New Regularized Algorithms for Transductive Learning

Partha Pratim Talukdar and Koby Crammer

Computer & Information Science Department
University of Pennsylvania
Philadelphia, PA 19104
{partha,crammer}@cis.upenn.edu

Abstract. We propose a new graph-based label propagation algorithm for transductive learning. Each example is associated with a vertex in an undirected graph and a weighted edge between two vertices represents similarity between the two corresponding example. We build on Adsorption, a recently proposed algorithm and analyze its properties. We then state our learning algorithm as a convex optimization problem over multi-label assignments and derive an efficient algorithm to solve this problem. We state the conditions under which our algorithm is guaranteed to converge. We provide experimental evidence on various real-world datasets demonstrating the effectiveness of our algorithm over other algorithms for such problems. We also show that our algorithm can be extended to incorporate additional prior information, and demonstrate it with classifying data where the labels are not mutually exclusive.

Key words: label propagation, transductive learning, graph based semi-supervised learning.

1 Introduction

Supervised machine learning methods have achieved considerable success in a wide variety of domains ranging from Natural Language Processing, Speech Recognition to Bioinformatics. Unfortunately, preparing labeled data for such methods is often expensive and time consuming, while unlabeled data are widely available in many cases. This was the major motivation that led to the development of semi-supervised algorithms which learn from limited amounts of labeled data and vast amounts of freely available unannotated data.

Recently, graph based semi-supervised algorithms have achieved considerable attention [2, 7, 11, 14, 17]. Such methods represent instances as vertices in a graph with edges between vertices encoding similarities between them. Graph-based semi-supervised algorithms often propagate the label information from the few labeled vertices to the entire graph. Most of the algorithms tradeoff between *accuracy* (initially labeled nodes should retain those labels, relaxations allowed by some methods) with *smoothness* (adjacent vertices in the graph should be assigned similar labels). Most algorithms only output label information to the unlabeled data in a transductive setting, while some algorithms are designed for the semi-supervised framework and build a classification model which can be applied to out-of-sample examples.

Adsorption [1] is one such recently proposed graph based semi-supervised algorithm which has been successfully used for different tasks, such as recommending YouTube videos to users [1] and large scale assignment of semantic classes to entities within Information Extraction [13]. Adsorption has many desirable properties: it can perform multiclass classification, it can be parallelized and hence can be scaled to handle large data sets which is of particular importance for semi-supervised algorithms. Even though Adsorption works well in practice, to the best of our knowledge it has never been analyzed before and hence our understanding of it is limited. Hoping to fill this gap, we make the following contributions in this paper:

- We analyze the Adsorption algorithm [1] and show that there does not exist an objective function whose local optimization would be the output of the Adsorption algorithm.
- Motivated by this negative result, we propose a new graph based semi-supervised algorithm (Modified Adsorption, MAD), which shares Adsorption's desirable properties, yet with some important differences.
- We state the learning problem as an optimization problem and develop efficient (iterative) methods to solve it. We also list the conditions under which the optimization algorithm – MAD – is guaranteed to converge.
- The transition to an optimization based learning algorithm provides a flexible and general framework that enables us to specify a variety requirements. We demonstrate this framework using data with non-mutually exclusive labels, resulting in the Modified Adsorption for Dependent Labels (MADDL, pronounced *medal*) algorithm.
- We provide experimental evidence demonstrating the effectiveness of our proposed algorithm on various real world datasets.

2 Adsorption Algorithm

Adsorption [1] is a general algorithmic framework for transductive learning where the learner is often given a small set of labeled examples and a very large set of unlabeled examples. The goal is to label all the unlabeled examples, and possibly under the assumption of label-noise, also to relabel the labeled examples.

As many other related algorithms [17, 12, 5], Adsorption assumes that the learning problem is given in a *graph* form, where examples or instances are represented as nodes or vertices and edges code *similarity* between examples. Some of the nodes are associated with a pre-specified label, which is correct in the noise-free case, or can be subject to label-noise. Additional information can be given in the form of *weights* over the labels. Adsorption propagates label-information from the labeled examples to the entire set of vertices via the edges. The labeling is represented using a non-negative score for each label, with high score for some label indicating high-association of a vertex (or its corresponding instance) with that label. If the scores are additively normalized they can be thought of as a conditional distribution over the labels given the node (or example) identity.

More formally, Adsorption is given an undirected graph $G = (V, E, W)$, where a node $v \in V$ corresponds to an example, an edge $e = (a, b) \in V \times V$ indicates that the

label of the two vertices $a, b \in V$ should be similar and the weight $W_{ab} \in \mathbb{R}_+$ reflects the strength of this similarity.

We denote the total number of examples or vertices by $n = |V|$, by n_l the number of examples for which we have prior knowledge of their label and by n_u the number of unlabeled examples to be labeled. Clearly $n_l + n_u = n$. Let $\mathcal{L}$ be the set of possible labels, their total number is denoted by $m = |\mathcal{L}|$ and without loss of generality we assume that the possible labels are $\mathcal{L} = \{1 \dots m\}$. Each instance $v \in V$ is associated with two row-vectors $\mathbf{Y}_v, \hat{\mathbf{Y}}_v \in \mathbb{R}_+^m$. The lth element of the vector $\mathbf{Y}_v$ encodes the prior knowledge for vertex v. The higher the value of $\mathbf{Y}_{vl}$ the stronger we a-priori believe that the label of v should be $l \in \mathcal{L}$ and a value of zero $\mathbf{Y}_{vl} = 0$ indicates no prior about the label l for vertex v. Unlabeled examples have all their elements set to zero, that is $\mathbf{Y}_{vl} = 0$ for $l = 1 \dots m$. The second vector $\hat{\mathbf{Y}}_v \in \mathbb{R}_+^m$ is the output of the algorithm, using similar semantics as $\mathbf{Y}_v$. For example, a high value of $\hat{\mathbf{Y}}_{vl}$ indicates that the algorithm believes that the vertex v should have the label l. We denote by $\mathbf{Y}, \hat{\mathbf{Y}} \in \mathbb{R}_+^{n \times m}$ the matrices whose rows are $\mathbf{Y}_v$ and $\hat{\mathbf{Y}}_v$ respectively. Finally, we denote by $\mathbf{0}_d$ the all-zeros row vector of dimension d.

