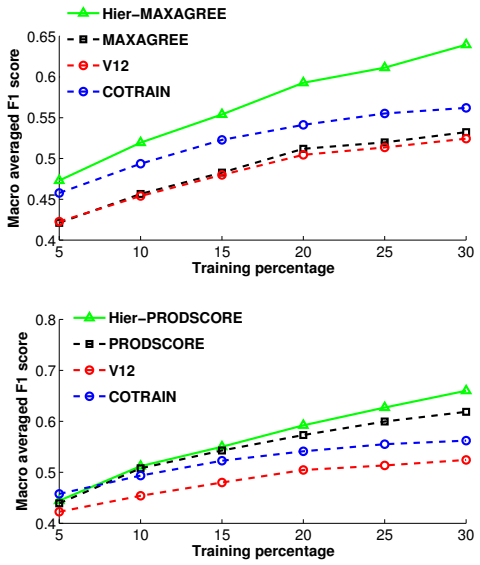

Figure 6: Effect of varying the training percentage for various flat and hierarchical multi-view methods on NELL_mv dataset. (Please refer to Section 4.5.)

work. It was also observed that with small amount of training data, Hier-MAXAGREE method gave state-of-the-art performance on NELL_mv dataset on the task of hierarchical multi-view learning in terms of both macro and micro averaged F1 scores.

5. RELATED WORK

Multi-view learning as defined by Xu et al. [37] is a paradigm that introduces a different function to model each view and jointly optimizes all the functions to exploit the redundant views of the same input data and improves the learning performance. Blum and Mitchell [2] proposed the co-training algorithm for problems where the examples are described by two conditionally independent views. It jointly trains two classifiers such that classifier A adds examples to the labeled set that classifier B will then be able to use for learning. If the two views are conditionally independent, then co-training will always improve the results, otherwise it may not be successful. Later, Nigam and Ghani [31] analyzed the performance of co-training when certain assumptions are violated. More generally, one can define a learning paradigm that utilizes the agreement among different learners, and the particular assumptions of co-training are not required. Instead, multiple hypotheses with different inductive biases, e.g., decision trees, SVMs, etc. can be trained from the same labeled data set, and are required to make similar predictions on any given unlabeled instance. Sindhwani et al. [34] and Brefeld et al. [4] proposed multi-view techniques for the semi-supervised regression problem.

Another related area of research is multi-task learning. Jin et al. [23] proposed a single framework to incorporate multiple views and multiple tasks by learning shared predictive structures. On similar lines, Hu et al. [17] handle the problem of heterogeneous feature spaces in the context of transfer learning, and show improved performance on the tag recommendation task. Kang and Choi [38] developed a method based on restricted deep belief networks for multi-view problems, such that layer of hidden nodes in the belief network have view-specific shared hidden nodes. Usunier et al. [30] focus on learning to rank multilingual documents, using machine translation of documents in other languages as different data views. Several boosting approaches like Mumbo [25] and ShareBoost [33] are also proposed for multi-view learning problems. Our techniques are weakly supervised and do not assume the amount of training data required to train such boosting algorithms.

Recent research on multiple kernel learning has proposed a number of approaches for combining kernels in the regularized risk minimization framework [35, 27, 18]. Researchers have also explored dimensionality reduction techniques to create unified low-dimensional embeddings from multi-view datasets so as to benefit semi-supervised learning and information extraction tasks [15, 12]. Cai et al. [6] proposed the use of structured sparsity inducing norms to make K-Means algorithm run on large datasets using multi-threaded machines. Their method is complementary to our techniques because we focus on consistent label assignment for a single datapoint in the E step of every K-Means iteration, hence can be incorporated in their multi-threaded setting.

