Graph-Based Transduction with Confidence

Matan Orbach and Koby Crammer

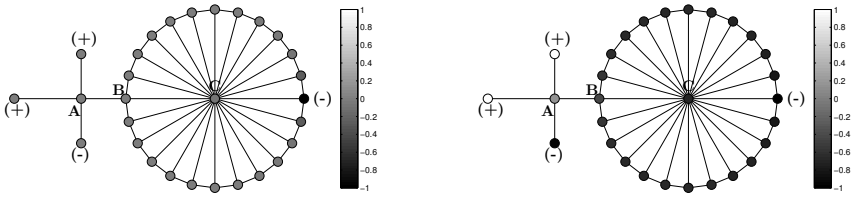
Dept. of Electrical Engineering
Technion - Israel Institute of Technology
Haifa 32000, Israel
{matanorb@tx,koby@ee}.technion.ac.il

Abstract. We present a new multi-class graph-based transduction algorithm. Examples are associated with vertices in an undirected weighted graph and edge weights correspond to a similarity measure between examples. Typical algorithms in such a setting perform label propagation between neighbours, ignoring the quality, or estimated quality, in the labeling of various nodes. We introduce an additional quantity of confidence in label assignments, and learn them jointly with the weights, while using them to dynamically tune the influence of each vertex on its neighbours. We cast learning as a convex optimization problem, and derive an efficient iterative algorithm for solving it. Empirical evaluations on seven NLP data sets demonstrate our algorithm improves over other state-of-the-art graph-based transduction algorithms.

1 Introduction

Supervised machine learning algorithms are powerful. Given a training set composed of example-label pairs, many methods have been proposed and successfully used for a variety of real-world tasks in different domains. Typically, performance improves as the size of the training set increases. However, this improvement comes with a price, labeling a large amount of data is slow, costly and prone to human errors, especially in complex tasks. This is in contrast to the fact that a large amount of unlabeled data can be cheaply collected in many different domains. As a result, it has become beneficial to research and develop semi-supervised learning (SSL) algorithms. These algorithms are designed to learn from a small set of labeled data and a much larger set of unlabeled data.

Graph-based methods are an important class of SSL algorithms [3,11,12,14]. These methods construct an undirected weighted graph reflecting a similarity relation between examples, both labeled and unlabeled. Each example is associated with a vertex. Edge weights correspond to a similarity measure between examples. Learning is then based on a *smoothness assumption* [3]: two neighbour vertices are likely to have the same label. In this paper we consider the transductive graph-based setting. The goal of the learning algorithm is to assign labels to the unlabeled vertices of the graph. Starting with the small set of labeled vertices, the learning algorithm propagates label information from the small set of known labels to the rest of the graph. In other settings, algorithms learn (also) a model and are thus also capable of classifying new, previously unseen, examples.



(a) MAD (b) Ours (TACO)

Fig. 1. Illustration of label-propogation properties for two algorithms : MAD [12] and ours (named TACO). MAD is sensitive to vertex-degree differences while our algorithm is robust to it.

Two new transductive graph-based algorithms have been introduced recently: Adsorption [1] and Modified Adsorption (MAD) [12]. The label propagation in Adsorption can be viewed as a controlled random walk over the graph. The transition probabilities in the random walk are determined by edge weights and the degree of each vertex. Specifically, transition probabilities to and from high degree vertices are reduced, assuming a larger neighbours set is more likely to contain disagreements among the neighbours. We further discuss this assumption shortly. MAD builds upon Adsorption, formulating a convex optimization problem including the controlled random walk transition probabilities, and solving this problem with an efficient iterative algorithm.

We present a new multi-class graph-based transductive algorithm. The main motivation for our new algorithm is the following question: should we always discourage high degree vertices? While both Adsorption and MAD are based on a definite positive answer (and label propagation (LP) [14] is somewhat based on an implicit negative answer), our goal is aimed at letting the data decide for us. Assume we have a high degree vertex, and our label propagation algorithm is in a state where almost all of its neighbours have the same estimated label. Here, we can be highly confident in our estimated label because we have many agreeing neighbours. Thus, in this case, the influence of the high degree vertex on its neighbours should not be reduced. Only when we have a high degree vertex, combined together with a large measure of disagreement among neighbours, we should lower the effect of this vertex on its neighbours.

Our motivation is illustrated in Fig. 1, showing a simple graph for a binary label propagation problem. Labeled vertices with positive class are marked with $(+)$ (white) and negative class with $(-)$ (black). Vertex **A** has low degree and a large measure of disagreement among its neighbours, and vertex **C** has high degree and is connecting many other vertices (in the middle of a cluster). Vertex **B** is member of the cluster, and is also connected to vertex **A**. Gray level indicates the assigned label value by the algorithms. Our intuition implies that the correct action will be to lower the effect of the confusing **A**, while propagating the negative label from right to left through **C** to **B** and the entire cluster. MAD lowers the effect of **C**, thus damaging the ability to identify all vertices in the

cluster. In contrast, our algorithm (named TACO below) correctly identifies the cluster while ignoring the unreliable vertex **A**.

To measure the level of agreement between neighbours, as well as account for vertex degree, our algorithm maintains per vertex confidence information in addition to label information. The confidence parameters capture the quality of the estimated label information. One consequence of that model is that the algorithm does not reduce the effect of *all* high degree vertices, but only the effect of vertices with low confidence in their estimated labels. Both confidence and label information are propagated throughout the graph.

After introducing our model we cast learning as a convex optimization problem and propose an efficient algorithm for solving it. We experiment with seven text categorization problems of various sizes and show that our algorithm, evaluated with several metrics, outperforms other algorithms (MAD and alternating minimization (AM) [10]) on most tasks.

2 Problem Formulation

We focus on transductive learning using graphs. The input is constituted of two sets: a set of n_l labeled examples $D_l = \{(\boldsymbol{x}_i, y_i)\}_{i=1}^{n_l}$ and a set of n_u unlabeled examples $D_u = \{\boldsymbol{x}_i\}_{i=n_l+1}^{n_l+n_u}$. Examples belong to some input space $\boldsymbol{x}_i \in \mathcal{X}$, where we assume in this paper that it is a vector space $\mathcal{X} = \mathbb{R}^d$. Labels belong to a finite label set $y_i \in L$ denoted by $L = \{1, \ldots, m\}$. The goal of the learning algorithm is to assign a label $\hat{y}_i \in L$ to each of the (unlabeled) examples in D_u. We denote the total number of examples by $n = n_l + n_u$.