2.1 Random-Walk View

The Adsorption algorithm can be viewed as a controlled random walk over the graph G. The control is formalized via three possible actions: `inject`, `continue` and `abandon` (denoted by *inj, cont, abnd*) with pre-defined probabilities $p_v^{inj}, p_v^{cont}, p_v^{abnd} \geq 0$ per vertex $v \in V$. Clearly their sum is unit: $p_v^{inj} + p_v^{cont} + p_v^{abnd} = 1$. To label any vertex $v \in V$ (either labeled or unlabeled) we initiate a random-walk starting at v facing three options: with probability p_v^{inj} the random-walk stops and return (i.e. *inject*) the pre-defined vector information $\mathbf{Y}_v$. We constrain $p_v^{inj} = 0$ for unlabeled vertices v. Second, with probability p_v^{abnd} the random-walk *abandons* the labeling process and return the all-zeros vector $\mathbf{0}_m$. Third, with probability p_v^{cont} the random-walk *continues* to one of v's neighbors v' with probability proportional to $W_{v'v} \geq 0$. Note that by definition $W_{v'v} = 0$ if $(v, v') \notin E$. We summarize the above process with the following set of equations. The transition probabilities are,

$$\Pr[v'|v] = \begin{cases} \frac{W_{v'v}}{\sum_{u\,:\,(u,v)\in E} W_{uv}} & (v', v) \in E \\ 0 & \text{otherwise} \end{cases} . \tag{1}$$

The (expected) score $\hat{\mathbf{Y}}_v$ for node $v \in V$ is given by,

$$\hat{\mathbf{Y}}_v = p_v^{inj} \times \mathbf{Y}_v + p_v^{cont} \times \sum_{v'\,:\,(v',v)\in E} \Pr[v'|v]\, \hat{\mathbf{Y}}_{v'} + p_v^{abnd} \times \mathbf{0}_m . \tag{2}$$

2.2 Averaging View

For this view we add a designated symbol called the dummy label denoted by $\nu \notin \mathcal{L}$. This additional label explicitly encodes ignorance about the correct label and it means that a dummy label can be used instead. Explicitly, we add an additional column to all

Algorithm 1 Adsorption Algorithm

Input:
- **Graph:** $G = (V, E, W)$
- **Prior labeling:** $\mathbf{Y}_v \in \mathbb{R}^{m+1}$ for $v \in V$
- **Probabilities:** $p_v^{inj}, p_v^{cont}, p_v^{abnd}$ for $v \in V$

Output:
- **Label Scores:** $\hat{\mathbf{Y}}_v$ for $v \in V$

1: $\hat{\mathbf{Y}}_v \leftarrow \mathbf{Y}_v$ for $v \in V$ {Initialization}
2:
3: **repeat**
4: $\quad D_v \leftarrow \dfrac{\sum_u W_{uv} \hat{\mathbf{Y}}_u}{\sum_u W_{uv}}$ for $v \in V$
5: $\quad$ **for all** $v \in V$ **do**
6: $\quad\quad \hat{\mathbf{Y}}_v \leftarrow p_v^{inj} \times \mathbf{Y}_v + p_v^{cont} \times D_v + p_v^{abnd} \times \mathbf{r}$
7: $\quad$ **end for**
8: **until** convergence

the vectors defined above, and have that $\mathbf{Y}_v, \hat{\mathbf{Y}}_v \in \mathbb{R}_+^{m+1}$ and $\mathbf{Y}, \hat{\mathbf{Y}} \in \mathbb{R}_+^{n \times (m+1)}$. We set $\mathbf{Y}_{v\nu} = 0$, that is, a-priori no vertex is associated with the dummy label, and replace the zero vector $\mathbf{0}_m$ with the vector $\mathbf{r} \in \mathbb{R}_+^{m+1}$ where $\mathbf{r}_l = 0$ for $l \neq \nu$ and $\mathbf{r}_\nu = 1$. In words, if the random-walk is abandoned, then the corresponding labeling vector is zero for all true labels in $\mathcal{L}$, and an arbitrary value of unit for the dummy label ν. This way, there is always positive score for at least one label, the ones in $\mathcal{L}$ or the dummy label.

The averaging view then defines a set of *fixed-point* equations to update the predicted labels. A summary of the equations appears in Algorithm 1. The algorithm is run until convergence which is achieved when the label distribution on each node ceases to change within some tolerance value. Since Adsorption is memoryless, it scales to tens of millions of nodes with dense edges and can be easily parallelized [1].

Baluja et. al. [1] show that up to the additional dummy label, these two views are equivalent. It remains to specify the values of p_v^{inj}, p_v^{cont} and p_v^{abnd}. For the experiments reported in Section 6, we set their value using the following heuristics (adapted from Baluja et. al. [1]) which depends on a parameter β which we set to $\beta = 2$. For each node v we define two quantities: c_v and d_v and define

$$p_v^{cont} \propto c_v \quad ; \quad p_v^{inj} \propto d_v \; .$$

The first quantity $c_v \in [0, 1]$ is monotonically decreasing with the number of neighbors for node v in the graph G. Intuitively, the higher the value of c_v, the lower the number of neighbors of vertex v and higher the information they contain about the labeling of v. The other quantity $d_v \geq 0$ is monotonically increasing with the entropy (for labeled vertices), and in this case we prefer to use the prior-information rather than the computed quantities from the neighbors.

Specifically we first compute the entropy of the transition probabilities for each node,

$$H[v] = -\sum_u \Pr[u|v] \log \Pr[u|v] \ ,$$

and then pass it through the following monotonically decreasing function,

$$f(x) = \frac{\log \beta}{\log(\beta \ + \ e^x))} \ .$$

Note that $f(0) = log(\beta)/\log(\beta + 1)$ and that $f(x)$ goes to zero, as x goes to infinity. We define,

$$c_v = f\left(H[v]\right) \ .$$

Next we define,

$$d_v = \begin{cases} (1 - c_v) \times \sqrt{H[v]} & \text{the vertex v is labeled} \\ 0 & \text{the vertex v is unlabled} \end{cases}$$

Finally, to ensure proper normalization of p_v^{cont}, p_v^{inj} and p_v^{abnd}, we define,

$$z_v = \max(c_v + d_v, 1) \ ,$$

and

$$p_v^{cont} = \frac{c_v}{z_v} \quad ; \quad p_v^{inj} = \frac{d_v}{z_v} \quad ; \quad p_v^{abnd} = 1 - p_v^{cont} - p_v^{inj} \ .$$

Thus, abandonment occurs only when the continuation and injection probabilities are low enough. This is most likely to happen at unlabeled nodes with high degree. Once the random walk reaches such a node (v), the walk is terminated with probability p_v^{abnd}. This, in effect, prevents the Adsorption algorithm from propagating information through high degree nodes. We note that the probabilities p_v^{inj}, p_v^{cont} and p_v^{abnd} for node v may be set with heuristics other than the fan-out entropy heuristics shown above to suit specific application contexts.