Brefeld et al. [3, 5] and Ganchev et al. [20] proposed multi-view learning techniques for more challenging structured output spaces. Our methods are different in the sense that, we make use of the well-studied EM framework, pose the label assignment in the E step as an optimization problem, and propose multiple linear and mixed-integer formulations to solve such optimization problem. Integer linear programming is used by several existing approaches to do constrained inference. Nakashole et. al [29] used integer linear programming while doing fine-grained semantic typing in an information extraction task. Gilpin et al. [22] have proposed a integer linear programming based method for hierarchical clustering. Our techniques are different in the sense that Gilpin et al. focus on unsupervised agglomerative clustering whereas we focus on semi-supervised and multi-view clustering in the presence of predefined

class hierarchy.

Graph based multi-view semisupervised learning techniques have also been proposed in past few years. Wang et al. [36] proposed a multi-graph based semisupervised learning technique that can incorporate multiple modalities, and multiple distance functions in the task of video annotation. Lee et al. [26] proposed a new graph-based multi-label propagation technique and applied it to large datasets by utilizing a map-reduce framework. Though these methods can handle multi-view data, they fail to address the scenarios where classes/labels are arranged in a hierarchy and inference needs to be done following certain constraints between these classes.

6. CONCLUSIONS

In this paper, we investigated the problem of semi-supervised learning in the presence of multiple-data views. We formulated the problem as an optimization problem, and solved it using the standard EM framework. We then focused on the sub-problem of assigning labels to each datapoint (part of E step), and studied various methods for such prediction. Our proposed method solves a linear program or mixed integer linear program to find consistent class assignments given the scores in each data view. Because our multi-view techniques are broadly similar to co-training based algorithms, we also compared them to a seeded version of multi-view spherical K-Means algorithm that is proposed by Bickel and Scheffer [1]. Our methods produced better performance in terms of macro-averaged F1 score compared to this co-training baseline.

However, all the multi-view baselines discussed here are limited to problems where each data point belongs to only one class. We showed that our proposed linear programming based formulation can be easily extended to multi-label classification and can incorporate hierarchical class constraints. For a dataset with hierarchy of classes, our extended optimization methods produced better results when compared to flat multi-view clustering baselines in terms of macro-averaged F1 score. An interesting direction for future research can be to apply these linear programming based techniques for multi-view Exploratory Learning[14] that does semi-supervised learning in the presence of unanticipated classes. Such techniques can further be used for populating knowledge bases along with discovering new classes from multi-view unlabeled data.

7. REFERENCES

[1] S. Bickel and T. Scheffer. Multi-View Clustering. In *ICDM*, 2004.

[2] A. Blum and T. Mitchell. Combining labeled and unlabeled data with co-training. In *COLT*, 1998.

[3] U. Brefeld, C. Buscher, and T. Scheffer. Multi-view discriminative sequential learning. In *ECML*. 2005.

[4] U. Brefeld, T. Gartner, T. Scheffer, and S. Wrobel. Efficient co-regularised least squares regression. In *ICML*, 2006.

[5] U. Brefeld and T. Scheffer. Semi-supervised learning for structured output variables. In *ICML*, 2006.

[6] X. Cai, F. Nie, and H. Huang. Multi-view k-means clustering on big data. In *IJCAI*. 2013.

[7] J. Callan. The clueweb09 dataset. `http://boston.lti.cs.cmu.edu/Data/clueweb09/`.

[8] A. Carlson, J. Betteridge, R. C. Wang, E. R. Hruschka, Jr., and T. M. Mitchell. Coupled semi-supervised learning for information extraction. In *WSDM*, 2010.

[9] G. Celeux and G. Govaert. A classification em algorithm for clustering and two stochastic versions. *Computational statistics & Data analysis*, 1992.

[10] F. C. Chik, R. W. Luk, and K. F. Chung. Text categorization based on subtopic clusters. In *Natural Language Processing and Information Systems*. 2005.

[11] B. Dalvi, W. Cohen, and J. Callan. Websets: Extracting sets of entities from the web using unsupervised information extraction. 2012.

[12] B. Dalvi and W. W. Cohen. Very fast similarity queries on semi-structured data from the web. In *SDM*, 2013.

[13] B. Dalvi, W. W. Cohen, and J. Callan. Classifying entities into an incomplete ontology. In *AKBC*, 2013.