In order to be able to propagate the labels from the labeled examples to the unlabeled ones, we assume the existence of an undirected weighted graph $G = (V, E, W)$ over all input examples. Each input example $\boldsymbol{x}_i$ is associated with a vertex $v_i \in V$. For ease of presentation we assume that the labeled input examples are associated with the first n_l vertices in V and we refer to them as *labeled vertices*. Similarly, the remaining n_u vertices in V are associated with the unlabeled input examples, and are called *unlabeled vertices*.

A weighted edge $e \in E = V \times V$ represents label-similarity. The larger the weight $w_{i,j} \in W$ of an edge between vertices v_i and v_j is, the more we believe the labels of $\boldsymbol{x}_i$ and $\boldsymbol{x}_j$ should be the same. We assume that the weight matrix $W \in \mathbb{R}^{n \times n}$ is symmetric with non-negative elements, and that there is an edge between any two vertices, although in practice most edges will have the lowest weight of zero.

We denote by $\delta_l(i) = \mathbf{1}_{[i \leq n_l]}$ the indicator of a vertex to be labeled, that is $\delta_l(i) = 1$ iff the vertex v_i is a labeled vertex. We associate a labels vector $\boldsymbol{y}_i \in \{0, 1\}^m$ with each vertex. For vertices associated with labeled examples v_i we set $y_{i,r} = 1$ iff the correct label of example $\boldsymbol{x}_i$ is $y_i = r$. All other entries are set to 0. For vertices associated with unlabeled examples v_j we set $\boldsymbol{y}_j = \mathbf{0}$, the vector with all elements equal to zero. We denote the complete prior information matrix by $\mathbf{Y} \in \mathbb{R}^{n \times m}$, where the i*th* row $\boldsymbol{y}_i$ corresponds to vertex v_i.

3 Algorithm

We cast learning as an optimization over two sets of parameters, each contains one element per vertex v_i in the graph. The first set of parameters the algorithm maintains is a per vertex score vector $\boldsymbol{\mu}_i = [\mu_{i,1}, \ldots, \mu_{i,m}]^\top \in \mathbb{R}^m$. The larger the r$th$ element $\mu_{i,r}$ is, the stronger is our belief that the input $\boldsymbol{x}_i$ associated with vertex v_i belongs to class r. The final predicted label is given according to the highest score, namely $\hat{y}_i = \arg\max_r \mu_{i,r}$.

In addition, the algorithm maintains a diagonal non-negative matrix per vertex, $\boldsymbol{\Sigma}_i \in \mathbb{R}^{m \times m}$, where we denote by $\sigma_{i,r}$ the rth diagonal element of $\boldsymbol{\Sigma}_i$. Each parameter $\sigma_{i,r}$ is associated with our uncertainty in the corresponding score parameter $\mu_{i,r}$. The lower the value of $\sigma_{i,r}$ is, the higher our confidence in the score value $\mu_{i,r}$.

Conceptually, the score vectors are our *first order* information over labels, while the uncertainty matrices are our *second order* information. Most, if not all, previous graph-based transduction algorithms maintain only first order information (scores) over vertices, ignoring agreement or disagreement between neighbour vertices. The second order information is designed to capture this exact information, allowing the algorithm to better label various vertices by considering agreement levels among their neighbours.

Next, we formulate an unconstrained convex optimization problem in the variables $\{(\boldsymbol{\mu}_i, \boldsymbol{\Sigma}_i)\}_{i=1}^n$ as our learning objective, and derive an efficient iterative algorithm for minimizing it.

3.1 Objective

We have three desired properties of the optimization problem used to define learning, the first two properties build on previous research [12,14], while the last is required for the usage of confidence.

First, a pair of close vertices v_i and v_j (i.e. with large $w_{i,j}$) should have close score vectors $\boldsymbol{\mu}_i$ and $\boldsymbol{\mu}_j$ if we are certain in both vectors, that is if both matrices $\boldsymbol{\Sigma}_i$ and $\boldsymbol{\Sigma}_j$ have low eigen-values. If at least one of these matrices has large eigen-values, then we are not confident in the score values of at least one of the corresponding vertices, and therefore relax the demand that the two score vectors $\boldsymbol{\mu}_i$ and $\boldsymbol{\mu}_j$ should be close.

Second, the score vectors of labeled vertices should be close to the true label vectors associated with them, again, if the corresponding uncertainty is low (or large certainty). In other words, if the eigenvalues of $\boldsymbol{\Sigma}_i$ are low, then $\boldsymbol{\mu}_i \approx \boldsymbol{y}_i$.

Third, the uncertainty should be far from infinity and not close to zero. As in both cases, the first two properties would be invalid. Specifically, we add a term that drives the uncertainty close to some predefined values, one can think of this term as a combination of both regularization (far from infinity) and barrier (strictly greater than zero) for uncertainty.

We formalize the above intuition using a symmetric Mahalanobis distance that is based on the uncertainty matrices. Specifically, given two pairs $(\boldsymbol{x}, \mathbf{S})$

and $(\boldsymbol{y}, \mathbf{T})$ of score vectors $\boldsymbol{x}, \boldsymbol{y} \in \mathbb{R}^m$ and uncertainty matrices $\mathbf{S}, \mathbf{T} \in \mathbb{R}^{m \times m}$ we define the squared-distance between them to be,

$$D_M(\boldsymbol{x}, \boldsymbol{y}, \mathbf{S}, \mathbf{T}) = (\boldsymbol{x} - \boldsymbol{y})^\top \left(\mathbf{S}^{-1} + \mathbf{T}^{-1}\right)(\boldsymbol{x} - \boldsymbol{y}) \ . \tag{1}$$

If either $\mathbf{S}$ or $\mathbf{T}$ have large eigenvalues, then the distance is low even if $\boldsymbol{x}$ and $\boldsymbol{y}$ are not close to each other. On the other hand, if both $\mathbf{S}$ and $\mathbf{T}$ have low eigenvalues, then in order for the distance to be low, we require that $\boldsymbol{x}$ and $\boldsymbol{y}$ would be close to each other.

Common [11,12,14] choice for formulating the first property is to require that neighbour vertices would have close scores, that is,

$$\sum_{i,j=1}^{n} w_{i,j} D(v_i, v_j) \tag{2}$$

where $D(v_i, v_j)$ is some distance function measuring the difference between scores for v_i and v_j.