3 Analysis of the Adsorption Algorithm

Our next goal is to find an objective function that the Adsorption algorithm minimizes. Our starting point is line 6 of Algorithm 1. We note that when the algorithm converges, both sides of the assignment operator equal each other before the assignment takes place. Thus when the algorithm terminates, we have for all $v \in V$:

$$\hat{\mathbf{Y}}_v = p_v^{inj} \times \mathbf{Y}_v + p_v^{cont} \times \frac{1}{N_v} \sum_u W_{uv} \hat{\mathbf{Y}}_u + p_i^{abnd} \times \mathbf{r} \ ,$$

where

$$N_v = \sum_{v'} W_{v'v} \ .$$

The last set of equalities is equivalent to,

$$G_v\left(\{\hat{\mathbf{Y}}_u\}_{u\in V}\right) = 0 \text{ for } v \in V\ , \tag{3}$$

where we define,

$$G_v\left(\{\hat{\mathbf{Y}}_u\}_{u\in V}\right) = p_v^{inj} \times \mathbf{Y}_v + p_v^{cont} \times \frac{1}{N_v}\sum_u W_{uv}\hat{\mathbf{Y}}_u + p_i^{abnd} \times \mathbf{r} - \hat{\mathbf{Y}}_v\ .$$

Now, if the Adsorption algorithm was minimizing some objective function (denoted by $\mathcal{Q}\left(\{\hat{\mathbf{Y}}_u\}_{u\in V}\right)$), the termination condition of Eq. (3) was in fact a condition on the vector of its partial derivatives where we would identify

$$G_v = \frac{\partial}{\partial \hat{\mathbf{Y}}_v}\mathcal{Q}\ . \tag{4}$$

Since the functions G_v are linear (and thus has continuous derivatives), necessary conditions for the existence of a function $\mathcal{Q}$ such that (4) holds is that the derivatives of G_v are symmetric [8], that is,

$$\frac{\partial}{\partial \hat{\mathbf{Y}}_v}G_u = \frac{\partial}{\partial \hat{\mathbf{Y}}_u}G_v\ .$$

Computing and comparing the derivatives we get,

$$\frac{\partial}{\partial \hat{\mathbf{Y}}_u}G_v = p_v^{cont}\left(\frac{W_{uv}}{N_v} - \delta_{u,v}\right) \neq p_u^{cont}\left(\frac{W_{vu}}{N_u} - \delta_{u,v}\right) = \frac{\partial}{\partial \hat{\mathbf{Y}}_v}G_u\ ,$$

which is true since in general $N_u \neq N_v$ and $p_v^{cont} \neq p_u^{cont}$. We conclude:

Theorem 1. *There does not exists a function $\mathcal{Q}$ with continuous second partial derivatives such that the Adsorption algorithm convergences when gradient of $\mathcal{Q}$ are equal to zero.*

In other words, we searched for a (well-behaved) function $\mathcal{Q}$ such that its local optimal would be the output of the Adsorption algorithm, and showed that this search will always fail. We use this negative results to define a new algorithm, which builds on the Adsorption algorithm and is optimizing a function of the unknowns $\hat{\mathbf{Y}}_v$ for $v \in V$.

4 New Algorithm: Modified Adsorption (MAD)

Our starting point is Sec. 2.2 where we assume to have been given a weighted-graph $G = (V, E, W)$ and a matrix $\mathbf{Y} \in \mathbb{R}_+^{n\times(m+1)}$ and are seeking for a labeling-matrix $\hat{\mathbf{Y}} \in \mathbb{R}_+^{n\times(m+1)}$. In this section it is more convenient to decompose the matrices $\mathbf{Y}$ and $\hat{\mathbf{Y}}$ into their columns, rather than rows. Specifically, we denote by $\mathbf{Y}_l \in \mathbb{R}_+^n$ the lth column of $\mathbf{Y}$ and similarly by $\hat{\mathbf{Y}}_l \in \mathbb{R}_+^n$ the lth column of $\hat{\mathbf{Y}}$. We distinguish the rows and columns of the matrices $\mathbf{Y}$ and $\hat{\mathbf{Y}}$ using their indices, the columns are indexed with the label index l, while the rows are indexed are with a vertex index v (or u).

We build on previous research [3, 17, 11] and construct an objective that reflects three requirements as follows. First, for the labeled vertices we like the output of the algorithm to be close to the a-priori given labels, that is $\mathbf{Y}_v \approx \hat{\mathbf{Y}}_v$. Second, for pair of vertices that are close according to the input graph, we would like their labeling to be close, that is $\hat{\mathbf{Y}}_u \approx \hat{\mathbf{Y}}_v$ if W_{uv} is large. Third, we want the output to be as uninformative as possible, this serves as additional regularization, that is $\hat{\mathbf{Y}}_v \approx \mathbf{r}$. We now further develop the objective in light of the three requirements.

We use the Euclidian distance to measure discrepancy between two quantities, and start with the first requirement above,

$$\begin{aligned}\sum_v p_v^{inj} \sum_l \left(\mathbf{Y}_{vl} - \hat{\mathbf{Y}}_{vl}\right)^2 = & \sum_l \sum_v p_v^{inj} \left(\mathbf{Y}_{vl} - \hat{\mathbf{Y}}_{vl}\right)^2 \\ = & \sum_l \left(\mathbf{Y}_l - \hat{\mathbf{Y}}_l\right)^\top \mathbf{S} \left(\mathbf{Y}_l - \hat{\mathbf{Y}}_l\right) ,\end{aligned}$$

where we define the diagonal matrix $\mathbf{S} \in \mathbb{R}^{n \times n}$ and $\mathbf{S}_{vv} = p_v^{inj}$ if vertex v is labeled and $\mathbf{S}_{vv} = 0$ otherwise. The matrix $\mathbf{S}$ captures the intuition that for different vertices we enforce the labeling of the algorithm to match the a-priori labeling with different extent.