[14] B. Dalvi, W. W. Cohen, and J. Callan. Exploratory learning. In *ECML*, 2013.

[15] B. Dalvi, W. W. Cohen, and C. Jamie. Collectively representing semi-structured data from the web. In *AKBC*, 2012.

[16] I. S. Dhillon and D. S. Modha. Concept decompositions for large sparse text data using clustering. *Mach. Learn.*, 2001.

[17] N. N. L.-Q. Y. Y. Y. Fangwei Hu, Tianqi Chen. Discriminative factor alignment across heterogeneous feature space. In *ECML/PKDD*, 2012.

[18] J. Farquhar, D. Hardoon, H. Meng, J. S. Shawe-taylor, and S. Szedmak. Two view learning: Svm-2k, theory and practice. In *NIPS*, 2005.

[19] M. Fujii, S. Izumino, R. Nakamoto, and Y. Seo. Operator inequalities related to cauchy-schwarz and holder-mccarthy inequalities. In *Nihonkai Mathematical Journal*, 1997.

[20] K. Ganchev, J. Graca, J. Blitzer, and B. Taskar. Multi-view learning over structured and non-identical outputs. *UAI*, 2008.

[21] L. Getoor. Linqs datasets `http://linqs.cs.umd.edu/projects//projects/lbc/index.html`.

[22] S. Gilpin, S. Nijssen, and I. Davidson. Formalizing hierarchical clustering as integer linear programming. In *AAAI*, 2013.

[23] X. Jin, F. Zhuang, S. Wang, Q. He, and Z. Shi. Shared structure learning for multiple tasks with multiple views. In *ECML/PKDD*, 2013.

[24] S. Johnson. Hierarchical clustering schemes. *Psychometrika*, 1967.

[25] S. Koço and C. Capponi. A boosting approach to multiview classification with cooperation. In *ECML/PKDD*, 2011.

[26] W.-Y. Lee, L.-C. Hsieh, G.-L. Wu, and W. Hsu. Graph-based semi-supervised learning with multi-modality propagation for large-scale image datasets. *Journal of Visual Communication and Image Representation*, 2013.

[27] B. Leskes. The value of agreement, a new boosting algorithm. In *Learning Theory*. 2005.

[28] C. Mallah, J. Cope, and J. Orwell. Plant leaf classification using probabilistic integration of shape, texture and margin features. In *Signal Processing, Pattern Recognition and Applications*. 2013.

[29] N. Nakashole, T. Tylenda, and G. Weikum. Fine-grained semantic typing of emerging entities. *ACL*, 2013.

[30] C. G. Nicolas Usunier, Massih-Reza Amini. Multiview semi-supervised learning for ranking multilingual documents. In *ECML/PKDD*, 2011.

[31] K. Nigam and R. Ghani. Analyzing the effectiveness and applicability of co-training. In *CIKM*, 2000.

[32] K. Nigam, A. McCallum, S. Thrun, and T. Mitchell. Text classification from labeled and unlabeled documents using em. *Machine learning*, 2000.

[33] J. Peng, C. Barbu, G. Seetharaman, W. Fan, X. Wu, and K. Palaniappan. Shareboost: Boosting for multi-view learning with performance guarantees. In *ECML/PKDD*, 2011.

[34] V. Sindhwani, P. Niyogi, and M. Belkin. A co-regularization approach to semi-supervised learning with multiple views. In *ICML Workshop on Learning with Multiple Views*, 2005.

[35] J. S.-T. Tom Diethe, David R. Hardoon. Constructing nonlinear discriminants from multiple data views. In *ECML/PKDD*, 2011.

[36] M. Wang, X.-S. Hua, R. Hong, J. Tang, G.-J. Qi, and Y. Song. Unified video annotation via multigraph learning. *Circuits and Systems for Video Technology, IEEE Transactions on*, 2009.

[37] C. Xu, D. Tao, and C. Xu. A survey on multi-view learning. *CoRR*, 2013.

[38] S. C. Yoonseop Kang. Restricted deep belief networks for multi-view learning. In *ECML/PKDD*, 2011.