The second desired property is that the scores for labeled vertices should be close to the input labels of these vertices. We formulate this requirement using the same conceptual ideas of the first property. We view each labeled vertex as two vertices, denoted by v_i and z_i. As before, the vertex v_i is associated with scores $\boldsymbol{\mu}_i$ and uncertainty $\boldsymbol{\Sigma}_i$. The new vertex z_i has *fixed* scores $\boldsymbol{y}_i$ and *fixed* uncertainty $\frac{1}{\gamma}\mathbf{I} \in \mathbb{R}^{m \times m}$ for some $\gamma > 0$. The new vertex z_i with fixed parameters is connected only to its counterpart v_i with an edge of weight 1. Similarly to (2) we add a second term to our objective capturing the second property,

$$\sum_{i=1}^{n_l} D(v_i, z_i) \ . \tag{3}$$

The third and last property is formally a regularization term forcing the uncertainty matrix to be close to some predefined matrix. We use the following convex function which is a sum of two terms, the first monotonic in the eigenvalues, and the second prevents matrices from having zero eigenvalues,

$$\sum_{i=1}^{n} \mathrm{Tr}\boldsymbol{\Sigma}_i - \eta \sum_{i=1}^{n} \log\det \boldsymbol{\Sigma}_i \ , \tag{4}$$

for some $\eta > 0$. Clearly, the minimizers of the last terms are the matrices $\boldsymbol{\Sigma}_i = \eta\mathbf{I}$.

Combining (2) (3) and (4) together with weights $\nu, \kappa, \alpha \geq 0$ and using our distance (1) we get the final objective,

$$
\begin{aligned}
C\left(G,\{\boldsymbol{\mu}_i\},\{\boldsymbol{\Sigma}_i\}\right) =& \frac{1}{4}\nu \sum_{i,j=1}^{n} w_{i,j}\left[\left(\boldsymbol{\mu}_i-\boldsymbol{\mu}_j\right)^\top\left(\boldsymbol{\Sigma}_i^{-1}+\boldsymbol{\Sigma}_j^{-1}\right)\left(\boldsymbol{\mu}_i-\boldsymbol{\mu}_j\right)\right] \\
&+ \frac{1}{2}\kappa\sum_{i=1}^{n_l}\left[\left(\boldsymbol{\mu}_i-\boldsymbol{y}_i\right)^\top\left(\boldsymbol{\Sigma}_i^{-1}+\frac{1}{\gamma}\mathbf{I}\right)\left(\boldsymbol{\mu}_i-\boldsymbol{y}_i\right)\right] \\
&+ \alpha\sum_{i=1}^{n}\mathrm{Tr}\boldsymbol{\Sigma}_i - \alpha\eta\sum_{i=1}^{n}\log\det\boldsymbol{\Sigma}_i \ .
\end{aligned}
$$

We can divide the last equation by one of the constants (e.g. ν), set in practice the other constant to be $\kappa = 1$ and denote by $\beta = \alpha\eta$, ending up with the following,

$$
\begin{aligned}
C\left(G,\{\boldsymbol{\mu}_i\},\{\boldsymbol{\Sigma}_i\}\right) =& \frac{1}{4}\sum_{i,j=1}^{n} w_{i,j}\left[\left(\boldsymbol{\mu}_i-\boldsymbol{\mu}_j\right)^\top\left(\boldsymbol{\Sigma}_i^{-1}+\boldsymbol{\Sigma}_j^{-1}\right)\left(\boldsymbol{\mu}_i-\boldsymbol{\mu}_j\right)\right] \\
&+ \frac{1}{2}\sum_{i=1}^{n_l}\left[\left(\boldsymbol{\mu}_i-\boldsymbol{y}_i\right)^\top\left(\boldsymbol{\Sigma}_i^{-1}+\frac{1}{\gamma}\mathbf{I}\right)\left(\boldsymbol{\mu}_i-\boldsymbol{y}_i\right)\right] \\
&+ \alpha\sum_{i=1}^{n}\mathrm{Tr}\boldsymbol{\Sigma}_i - \beta\sum_{i=1}^{n}\log\det\boldsymbol{\Sigma}_i \ . \qquad (5)
\end{aligned}
$$

We conclude this section by noting that the above objective is convex in all arguments. Thus, any algorithm for solving convex problems, like gradient-descent, can be used. In the next section we propose an efficient iterative algorithm for specifically solving (5). We then note the connections between the derived algorithm and our initial intuition.

3.2 An Iterative Algorithm

We now present an efficient algorithm for minimizing (5). The algorithm is iterative in nature. Let $\boldsymbol{\mu}_i^{(t)}$ and $\boldsymbol{\Sigma}_i^{(t)}$ denote the score vector and confidence matrix maintained by our algorithm at iteration t for vertex v_i. Roughly speaking, on each iteration t, the algorithm optimizes the objective over scores $\boldsymbol{\mu}_i$ given the score values of all other vertices $\boldsymbol{\mu}_j^{(t-1)}$ for $j \neq i$ and all uncertainty matrices $\boldsymbol{\Sigma}_j^{(t-1)}$ (for all j), and optimizes all the uncertainty matrices $\{\boldsymbol{\Sigma}_i\}$ given all the scores $\{\boldsymbol{\mu}_i^{(t-1)}\}$.

We first develop the update step for $\boldsymbol{\mu}_i^{(t)}$ and then follow with the update step for $\boldsymbol{\Sigma}_i^{(t)}$. Setting to zero the derivative of (5) with respect to $\boldsymbol{\mu}_i$ we get,

$$
\begin{aligned}
&\sum_{\substack{j=1\\ j\neq i}}^{n} w_{i,j}\left[\left(\boldsymbol{\Sigma}_i^{(t-1)}\right)^{-1}+\left(\boldsymbol{\Sigma}_j^{(t-1)}\right)^{-1}\right]\left(\boldsymbol{\mu}_i^{(t)}-\boldsymbol{\mu}_j^{(t-1)}\right) \\
&+ \delta_l(i)\left[\left(\boldsymbol{\Sigma}_i^{(t-1)}\right)^{-1}+\frac{1}{\gamma}\mathbf{I}\right]\left(\boldsymbol{\mu}_i^{(t)}-\boldsymbol{y}_i\right) = 0 \ . \qquad (6)
\end{aligned}
$$

For simplicity we introduce the following notation,

$$\mathbf{K}_{i,j}^{(t-1)} = w_{i,j}\left[\left(\boldsymbol{\Sigma}_i^{(t-1)}\right)^{-1} + \left(\boldsymbol{\Sigma}_j^{(t-1)}\right)^{-1}\right], \quad \mathbf{P}_i^{(t-1)} = \delta_l(i)\left[\left(\boldsymbol{\Sigma}_i^{(t-1)}\right)^{-1} + \frac{1}{\gamma}\mathbf{I}\right] \ . \tag{7}$$