Next, we modify the similarity weight between vertices to take into account the difference in degree of various vertices. In particular we define $\mathbf{W}'_{vu} = p_v^{cont} \times \mathbf{W}_{vu}$. Thus, a vertex u will *not* be similar to a vertex v if either the input weights $\mathbf{W}_{vu}$ are low or the vertex v has a large-degree (p_v^{cont} is low). We write the second requirement as,

$$\begin{aligned}&\sum_{v,u} \mathbf{W}'_{vu} \left\|\hat{\mathbf{Y}}_v - \hat{\mathbf{Y}}_u\right\|^2 = \sum_{v,u} \mathbf{W}'_{vu} \sum_l \left(\hat{\mathbf{Y}}_{vl} - \hat{\mathbf{Y}}_{ul}\right)^2 = \sum_l \sum_{v,u} \mathbf{W}'_{vu} \left(\hat{\mathbf{Y}}_{vl} - \hat{\mathbf{Y}}_{ul}\right)^2 \\ &= \sum_l \sum_v \left(\sum_u \mathbf{W}'_{vu}\right) \|\hat{\mathbf{Y}}_{vl}\|^2 + \sum_l \sum_u \left(\sum_v \mathbf{W}'_{vu}\right) \|\hat{\mathbf{Y}}_{ul}\|^2 - 2 \sum_l \sum_{u,v} \mathbf{W}'_{vu} \hat{\mathbf{Y}}_{ul} \hat{\mathbf{Y}}_{vl} \\ &= \sum_l \hat{\mathbf{Y}}_l^\top \mathbf{L} \hat{\mathbf{Y}}_l ,\end{aligned}$$

where,

$$\mathbf{L} = \mathbf{D} + \bar{\mathbf{D}} - \mathbf{W}' - \mathbf{W}'^\top ,$$

and $\mathbf{D}$, $\bar{\mathbf{D}}$ are $n \times n$ diagonal matrices with

$$\mathbf{D}_{vv} = \sum_u \mathbf{W}'_{uv} \quad , \quad \bar{\mathbf{D}}_{vv} = \sum_u \mathbf{W}'_{vu} .$$

Finally we define the matrix $\mathbf{R} \in \mathbb{R}_+^{n \times (m+1)}$ where the vth row of $\mathbf{R}$ equals $p_v^{abnd} \times \mathbf{r}$ (we define $\mathbf{r}$ in Sec 2.2). In other words the first m columns of $\mathbf{R}$ equal zero, and the last (m+1th column) equal the elements of p_v^{abnd}. The third requirement above is thus written as,

$$\sum_{vl} \left(\hat{\mathbf{Y}}_{vl} - \mathbf{R}_{vl}\right)^2 = \sum_l \left\|\hat{\mathbf{Y}}_l - \mathbf{R}_l\right\|^2 .$$

We combine the three terms above into a single objective (which we would like to minimize), giving to each term a different importance using the weights μ_1, μ_2, μ_3.

$$C(\hat{\mathbf{Y}}) = \sum_l \left[\mu_1 \left(\mathbf{Y}_l - \hat{\mathbf{Y}}_l \right)^\top \mathbf{S} \left(\mathbf{Y}_l - \hat{\mathbf{Y}}_l \right) \; + \; \mu_2 \hat{\mathbf{Y}}_l^T \, \mathbf{L} \, \hat{\mathbf{Y}}_l \; + \; \mu_3 \left|\left| \hat{\mathbf{Y}}_l - \mathbf{R}_l \right|\right|_2^2 \right] \quad (5)$$

The objective in Equation 5 is similar to the Quadratic Cost Criteria [3], with the exception that the matrices $\mathbf{S}$ and $\mathbf{L}$ have different constructions. We remind the reader that $\hat{\mathbf{Y}}_l, \mathbf{Y}_l, \mathbf{R}_l$ are the lth columns (each of size $n \times 1$) of the matrices $\hat{\mathbf{Y}}, \mathbf{Y}$ and $\mathbf{R}$ respectively.

4.1 Solving the Optimization Problem

We now develop an algorithm to optimize (5) similar to the quadratic cost criteria [3]. Differentiating Equation 5 w.r.t. $\hat{\mathbf{Y}}_l$ we get,

$$\begin{aligned} \frac{1}{2} \frac{\delta C(\hat{\mathbf{Y}})}{\delta \hat{\mathbf{Y}}_l} &= \mu_1 \mathbf{S}(\hat{\mathbf{Y}}_l - \mathbf{Y}_l) + \mu_2 \mathbf{L} \hat{\mathbf{Y}}_l + \mu_3 (\hat{\mathbf{Y}}_l - \mathbf{R}_l) \\ &= (\mu_1 \mathbf{S} + \mu_2 \mathbf{L} + \mu_3 \mathbf{I}) \hat{\mathbf{Y}}_l - (\mu_1 \mathbf{S} \mathbf{Y}_l + \mu_3 \mathbf{R}_l) \; . \end{aligned} \quad (6)$$

Differentiating once more we get,

$$\frac{1}{2} \frac{\delta C(\hat{\mathbf{Y}})}{\delta \hat{\mathbf{Y}}_l \delta \hat{\mathbf{Y}}_l} = \mu_1 \mathbf{S} + \mu_2 \mathbf{L} + \mu_3 \mathbf{I} \; ,$$

and since both $\mathbf{S}$ and $\mathbf{L}$ are symmetric and positive semidefinite matrices (PSD), we get that the Hessian is PSD as well. Hence, the optimal minima is obtained by setting the first derivative (i.e. Equation (6)) to 0 as follows,

$$(\mu_1 \mathbf{S} + \mu_2 \mathbf{L} + \mu_3 \mathbf{I}) \; \hat{\mathbf{Y}}_l = (\mu_1 \mathbf{S} \mathbf{Y}_l + \mu_3 \mathbf{R}_l) \; .$$

Hence, the new labels ($\hat{\mathbf{Y}}$) can be obtained by a matrix inversion followed by matrix multiplication. However, this can be quite expensive when large matrices are involved. A more efficient way to obtain the new label scores is to solve a set of linear equations using Jacobi iteration which we now describe.

4.2 Jacobi Method

Given the following linear system (in x)

$$\mathbf{M} x = b$$

the Jacobi iterative algorithm defines the approximate solution at the (t+1)th iteration given the solution at tth iteration as follows,

$$x_i^{(t+1)} = \frac{1}{\mathbf{M}_{ii}} \left(b_i - \sum_{j \neq i} \mathbf{M}_{ij} x_j^{(t)} \right) \; . \quad (7)$$

We apply the iterative algorithm to our problem by substituting $x = \hat{\mathbf{Y}}_l$, $\mathbf{M} = \mu_1\mathbf{S} + \mu_2\mathbf{L} + \mu_3\mathbf{I}$ and $b = \mu_1\mathbf{S}\mathbf{Y}_l + \mu_3\mathbf{R}_l$ in (7),

$$\hat{\mathbf{Y}}_{vl}^{(t+1)} = \frac{1}{\mathbf{M}_{vv}} \left(\mu_1(\mathbf{S}\mathbf{Y}_l)_v + \mu_3\mathbf{R}_{vl} - \sum_{u \neq v} \mathbf{M}_{vu}\hat{\mathbf{Y}}_{ul}^{(t)} \right) \tag{8}$$

Let us compute the values of $(\mathbf{S}\mathbf{Y}_l)_i$, $\mathbf{M}_{ij(j\neq i)}$ and $\mathbf{M}_{ii}$. First,

$$\mathbf{M}_{vu(v\neq u)} = \mu_1\mathbf{S}_{vu} + \mu_2\mathbf{L}_{vu} + \mu_3\mathbf{I}_{vu} \; .$$

Note that since $\mathbf{S}$ and $\mathbf{I}$ are diagonal, we have that $\mathbf{S}_{vu} = 0$ and $\mathbf{I}_{vu} = 0$ for $u \neq v$. Substituting the value of $\mathbf{L}$ we get,

$$\mathbf{M}_{vu(v\neq u)} = \mu_2\mathbf{L}_{vu} = \mu_2 \left(\mathbf{D}_{vu} + \bar{\mathbf{D}}_{vu} - \mathbf{W}^{'}_{vu} - \mathbf{W}^{'}_{uv} \right) \; ,$$

and as before the matrices $\mathbf{D}$ and $\bar{\mathbf{D}}$ are diagonal and thus $\mathbf{D}_{vu} + \bar{\mathbf{D}}_{vu} = 0$. Finally, substituting the values of $\mathbf{W}^{'}_{vu}$ and $\mathbf{W}^{'}_{uv}$ we get,

$$\mathbf{M}_{vu(v\neq u)} = -\mu_2 \times (p_v^{cont} \times \mathbf{W}_{vu} + p_u^{cont} \times \mathbf{W}_{uv}) \; . \tag{9}$$

We now compute the second quantity,

$$(\mathbf{S}\mathbf{Y}_l)_{vu} = \mathbf{S}_{vv}\mathbf{Y}_{vv} + \sum_{t\neq v} \mathbf{S}_{vt}\mathbf{Y}_{tu} = p_v^{inj} \times \mathbf{Y}_{vv} \; ,$$

where the second term equals zero since $\mathbf{S}$ is diagonal. Finally, the third term,

$$\begin{aligned} \mathbf{M}_{vv} &= \mu_1\mathbf{S}_{vv} + \mu_2\mathbf{L}_{vv} + \mu_3\mathbf{I}_{vv} \\ &= \mu_1 \times p_v^{inj} + \mu_2(\mathbf{D}_{vv} + \bar{\mathbf{D}}_{vv} - \mathbf{W}^{'}_{vv} - \mathbf{W}^{'}_{vv}) + \mu_3 \\ &= \mu_1 \times p_v^{inj} + \mu_2 \sum_{u\neq v} (p_v^{cont}\mathbf{W}_{vu} + p_u^{cont}\mathbf{W}_{uv}) + \mu_3 \; . \end{aligned}$$

Plugging the above equations into (8) and using the fact that the diagonal elements of $\mathbf{W}$ are zero, we get,

$$\hat{\mathbf{Y}}_v^{(t+1)} = \frac{1}{\mathbf{M}_{vv}} \left(\mu_1 p_v^{inj} \; \mathbf{Y}_v + \mu_2 \sum_u \left(p_v^{cont}\mathbf{W}_{vu} + p_u^{cont}\mathbf{W}_{uv} \right) \hat{\mathbf{Y}}_u^{(t)} + \mu_3 \; p_v^{abnd} \; \mathbf{r} \right) . \tag{10}$$

We call the new algorithm MAD for Modified-Adsorption and it is summarized in Algorithm 2. Note that for graphs G that are invariant to permutations of the vertices, and setting $\mu_1 = 2 \times \mu_2 = \mu_3 = 1$, MAD reduces to the Adsorption algorithm.

4.3 Convergence

A sufficient condition for the iterative process of Equation (7) to converge is that $\mathbf{M}$ is strictly diagonally dominant [10], that is if,

$$|\mathbf{M}_{vv}| > \sum_{u\neq v} |\mathbf{M}_{vu}| \quad \text{for all values of } v$$

Algorithm 2 Modified Adsorption (MAD) Algorithm

Input:
- **Graph:** $G = (V, E, W)$
- **Prior labeling:** $\mathbf{Y}_v \in \mathbb{R}^{m+1}$ for $v \in V$
- **Probabilities:** $p_v^{inj}, p_v^{cont}, p_v^{abnd}$ for $v \in V$

Output:
- **Label Scores:** $\hat{\mathbf{Y}}_v$ for $v \in V$

1: $\hat{\mathbf{Y}}_v \leftarrow \mathbf{Y}_v$ for $v \in V$ {Initialization}
2: $\mathbf{M}_{vv} \leftarrow \mu_1 \times p_v^{inj} + \mu_2 \sum_{u \neq v} (p_v^{cont} \mathbf{W}_{vu} + p_u^{cont} \mathbf{W}_{uv}) + \mu_3$
3: **repeat**
4: $\quad D_v \leftarrow \sum_u \left(p_v^{cont} \mathbf{W}_{vu} + p_u^{cont} \mathbf{W}_{uv}\right) \hat{\mathbf{Y}}_u$
5: $\quad$ **for all** $v \in V$ **do**
6: $\quad\quad \hat{\mathbf{Y}}_v \leftarrow \frac{1}{\mathbf{M}_{vv}} \left(\mu_1 \times p_v^{inj} \times \mathbf{Y}_v + \mu_2 \times D_v + \mu_3 \times p_v^{abnd} \times \mathbf{r}\right)$
7: $\quad$ **end for**
8: **until** convergence

We have,

$$|\mathbf{M}_{vv}| - \sum_{u \neq v} |\mathbf{M}_{vu}| = \mu_1 \times p_v^{inj} + \mu_2 \times \sum_{u \neq v} \left(p_v^{cont} \times \mathbf{W}_{vu} + p_u^{cont} \times \mathbf{W}_{uv}\right) + \mu_3 - \mu_2 \times \sum_{u \neq v} \left(p_v^{cont} \times \mathbf{W}_{vu} + p_u^{cont} \times \mathbf{W}_{uv}\right)$$
$$= \mu_1 \times p_v^{inj} + \mu_3 \tag{11}$$

Note that $p_v^{inj} \geq 0$ for all v and that μ_3 is a free parameter in (11). Thus we can guarantee a strict diagonal dominance (and hence convergence) by setting $\mu_3 > 0$.