Substituting (7) in (6) we get the following equation relating the optimal solution $\boldsymbol{\mu}_i^{(t)}$ to all other quantities, $\left(\sum_{\substack{j\neq i\\ j=1}}^{n}\mathbf{K}_{i,j}^{(t-1)}\right)\boldsymbol{\mu}_i^{(t)} + \mathbf{P}_i^{(t-1)}\boldsymbol{\mu}_i^{(t)} = \sum_{\substack{j\neq i\\ j=1}}^{n}\mathbf{K}_{i,j}^{(t-1)}\boldsymbol{\mu}_j^{(t-1)} + \mathbf{P}_i^{(t-1)}\boldsymbol{y}_i$. Solving for $\boldsymbol{\mu}_i^{(t)}$ we obtain,

$$\boldsymbol{\mu}_i^{(t)} = \left(\sum_{\substack{j\neq i\\ j=1}}^{n}\mathbf{K}_{i,j}^{(t-1)} + \mathbf{P}_i^{(t-1)}\right)^{-1}\left(\sum_{\substack{j\neq i\\ j=1}}^{n}\mathbf{K}_{i,j}^{(t-1)}\boldsymbol{\mu}_j^{(t-1)} + \mathbf{P}_i^{(t-1)}\boldsymbol{y}_i\right) \ . \tag{8}$$

Note that $\boldsymbol{\mu}_i^{(t)}$ is a matrix weighted average of all the other score vectors $\left\{\boldsymbol{\mu}_j^{(t-1)}\right\}_{j\neq i}$, and the true labels $\boldsymbol{y}_i$ (for a labeled vertex). The matrix-weights are exactly the inverse of the uncertainty matrices.

Next, we develop the update for the confidence matrices. Setting to zero the derivative of (5) with respect to $\boldsymbol{\Sigma}_i$ we get,

$$\begin{aligned}
&-\frac{1}{2}\sum_{j=1}^{n} w_{i,j}\left[\left(\boldsymbol{\Sigma}_i^{(t)}\right)^{-1}\left(\boldsymbol{\mu}_i^{(t-1)} - \boldsymbol{\mu}_j^{(t-1)}\right)\left(\boldsymbol{\mu}_i^{(t-1)} - \boldsymbol{\mu}_j^{(t-1)}\right)^{\top}\left(\boldsymbol{\Sigma}_i^{(t)}\right)^{-1}\right] \\
&-\frac{1}{2}\delta_l(i)\left[\left(\boldsymbol{\Sigma}_i^{(t)}\right)^{-1}\left(\boldsymbol{\mu}_i^{(t-1)} - \boldsymbol{y}_i\right)\left(\boldsymbol{\mu}_i^{(t-1)} - \boldsymbol{y}_i\right)^{\top}\left(\boldsymbol{\Sigma}_i^{(t)}\right)^{-1}\right] \\
&+\alpha\mathbf{I} - \beta\left(\boldsymbol{\Sigma}_i^{(t)}\right)^{-1} = 0 \ .
\end{aligned} \tag{9}$$

We define the following positive semi-definite matrix,

$$\begin{aligned}
\mathbf{R}_i^{(t-1)} =& \frac{1}{2}\sum_{j=1}^{n} w_{i,j}\left(\boldsymbol{\mu}_i^{(t-1)} - \boldsymbol{\mu}_j^{(t-1)}\right)\left(\boldsymbol{\mu}_i^{(t-1)} - \boldsymbol{\mu}_j^{(t-1)}\right)^{\top} \\
&+ \frac{1}{2}\delta_l(i)\left(\boldsymbol{\mu}_i^{(t-1)} - \boldsymbol{y}_i\right)\left(\boldsymbol{\mu}_i^{(t-1)} - \boldsymbol{y}_i\right)^{\top}
\end{aligned} \tag{10}$$

and rewrite (9) as $-\left(\boldsymbol{\Sigma}_i^{(t)}\right)^{-1}\mathbf{R}_i^{(t-1)}\left(\boldsymbol{\Sigma}_i^{(t)}\right)^{-1} + \alpha\mathbf{I} - \beta\left(\boldsymbol{\Sigma}_i^{(t)}\right)^{-1} = 0$. Multiplying both sides by $\boldsymbol{\Sigma}_i^{(t)}$ leads to a quadratic matrix equation in $\boldsymbol{\Sigma}_i^{(t)}$:

$$\alpha\left(\boldsymbol{\Sigma}_i^{(t)}\right)^2 - \beta\boldsymbol{\Sigma}_i^{(t)} - \mathbf{R}_i^{(t-1)} = 0 \ .$$

This is matrix quadratic equation with solution (as in scalars),

$$\boldsymbol{\Sigma}_i^{(t)} = \frac{\beta}{2\alpha}\mathbf{I} + \frac{1}{2\alpha}\left(\beta^2\mathbf{I} + 4\alpha\mathbf{R}_i^{(t-1)}\right)^{\frac{1}{2}} \ . \tag{11}$$

Both updates (8) and (11) are valid for full matrices $\mathbf{\Sigma}_i$. Diagonal matrices are obtained by diagonalizing the update (11) or specifically, taking the diagonal elements of the matrix $\mathbf{R}_i^{(t-1)}$ defined in (10). Thus, denoting the diagonal elements by $\mathbf{\Sigma}_i^{(t)} = diag\left(\sigma_{i,1}^{(t)}, \ldots, \sigma_{i,m}^{(t)}\right)$ and substituting into (8) we obtain the following update step for each score $\mu_{i,r}^{(t)}$:

$$\mu_{i,r}^{(t)} = \frac{\sum_{\substack{j=1 \\ j \neq i}}^{n} w_{i,j} \left(\frac{1}{\sigma_{i,r}^{(t-1)}} + \frac{1}{\sigma_{j,r}^{(t-1)}}\right) \mu_{j,r}^{(t-1)} + \delta_l(i) \left(\frac{1}{\sigma_{i,r}^{(t-1)}} + \frac{1}{\gamma}\right) y_{i,r}}{\sum_{\substack{j=1 \\ j \neq i}}^{n} w_{i,j} \left(\frac{1}{\sigma_{i,r}^{(t-1)}} + \frac{1}{\sigma_{j,r}^{(t-1)}}\right) + \delta_l(i) \left(\frac{1}{\sigma_{i,r}^{(t-1)}} + \frac{1}{\gamma}\right)} \tag{12}$$

Similarly, using (11) we obtain an update step for each $\sigma_{i,r}^{(t)}$:

$$\sigma_{i,r}^{(t)} = \frac{\beta}{2\alpha} + \frac{1}{2\alpha} \sqrt{\beta^2 + 2\alpha \left[\sum_{j=1}^{n} w_{i,j} \left(\mu_{i,r}^{(t-1)} - \mu_{j,r}^{(t-1)}\right)^2 + \delta_l(i) \left(\mu_{i,r}^{(t-1)} - y_{i,r}\right)^2\right]} \tag{13}$$

Note both updates (12) and (13) are separable in the labels and involve only variables with one specific label index r. In fact, if the confidence matrices are forced to be diagonal, the objective (5) can be decomposed into m separate optimization problems in independent parameters. In the experiments below we evaluated both the diagonal and full versions of the algorithm (updates (8) and (11)) and found no advantage for full confidence matrices over diagonal ones. We thus restrict ourself to diagonal matrices.