5 Extensions: Non-Mutually Exclusive Labels

In many learning settings, labels are not mutually exclusive. For example, in hierarchical classification, labels are organized in a tree. In this section, we extend the MAD algorithm to handle dependence among labels. This can be easily done using our new formulation which is based on objective optimization. Specifically, we shall add additional terms to the objective for each pair of dependent labels. Let C be a $m \times m$ matrix where m is the number of labels (excluding the dummy label) as before. Each entry, $C_{ll'}$, of this matrix C represents the dependence or similarity among the labels l and l'. By encoding dependence in this pairwise fashion, we can capture dependencies among labels represented as arbitrary graphs. The extended objective is shown in Equation 12.

$$C(\hat{\mathbf{Y}}) = \sum_l \Bigg[\mu_1 \left(\mathbf{Y}_l - \hat{\mathbf{Y}}_l\right)^\top \mathbf{S} \left(\mathbf{Y}_l - \hat{\mathbf{Y}}_l\right) \; + \; \mu_2 \hat{\mathbf{Y}}_l^T \, \mathbf{L} \, \hat{\mathbf{Y}}_l \; + \; \mu_3 \left|\left| \hat{\mathbf{Y}}_l - \mathbf{R}_l \right|\right|_2^2 + \mu_4 \sum_i \sum_{l,l'} C_{ll'} (\hat{\mathbf{Y}}_{il} - \hat{\mathbf{Y}}_{il'})^2 \Bigg] \tag{12}$$

The last term in Equation 12 penalizes the algorithm if similar labels (as determined by the matrix C) are assigned different scores, with severity of the penalty controlled by μ_4. Now, analyzing the objective in Equation 12 in the manner outlined in Section 4, we arrive at the update rule shown in Equation 13.

$$\hat{\mathbf{Y}}_{vl}^{(t+1)} = \frac{1}{\mathbf{M}_{vv}^{l}} \left(\mu_1 p_v^{inj} \, \mathbf{Y}_{vl} + \mu_2 \sum_u \left(p_v^{cont} \mathbf{W}_{vu} + p_u^{cont} \mathbf{W}_{uv} \right) \hat{\mathbf{Y}}_{ul}^{(t)} + \mu_3 \, p_v^{abnd} \, \mathbf{r}_l + \mu_4 \sum_{l'} C_{ll'} \hat{\mathbf{Y}}_{vl'}^{(t)} \right) \tag{13}$$

where,

$$\mathbf{M}_{vv}^{l} = \mu_1 \times p_v^{inj} + \mu_2 \times \sum_{u \neq v} (p_v^{cont} \mathbf{W}_{vu} + p_u^{cont} \mathbf{W}_{uv}) + \mu_3 + \mu_4 \sum_{l'} C_{ll'}$$

Replacing Line 6 in MAD (Algorithm 2) with Equation 13, we end up with a new algorithm: Modified Adsorption for Dependent Labels (MADDL). In Section 6.4, we shall use MADDL to obtain smooth ranking for sentiment classification.

6 Experimental Results

We compare MAD with various state-of-the-art learning algorithms on two tasks, text classification (Sec. 6.1) and sentiment analysis (Sec. 6.2), and demonstrate its effectiveness. In Sec. 6.3, we also provide experimental evidence showing that MAD is quite insensitive to wide variation of values of its hyper-parameters. In Sec. 6.4, we present evidence showing how MADDL can be used to obtain smooth ranking for sentiment prediction, a particular instantiation of classification with non-mutually exclusive labels. For the experiments reported in this section involving Adsorption, MAD and MADDL, the a-priori label matrix $\mathbf{Y}$ was column-normalized so that all labels have equal overall injection score. Also, the dummy label was ignored during evaluation as its main role is to add regularization during learning phase only.

6.1 Text Classification

World Wide Knowledge Base (WebKB) is a text classification dataset widely used for evaluating transductive learning algorithms. Most recently, the dataset was used by Subramanya and Bilmes [11], who kindly shared their preprocessed complete WebKB graph with us. There are a total of $4,204$ vertices in the graph, with the nodes labeled with one of four categories: *course, faculty, project, student*. A K-NN graph is created from this complete graph by retaining only top K neighbors of each node, where the value of K is treated as a hyper-parameter.

We follow the experimental protocol in [11]. The dataset was randomly partitioned into four sets. A transduction set was generated by first selecting one of the four splits at random and then sampling n_l documents from it; the remaining three sets are used as the

Class	SVM	TSVM	SGT	LP	AM	Adsorption	MAD
course	46.5	43.9	29.9	45.0	**67.6**	61.1	67.5
faculty	14.5	31.2	42.9	40.3	42.5	**52.8**	42.2
project	15.8	17.2	17.5	27.8	42.3	**52.6**	45.5
student	15.0	24.5	56.6	51.8	55.0	39.8	**59.6**
average	23.0	29.2	36.8	41.2	51.9	51.6	**53.7**

Table 1. PRBEP for the WebKB data set with n_l = 48 training and 3148 testing instances. All results are averages over 20 randomly generated transduction sets. The last row is the macro-average over all the classes. MAD is the proposed approach. Results for SVM, TSVM, SGT, LP and AM are reproduced from Table 2 of [11].

test set for evaluation. This process was repeated 21 times to generate as many training-test splits. The first split was used to tune the hyper-parameters, with search over the following: $K \in \{10, 50, 100, 500, 1000, 2000, 4204\}$, $\mu_2, \mu_3 \in \{1e{-}8, 1e{-}4, 1e{-}2, 1, 10, 1e2, 1e3\}$. The value of μ_1 was set to 1 for this experiment. Both for Adsorption and MAD, the optimal value of K was $1,000$. Furthermore, the optimal value for the other parameters were found to be $\mu_2 = \mu_3 = 1$. As in previous work [11], we use Precision-Recall Break Even Point (PRBEP) [9] as the evaluation metric. Same evaluation measure, dataset and the same experimental protocol makes the results reported here directly comparable to those reported previously [11]. For easier readability, the results from Table 2 of Subramanya and Bilmes [11] are cited in Table 1 of this paper, comparing performance of Adsorption based methods (Adsorption and MAD) to many previously proposed approaches: SVM [6], Transductive-SVM [6], Spectral Graph Transduction (SGT) [7], Label Propagation (LP) [16] and Alternating Minimization (AM) [11]. The first four rows in Table 1 shows PRBEP for individual categories, with the last line showing the macro-averaged PRBEP across all categories. The MAD algorithm achieves the best performance overall (for n_l = 48).

Performance comparison of MAD and Adsorption for increasing n_l are shown in Figure 1. Comparing these results against Fig. 2 in Subramanya and Bilmes [11], it seems that MAD outperforms all other methods compared (except AM [11]) for all values of n_l. MAD performs better than AM for n_l = 48, but achieves second best solution for the other three values of n_l. We are currently investigating why MAD is best for settings with fewer labeled examples.

6.2 Sentiment Analysis

The goal of sentiment analysis is to automatically assign polarity scores to text collections, with a high score reflecting positive sentiment (user likes) and a low score reflecting negative sentiment (user dislikes). In this section, we report results on sentiment classification in the transductive setting. From Section 6.1 and [11], we observe that Label Propagation (LP) [16] is one of the best performing L2-norm based transductive learning algorithm. Hence, we compare the performance of MAD against Adsorption and LP.

For the experiments in this section, we use a set of $4,768$ user reviews from the electronics domain [4]. Each review is assigned one of the four scores: 1 (worst), 2,

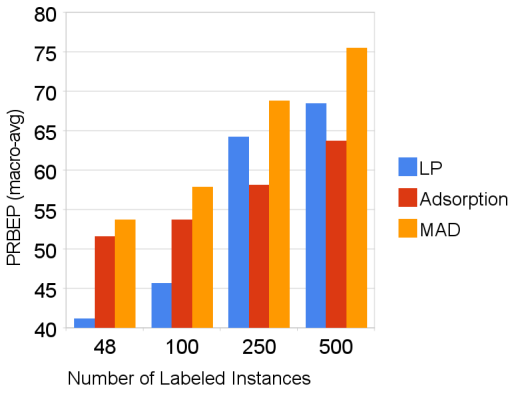

Fig. 1. PRBEP (macro-averaged) for the WebKB dataset with 3148 testing instances. All results are averages over 20 randomly generated transduction sets.

3, 4 (best). We create a K-NN graph from these reviews by using cosine similarity as the measure of similarity between reviews. We created 5 training-test splits from this data using the process described in Section 6.1. One split was used to tune the hyper-parameters while the rest were used for training and evaluation. Hyper-parameter search was carried over the following ranges: $K \in \{10, 100, 500\}$, $\mu_1 \in \{1, 100\}$, $\mu_2 \in \{1e{-}4, 1, 10\}$, $\mu_3 \in \{1e{-}8, 1, 100, 1e3\}$. Precision is used as the evaluation metric. Comparison of different algorithms for varying number of labeled instances are shown in Figure 2. From this, we note that MAD and Adsorption outperform LP, while Adsorption and MAD are competitive

6.3 Parameter Sensitivity

We evaluated the sensitivity of MAD to variations of its μ_2 and μ_3 hyper-parameters, with all other hyper-parameters fixed. We used a 2000-NN graph constructed from the WebKB dataset and a 500-NN graph constructed from the Sentiment dataset. In both cases, 100 nodes were labeled. We tried three values each for μ_2 and μ_3, ranging in at least 3 order of magnitude. For the WebKB, the PRBEP varied between $43.1{-}49.9$ and for the sentiment data, the precision varied in the range $31.4{-}36.4$ with $\mu_2 \leq \mu_3$ while precision dropped to 25 with $\mu_2 > \mu_3$. This underscores the need for regularization in these models, which is enforced with high μ_3. We note that in both cases the algorithm is less sensitive to the value of μ_2 than the value of μ_3. In general, we have found that setting μ_3 to one or two order magnitude more than μ_2 is a reasonable choice. We have also found that the MAD algorithm is quite insensitive to variations in μ_1. For example on the sentiment dataset, we tried two values for μ_1 ranging two order of magnitude, with other hyper-parameters fixed. In this case, precision varied in the range 36.2 - 36.3.

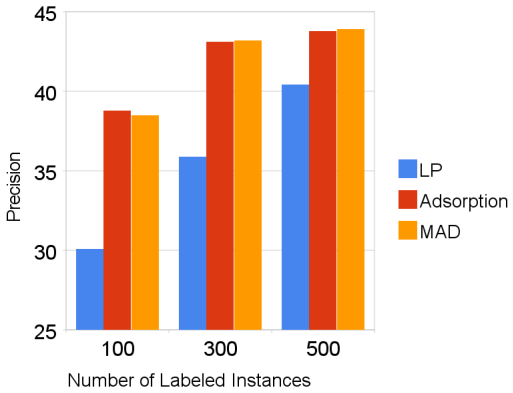

Fig. 2. Precision for the Sentiment Analysis dataset with 3568 testing instances. All results are averages over 4 randomly generated transduction sets.

	μ_4					
	0	1	10	100	1e3	1e4
Prediction Loss (L1) at rank 1	0.93	0.93	0.92	0.90	0.90	0.90
Prediction Loss (L1) at rank 2	1.21	1.20	1.12	0.96	0.97	0.97

Table 2. Average prediction loss at ranks 1 & 2 (for various values of μ_4) for sentiment prediction. All results are averaged over 4 runs. See Section 6.4 for details.

6.4 Smooth Ranking for Sentiment Analysis

We revisit the sentiment prediction problem in Section 6.2, but with the additional requirement that ranking of the labels (1, 2, 3, 4) generated by the algorithm should be smooth i.e. we prefer the ranking $1 > 2 > 3 > 4$ over the ranking $1 > 4 > 3 > 2$, where $3 > 2$ means that the algorithm ranks label 3 higher than label 2. The ranking $1 > 2 > 3 > 4$ is smoother as it doesn't involve rough transition $1 > 4$ which is present in $1 > 4 > 3 > 2$. We use the framework of stating requirements as an objective to be optimized. We use the MADDL algorithm of Sec. 5 initializing the matrix C as follows (assuming that labels 1 and 2 are related, while labels 3 and 4 are related):

$$C_{12} = C_{21} = 1 \quad , \quad C_{34} = C_{43} = 1$$

with all other entries in matrix C set to 0. Such constraints (along with appropriate μ_4 in Equation (12)) will force the algorithm to assign similar scores to dependent labels, thereby assigning them adjacent ranks in the final output. MAD and MADDL were then used to predict ranked labels for vertices on a 1000-NN graph constructed from the sentiment data used in Sec. 6.2, with 100 randomly selected nodes labeled. For this experiment we set $\mu_1 = \mu_2 = 1$, $\mu_3 = 100$. The $L1$-loss between the gold label and labels predicted at ranks $r = 1, 2$ for increasing values of μ_4 are given in Table 2. Note that, MADDL with $\mu_4 = 0$ corresponds to MAD. From Table 2 we observe that with

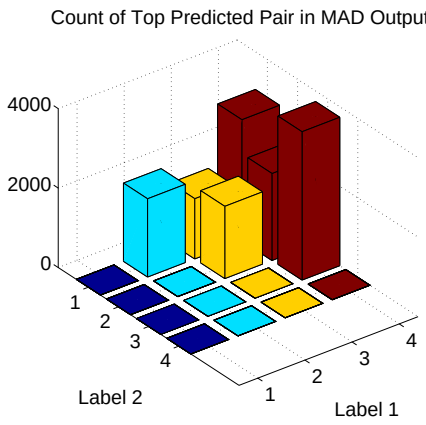

Fig. 3. Plot of counts of top predicted label pairs (order ignored) in MAD's predictions with $\mu_1 = \mu_2 = 1$, $\mu_3 = 100$.