We call our algorithm *TACO* for *Transduction Algorithm with COnfidence*, and its pseudocode is summarized in Figure 2. The algorithm iterates over vertices, updating both parameters for each vertex, until some convergence criteria is met. In practice, we ran the algorithm for not more than 10 iterations.

As mentioned above for full matrices, the update of (12) sets $\mu_{i,r}^{(t)}$ to be a weighted average of neighbouring scores for each label r (and the true label if given), where each weight is a product of the edge-weight and a correction factor that depends on estimated confidence parameters. The more certain we are in a neighbour, the higher relative weight it will have.

Looking at the update step for the uncertainty matrices (either full in (11) or diagonal (13)) we note that $\sigma_{i,r}^{(t)}$ is monotonic on a quadratic measure of disagreement between neighbours, $\sum_{j=1}^{n} w_{i,j} \left(\mu_{i,r}^{(t-1)} - \mu_{j,r}^{(t-1)}\right)^2 + \delta_l(i) \left(\mu_{i,r}^{(t-1)} - y_{i,r}\right)^2$. In addition, this measure is *not* normalized using the degree of v_i, implying that for high degree vertices this measure is more likely to have a high-value, lowering the influence of the high degree vertex on its neighbours. However, this happens only when combined together with neighbours disagreement. If the score of all neighbour nodes is about the same, even if the number of them is very large, the certainty would be large (or uncertainty low).

Parameters: $\alpha > 0$, $\beta > 0$,$\gamma > 0$
Input: A graph $G = (V, E, W)$ and prior labeling $\boldsymbol{y}_i$ for all $v_i \in V$
Initialize: $t = 1$, $\boldsymbol{\mu}_i^{(0)} = \mathbf{0}$ and $\boldsymbol{\Sigma}_i^{(0)} = \mathbf{I}$ for all $v_i \in V$
Repeat

– For $v_i \in V$:
  • Compute $\boldsymbol{\mu}_i^{(t)}$ from $\boldsymbol{\mu}_j^{(t-1)}$ and $\boldsymbol{\Sigma}_j^{(t-1)}$:

$$\mu_{i,r}^{(t)} = \frac{\sum_{\substack{j=1 \\ j \neq i}}^{n} w_{i,j} \left(\frac{1}{\sigma_{i,r}^{(t-1)}} + \frac{1}{\sigma_{j,r}^{(t-1)}} \right) \mu_{j,r}^{(t-1)} + \delta_l(i) \left(\frac{1}{\sigma_{i,r}^{(t-1)}} + \frac{1}{\gamma} \right) y_{i,r}}{\sum_{\substack{j=1 \\ j \neq i}}^{n} w_{i,j} \left(\frac{1}{\sigma_{i,r}^{(t-1)}} + \frac{1}{\sigma_{j,r}^{(t-1)}} \right) + \delta_l(i) \left(\frac{1}{\sigma_{i,r}^{(t-1)}} + \frac{1}{\gamma} \right)} \tag{12}$$

  • Compute $\boldsymbol{\Sigma}_i^{(t)}$ from $\boldsymbol{\mu}_j^{(t-1)}$:

$$\sigma_{i,r}^{(t)} = \frac{\beta}{2\alpha} + \frac{1}{2\alpha} \sqrt{\beta^2 + 2\alpha \left[\sum_{j=1}^{n} w_{i,j} \left(\mu_{i,r}^{(t-1)} - \mu_{j,r}^{(t-1)} \right)^2 + \delta_l(i) \left(\mu_{i,r}^{(t-1)} - y_{i,r} \right)^2 \right]} \tag{13}$$

– $t \leftarrow t + 1$

Until convergence
Output: Score vectors $\boldsymbol{\mu}_i^{(t)}$ and confidence matrices $\boldsymbol{\Sigma}_i^{(t)}$.

Fig. 2. The TACO algorithm for graph-based transduction.

4 Empirical Evaluation

We evaluate our algorithm along with two other state-of-the-art graph-based transduction algorithms: Alternating Minimization (AM) [11] and Modified Adsorption (MAD) [12]. Both were empirically shown to outperform LP [12,11].

4.1 Data Sets

We evaluate our algorithm using seven NLP data sets summarized in Table 1.
WebKB: World Wide Knowledge Base is a text categorization data set previously used for the evaluation of several transductive algorithms [7,8,11,12]. Documents in the data set are web pages from four academic web domains, categorized according to domain and topic. We used documents from four topic categories *course*, *faculty*, *project* and *student*, for a total of 4,199 documents. We construct the graph by first removing all non-textual information (HTML tags), followed by lower casing all tokens, computing TFIDF features, and lastly using cosine similarity to form edge weights. This graph yields better results for all evaluated algorithms in comparison to the graph used previously [11,12] where both textual and non-textual information was used.

20 Newsgroups: The dataset contains a collection of 18,828 newsgroup documents partitioned across 20 different newsgroups[1]. We select one-quarter of the documents from each category, yielding a total of 4,715 documents. Graph construction is as described for WebKB except for HTML removal.
Sentiment: Sentiment classification of book reviews from Amazon. Each review is labeled with stars corresponding to the opinion of the author. The range of the stars (labels) is from one (low rating) to five (high rating). We used 5,000 reviews. Graph construction follows TFIDF features along with cosine similarity.

Data set	Instances	Labels	Test set size
WebKB	4,199	4	3,148
20 News	4,175	20	3,534
Sentiment	5,000	5	3,750
Reuters	4,000	4	3,000
Enron A	3,019	10	2,262
Enron B	3,171	10	2,376
Amazon3	7,000	3	5,250

Table 1. A summary of data sets used in empirical evaluation.

In addition to the aforementioned data sets we use the following data of Crammer et al [4]. This data contains preprocessed feature vectors generated for several NLP data sets. Further details concerning feature extraction per data set can be found in [4]. For each data set, we computed TFIDF features from the given counts, and constructed the graph using cosine similarity.
Reuters: Topic classification of newswire stories (RCV1-v2) [9]. We used 4,000 instances, categorized by the following general topics: *corporate*, *economic*, *government* and *markets*. **Enron:** A collection of e-mails from over 100 different users organized in folders[2]. The task is automatic classification of e-mails into folders. E-mails from two users are used: farmer-d (Enron A) and kaminski-v (Enron B). Enron A contains 3,019 instances and Enron B has 3,171 instances. For each user there are 10 labels - email folders. **Amazon3:** Product reviews from Amazon. The task is to classify reviews to one of the 3 product domains: *books*, *dvds* and *music*. 7,000 reviews are used.

4.2 Experimental Setup

We follow previous experimental setup [11,12]. From the initial input graph we construct a K Nearest Neighbour (K-NN) graph, by keeping for every vertex only its K closest neighbours. The result of this preprocessing step is a directed graph. We then removed the direction of edges, ending up with an undirected graph, where degrees of edges may be larger than K. Next, we assign labels to randomly sampled n_l documents, under the constraint that each class is represented by at least one sample in the labeled set. Finally, we sample three-quarters of the remaining documents to constitute the test set. This process gives a single transduction set, and we repeat it 21 times.

[1] `http://people.csail.mit.edu/jrennie/20Newsgroups/`
[2] `http://www.cs.cmu.edu/~enron/`

We use the first transduction set for hyper-parameter tuning, and the others for evaluation. The hyper-parameters search grid for each of the evaluated algorithms is as follows: for AM α, μ, and ν as described in [11], for MAD μ_1, μ_2 and μ_3 as described in [12] (Sec. 6.1) and for TACO $\alpha, \beta \in \{\text{1e-8}, \text{1e-4}, \text{1e-2}, 1, 10, 100\}$ and $\gamma \in \{1, 2, 5\}$. We search over same range of K for all algorithms: $K \in \{100, 500, 1000, 2000\}$. We found that setting the maximum number of iterations per single run to 10, for all evaluated algorithms, was sufficient for convergence.

We perform class prior normalization (CPN) by column normalizing the prior information matrix $\mathbf{Y}$ for MAD and TACO. This ensures the total initial input score is the same for all classes and reduces the advantage common classes have on rare classes during label propagation. Since the rows of $\mathbf{Y}$ for AM are probability distribution, it is not possible to perform CPN prior to running AM, and it was not used when experimenting with AM before [11].

4.3 Evaluation Metrics

We evaluate the results of our experiments using five evaluation metrics: macro-averaged Precision-Recall Break Even Point (PRBEP), accuracy (ACC), macro-averaged accuracy (M-ACC), Mean Reciprocal Rank (MRR) and macro-averaged Mean Reciprocal Rank (M-MRR).

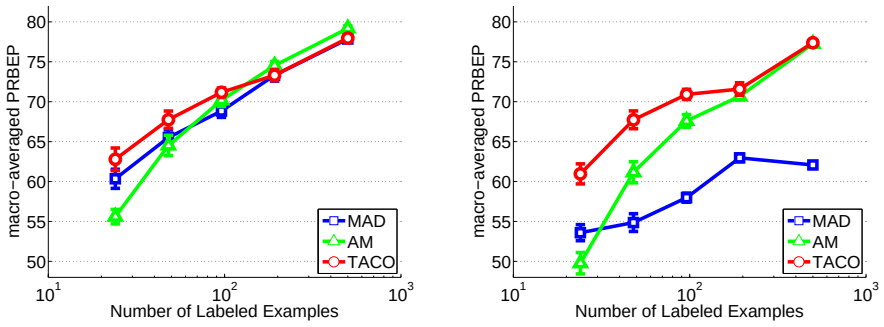
PRBEP is defined as the point in which precision and recall are equal. For each label $r \in L$ we compute PRBEP as follows. Let $\boldsymbol{s}_r = [\mu_{1,r}, \ldots, \mu_{n,r}]$ denote a vector containing the score assigned for all vertices and the label r. We define a threshold τ and use it for prediction: each vertex v_i with score $\mu_{i,r} > \tau$ is considered as being in class r. We move τ in steps through the complete range of values in $\boldsymbol{s}_r$ in descending order. For each step we update our prediction and calculate precision and recall. PRBEP is reported when the values are equal. Finally, we macro-average over all possible labels.

While PRBEP has been used as the evaluation metric in previous work [11,12], it does not capture the performance of the common multi-class inference rule $\hat{y}_i = \arg\max_r \mu_{i,r}$ used also by TACO. We thus also report accuracy. However, graph-based transduction algorithms tend to create degenerate solutions, such as solutions for which all unlabeled vertices are classified as being in the most common single class. Therefore, we also report macro-averaged accuracy: we divide the complete test set into disjoint sets according to true labels, compute accuracy on each individual set and average. This increases the effect of prediction accuracy in rare classes and demotes degenerate solutions.

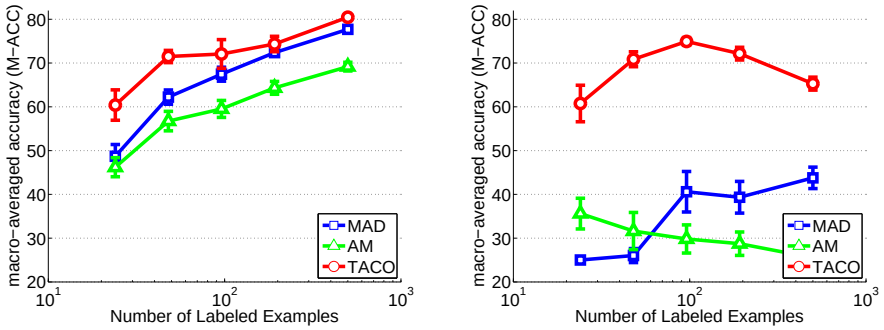
In addition to PRBEP and accuracy, we report Mean Reciprocal Rank (MRR), $\text{MRR} = (1/|Q|) \sum_{v_i \in Q} (1/r_i)$ where Q is the test set and r_i is the rank of the true label y_i within the score vector $\boldsymbol{\mu}_i$. This metric has been recently used for comparing several graph-based transductive algorithms on the task of acquiring class-instance pairs from text [13]. While accuracy is based on prediction alone, this metric favours both accurate prediction as well as a good rank for the true label in case of a mistake. As before, in addition to MRR we report also macro-averaged MRR, discouraging degenerate solutions.