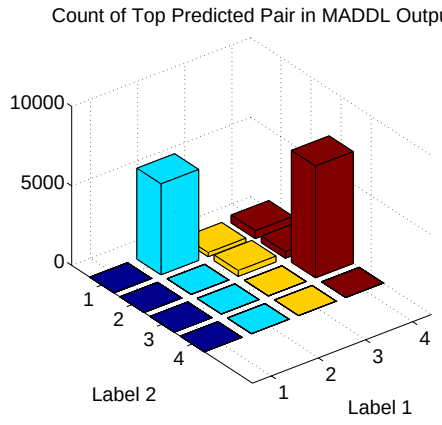

Fig. 4. Plot of counts of top label pairs (order ignored) in MADDL's predictions (Section 5), with $\mu_1 = \mu_2 = 1$, $\mu_3 = 100, \mu_4 = 1e3$.

increasing μ_4, MADDL is ranking at $r = 2$ a label which is related (as per C) to the top ranked label at $r = 1$, but at the same time maintain the quality of prediction at $r = 1$ (first row of Table 2), thereby ensuring a smoother ranking. From Table 2, we also observe that MADDL is insensitive to variations of μ_4 beyond a certain range. This suggests that μ_4 may be set to a (high) value and that tuning it may not be necessary.

Another view of the same phenomenon is shown in Fig. 3 and Fig. 4. In these figures, we plot the counts of top predicted label pair (order of prediction is ignored for better readability) generated by the MAD and MADDL algorithms. By comparing these two figures we observe that label pairs (e.g. (2,1) and (4,3)) favored by C (above) are more frequent in MADDL's predictions than in MAD's. At the same time, non-smooth predictions (e.g. (4, 1)) are virtually absent in MADDL's predictions while they are quite frequent in MAD's. These clearly demonstrate MADDL's ability to generate smooth predictions in a principled way, and more generally the ability to handle data with non-mutually exclusive or dependent labels.

7 Related Work

LP [16] is one of the first graph based semi-supervised algorithms. Even though there are several similarities between LP and MAD, there are important differences: (1) LP doesn't allow the labels on seeded nodes to change (while MAD does). As was pointed out previously [3], this can be problematic in case of noisy seeds. (2) There is no way for LP to express label uncertainty about a node. MAD can accomplish this by assigning high score to the dummy label. More recently, a KL minimization based algorithm was presented in [11]. Further investigation is necessary to determine the merits of each approach. For a general introduction to the area of graph-based semi-supervised learning, the reader is referred to a survey by Zhu [15].

8 Conclusion

In this paper we have analyzed the Adsorption algorithm [1] and proposed a new graph based semi-supervised learning algorithm, MAD. We have developed efficient (iterative) solution to solve our convex optimization based learning problem. We have also listed the conditions under which the algorithm is guaranteed to converge. Transition to an optimization based learning algorithm allows us to easily extend the algorithm to handle data with non-mutually exclusive labels, resulting in the MADDL algorithm. We have provided experimental evidence demonstrating effectiveness of our proposed methods. As part of future work, we plan to evaluate the proposed methods further and apply the MADDL method in problems with dependent labels (e.g. Information Extraction).

Acknowledgment

This research is partially supported by NSF grant #IIS-0513778. The authors would like to thank F. Pereira and D. Sivakumar for useful discussions.

References

1. S. Baluja, R. Seth, D. Sivakumar, Y. Jing, J. Yagnik, S. Kumar, D. Ravichandran, and M. Aly. Video suggestion and discovery for youtube: taking random walks through the view graph. In *WWW*, 2008.
2. M. Belkin, P. Niyogi, and V. Sindhwani. Manifold regularization: A geometric framework for learning from labeled and unlabeled examples. *JMLR*, 7:2399–2434, 2006.
3. Y. Bengio, O. Delalleau, and N. Roux. Semi-Supervised Learning, chapter Label Propogation and Quadratic Criterion, 2007.
4. J. Blitzer, M. Dredze, and F. Pereira. Biographies, bollywood, boom-boxes and blenders: Domain adaptation for sentiment classification. In *ACL*, 2007.
5. P. Indyk and J. Matousek. Low-distortion embeddings of finite metric spaces. *Handbook of Discrete and Computational Geometry*, 2004.
6. T. Joachims. Transductive inference for text classification using support vector machines. In *ICML*, 1999.
7. T. Joachims. Transductive learning via spectral graph partitioning. In *ICML*, 2003.
8. V. J. Katz. The history of stokes' theorem. *Mathematics Magazine*, 52(3):146–156, 1979.
9. V. Raghavan, P. Bollmann, and G. Jung. A critical investigation of recall and precision as measures of retrieval system performance. *ACM TOIS*, 7(3):205–229, 1989.
10. Y. Saad. *Iterative Methods for Sparse Linear Systems*. Society for Industrial Math., 2003.
11. A. Subramanya and J. Bilmes. Soft-Supervised Learning for Text Classification. In *EMNLP*, 2008.
12. M. Szummer and T. Jaakkola. Partially labeled classification with markov random walks. *NIPS*, 2002.
13. P. P. Talukdar, J. Reisinger, M. Pasca, D. Ravichandran, R. Bhagat, and F. Pereira. Weakly supervised acquisition of labeled class instances using graph random walks. In *EMNLP*, 2008.
14. J. Wang, T. Jebara, and S. Chang. Graph transduction via alternating minimization. In *ICML*, 2008.
15. X. Zhu. Semi-supervised learning literature survey. 2005.
16. X. Zhu and Z. Ghahramani. Learning from labeled and unlabeled data with label propagation. Technical report, CMU CALD tech report, 2002.
17. X. Zhu, Z. Ghahramani, and J. Lafferty. Semi-supervised learning using gaussian fields and harmonic functions. *ICML*, 2003.