		Optimized by PRBEP			Optimized by M-ACC		
		MAD	AM	TACO	MAD	AM	TACO
WebKB 48 labeled	PRBEP	65.5	64.5	**67.7**	54.9	61.2	**67.7**
	ACC	22.0	43.8	**72.5**	59.5	63.5	**72.5**
	M-ACC	26.0	31.6	**70.9**	62.2	56.7	**71.5**
	MRR	49.8	66.1	**84.5**	76.5	78.7	**84.6**
	M-MRR	52.7	57.1	**83.4**	78.0	73.9	**83.9**
20 News 105 labeled	PRBEP	50.7	49.8	**57.0**	49.3	47.7	**55.6**
	ACC	16.6	40.3	**58.8**	50.0	45.8	**61.0**
	M-ACC	16.6	39.9	**57.9**	49.6	45.3	**60.2**
	MRR	34.9	54.5	**72.9**	65.3	59.8	**74.8**
	M-MRR	35.1	54.2	**72.4**	65.1	59.5	**74.4**
Sentiment 500 labeled	PRBEP	34.4	**34.9**	34.5	27.8	31.0	**33.7**
	ACC	24.3	**38.6**	25.4	32.6	**41.7**	38.3
	M-ACC	26.6	20.0	**27.4**	**32.5**	29.3	32.2
	MRR	50.7	**59.9**	51.6	56.9	**61.9**	60.7
	M-MRR	52.4	45.8	**53.1**	**56.6**	52.9	55.7
Reuters 48 labeled	PRBEP	**73.2**	**73.2**	73.0	72.0	67.6	**74.9**
	ACC	60.0	69.4	**82.7**	76.3	75.8	**81.6**
	M-ACC	59.1	55.6	**73.0**	75.6	67.2	**76.2**
	MRR	76.7	82.7	**90.0**	86.6	86.3	**89.7**
	M-MRR	76.2	72.8	**83.3**	**85.9**	80.1	**85.9**
Enron A 48 labeled	PRBEP	54.3	**54.8**	54.2	46.7	50.5	**50.9**
	ACC	45.4	**60.0**	46.0	49.5	**56.6**	55.6
	M-ACC	51.3	41.1	**52.0**	**52.4**	49.0	50.0
	MRR	62.8	**74.2**	63.3	66.2	70.2	**71.6**
	M-MRR	66.1	59.4	**66.7**	**67.0**	64.1	66.3
Enron B 48 labeled	PRBEP	39.7	36.3	**41.7**	36.7	35.7	**41.5**
	ACC	20.4	26.4	**41.4**	36.5	35.8	**41.1**
	M-ACC	21.0	16.6	**38.5**	**38.7**	29.9	38.4
	MRR	40.8	46.0	**59.1**	54.9	54.2	**58.7**
	M-MRR	39.3	36.4	**56.6**	55.5	48.4	**56.6**
Amazon3 35 labeled	PRBEP	**89.4**	77.7	88.4	75.4	74.1	**87.8**
	ACC	34.7	43.9	**88.5**	73.0	68.9	**88.9**
	M-ACC	34.8	43.9	**88.5**	73.0	68.9	**88.9**
	MRR	62.7	67.4	**93.9**	85.2	83.0	**94.2**
	M-MRR	62.8	67.4	**93.9**	85.2	83.0	**94.2**

Table 2. A comparison of empirical results for three algorithms on all data sets. Each reported result is an average over 20 randomly generated transduction sets. Left block results were obtained by tuning hyper-parameters using macro-averaged PRBEP, while results on the right block were tuned using macro-averaged accuracy (M-ACC).



(a) PRBEP tuned according to PRBEP (b) PRBEP tuned according to M-ACC



(c) M-ACC tuned according to M-ACC (d) M-ACC tuned according to PRBEP

Fig. 3. A comparison of macro-averaged PRBEP and macro-averaged accuracy (M-ACC) for different amounts of labeled data on the WebKB data set. All results are averages over 20 randomly generated transduction sets. In figures (a) and (d) hyper-parameters were tuned according to PRBEP. In figures (b) and (c) tuning is according to M-ACC. Error bars indicate 95% confidence intervals.

4.4 Results

Results for seven NLP tasks are summarized in Table 2. All results are averages over 20 randomly generated transduction sets as described in Sec. 4.2. We performed hyper-parameters tuning according to PRBEP since it is the reported metric in previous work [7,8,11,12]. In addition, we tune by M-ACC since it better relates to our inference rule and also penalizes degenerate solutions.

Results reported in the left part of Table 2 are from experiments where tuning was performed according to PRBEP. Focusing on the tuned metric, TACO outperforms the other two algorithms on three of the seven data sets by at least 2 points and is worse by at most 1 point. However, TACO outperforms the other algorithms on all datasets when evaluated using either M-ACC or M-MRR. This implies solutions generated by TACO are preferable.

Results reported in the right part of Table 2 were obtained by tuning with macro-averaged accuracy (M-ACC). Here, considering the tuned metric, TACO

is best on four of the seven data sets, three of which by at least 9 points. When TACO is not the best performing algorithm, the second best algorithm has M-ACC not larger than 1 point, except in one case where the difference is about 2.5 points. As before, considering other metrics, TACO is best on at least four data sets for all other metrics.

Comparing both parts of Table 2 suggests TACO is robust to the choice of the metric used for tuning. For example, on WebKB, tuning according to PRBEP or M-ACC does not change much the reported results. AM and MAD are more sensitive to the specific choice of a metric for tuning. Considering again WebKB, there are significant differences between results from the left and right parts of Table 2. In general, for most data sets, the difference between results in experiments tuned according to PRBEP or M-ACC is substantially smaller for TACO compared to both MAD and AM.

Overall performance is high in absolute values on WebKB, 20 Newsgroups, Reuters and Amazon3. On these data sets, TACO is best by almost any used evaluation metric. This suggests our algorithm gains from the merits of a good input graph better than others. Finally, comparing ACC vs M-ACC, and MRR vs M-MRR, we note that using averaged metrics the results are in general higher than using their macro-average counterparts. Yet, for TACO this difference is smaller, which indicates that TACO predicts better the less frequent classes, while not harming performance on the more frequent classes (see also Table 3).

class	size	MAD	AM	TACO
course	930	**85.4**	81.1	**85.4**
faculty	1,124	58.9	59.9	**60.8**
project	504	41.4	42.6	**46.2**
student	1,641	76.1	74.4	**78.4**
average	-	65.5	64.5	**67.7**

Table 3. Precision-Recall Break Even Point (PRBEP) results per class on WebKB data set, with 48 labeled examples. All results are averages over 20 randomly generated transduction sets.

A comparison of PRBEP and M-ACC vs amount of labeled examples is given in Fig. 3. In Fig. 3(a) and Fig. 3(c) the evaluation metric used for tuning is also the reported metric. In these two plots, we observe that performance increases as more labeled examples are given as input, as expected. TACO outperforms other algorithms, especially when the amount of labeled examples is small. Fig. 3(d) and Fig. 3(b) show how tuning by one metric affects the performance of the other metric. After tuning using PRBEP, TACO achieves far better M-ACC score than other algorithms, and vice-versa, although with smaller gap, tuning with M-ACC, TACO achieves better PRBEP score than others algorithms. Additionally, as more labeled examples are used, performance in the not-tuned metric may not grow. This implies tuning by PRBEP overfits the algorithm output towards solutions specifically maximizing the PRBEP measure, and results evaluated with other evaluation metrics such as M-ACC, are decreased. This phenomenon is most visible considering AM. As illustrated by Fig. 3(b), as the number of input la-

beled examples grows, performance in the not-tuned metric, namely PRBEP, increases. This suggests tuning by M-ACC is better for the WebKB data set.

Finally, per-class PRBEP for the WebKB dataset is summarized in Table 3. The average PRBEP (bottom line) corresponds to the top-left line in Table 2. TACO achieves the best performance in all four labels, and there is no clear winner between the two other algorithms. The highest improvement is for the *project* category, the one with the smallest number of test samples, (from 42.6 to 46.2), for which all algorithms obtain lowest performance among the four categories. It seems that the tuning via confidence information allows TACO to automatically adapt to unbalanced data.

5 Related Work

Our work builds on previous graph-based transduction learning, where the graph is built using nearest-neighbours based on some distance. Label propagation (LP) [14] and Modified Adsorption (MAD) [12] use squared Euclidean distance, while alternating minimization (AM) [10,11] uses the Kullback-Leibler divergence. To the best of our knowledge, our work is the first to apply second order information into transductive learning, implemented using the Mahalanobis distance with parameters (matrix) being optimized as well.

One of the first and best performing graph-based transduction algorithm based on ℓ_2 norm is label propagation (LP) [14], with update rule,

$$\mu_{i,r}^{(t)} = \delta_l(i) y_{i,r} + (1 - \delta_l(i)) \frac{\sum_{j=1}^{n} w_{i,j} \mu_{j,r}^{(t-1)}}{\sum_{j=1}^{n} w_{i,j}} \ .$$

Setting our confidence parameters $\sigma_{i,r}$ to 2, our update (12) becomes

$$\mu_{i,r}^{(t)} = \frac{\sum_{j=1}^{n} w_{i,j} \mu_{j,r}^{(t-1)} + C \cdot \delta_l(i) y_{i,r}}{\sum_{j=1}^{n} w_{i,j} + C \cdot \delta_l(i)} \tag{14}$$

where we set $C = 1/2 + 1/\gamma$ and assume $w_{i,i} = 0$. The update step in (14) is a close variant of LP, allowing label information for labeled vertices to differ from the given known input labels. A very close version appears in a recent book [3] (Section 11.2, Algorithm 11.2). Thus, we can view this variant of LP as a special case of our algorithm with constant confidence parameters.

The idea of discouraging high degree vertices in label propagation first appears in Adsorption [1] and is later used in MAD [12]. Both algorithms use a static measure that considers only vertex degree to limit the effect of vertices with a large degree. TACO performs similar tuning, but automatically or dynamically based on the confidence parameters, which measure both degree and agreement level among neighbours.

Measures of confidence in estimated parameters have been successfully used in a number of non-SSL settings such as online learning [2,6] and multi-armed bandits with partial feedback [5].

6 Conclusion

We introduced the notion of confidence in label assignments to graph-based transduction. We formulated learning as an unconstrained convex optimization problem in both confidence and label parameters, and derived an efficient iterative algorithm for solving it. Our algorithm uses its confidence parameters to dynamically control the influence each vertex has on its neighbours during label propagation. Empirical evaluations on seven NLP tasks demonstrate that TACO improves over current state-of-the-art graph-based transductive algorithms, and is more robust to parameter tuning.

Currently, we plan to analyze TACO formally, experiment with more large-scale data and extend our work to multi-label, complex and structured problems. We also plan to generalize TACO to be able to generate models that will allow labeling unseen inputs beyond the transduction setting.

References

1. Shumeet Baluja, Rohan Seth, D. Sivakumar, Yushi Jing, Jay Yagnik, Shankar Kumar, Deepak Ravich, and Ran Mohamed Aly. Video suggestion and discovery for youtube: taking random walks through the view graph. In *WWW*, 2008.
2. Nicolo Cesa-Bianchi, Alex Conconi, and Claudio Gentile. A Second-Order Perceptron Algorithm. *SIAM Journal on Computing*, 34(3):640, 2005.
3. O. Chapelle, B. Schölkopf, and A. Zien, editors. *Semi-Supervised Learning*. MIT Press, Cambridge, MA, 2006.
4. Koby Crammer, Mark Dredze, and Alex Kulesza. Multi-class confidence weighted algorithms. In *EMNLP*, 2009.
5. Koby Crammer and Claudio Gentile. Multiclass classification with bandit feedback using adaptive regularization. In *ICML*, pages 273–280, 2011.
6. Mark Dredze and Koby Crammer. Confidence-weighted linear classification. In *ICML*, pages 264–271, 2008.
7. Thorsten Joachims. Transductive inference for text classification using support vector machines. In *ICML*, pages 200–209, 1999.
8. Thorsten Joachims. Transductive learning via spectral graph partitioning. In *ICML*, pages 290–297, 2003.
9. David D. Lewis, Yiming Yang, Tony G. Rose, Fan Li, and Thomas G. Dietterich. Rcv1: A new benchmark collection for text categorization research. *JMLR*, 2004.
10. Amarnag Subramanya and Jeff Bilmes. Soft-supervised learning for text classification. In *EMNLP*, 2008.
11. Amarnag Subramanya and Jeff Bilmes. Semi-Supervised Learning with Measure Propagation. *The Journal of Machine Learning Research*, 12:3311–3370, 2011.
12. Partha P. Talukdar and Koby Crammer. New regularized algorithms for transductive learning. In *ECML-PKDD*, 2009.
13. Partha P. Talukdar and Fernando Pereira. Experiments in graph-based semi-supervised learning methods for class-instance acquisition. In *ACL*, 2010.
14. Xiaojin Zhu, Zoubin Ghahramani, and John Lafferty. Semi-supervised learning using gaussian fields and harmonic functions. In *ICML*, 2003.