1 Semi-Supervised Learning with Conditional Harmonic Mixing

Christopher J. C. Burges
John C. Platt

Recently graph-based algorithms, in which nodes represent data points and links encode similarities, have become popular for semi-supervised learning. In this chapter we introduce a general probabilistic formulation called 'Conditional Harmonic Mixing', in which the links are directed, a conditional probability matrix is associated with each link, and where the numbers of classes can vary from node to node. The posterior class probability at each node is updated by minimizing the KL divergence between its distribution and that predicted by its neighbours. We show that for arbitrary graphs, as long as each unlabeled point is reachable from at least one training point, a solution always exists, is unique, and can be found by solving a sparse linear system iteratively. This result holds even if the graph contains loops, or if the conditional probability matrices are not consistent. We show how, given a classifier for a task, CHM can learn its transition probabilities. Using the Reuters database, we show that CHM improves the accuracy of the best available classifier, for small training set sizes.

1.1 Introduction

Graphical[1], models provide a powerful framework for approaching machine learning problems. Two common examples are probabilistic graphical models Jordan [1999] and semi-supervised learning on graphs (see Zhu and Ghahramani [2002], Zhu et al. [2003a], Zhou et al. [2003]) and which we refer to here as Laplacian SSL. Graphs have been used as a general representation of preference relations in ranking

1. To Appear in *Semi-Supervised Learning*, MIT Press, 2006, Eds. O. Chapelle, B. Schölkopf, A. Zien

problems (Dekel et al. [2004]) and play a role in various approaches to dimensional reduction (Burges [2005]). In this paper, we propose a new graph-based approach to semi-supervised learning called Conditional Harmonic Mixing (CHM).

Conditional Harmonic Mixing

Probabilistic graphical models such as Bayes nets write a probability distribution as a product of conditionals, which live on the nodes; the arcs encode conditional independence assumptions. Laplacian SSL is more closely related to random walks on networks (Doyle and Snell [1984]): each arc encodes the similarity between the nodes at its endpoints, and the goal is to use neighborhood structure to guide the choice of classification (or regression, clustering or ranking) function. For Laplacian SSL models, the probabilistic interpretation is somewhat indirect (Zhu et al. [2003a]); for probabilistic graphical models, it is central. In CHM we propose an intermediate model, where both a probabilistic interpretation and Laplacian SSL are central. In CHM, no attempt is made to model an overall joint density; we are only interested, ultimately, in the class conditional posteriors. Although it is assumed that there exists some underlying joint, the model itself is viewed as an approximation; in particular, the conditional probabilities may be approximations, and as a result, inconsistent (i.e. no joint may exist for which they are the conditionals). This results in some striking differences between CHM and well known probabilistic graphical models such as Bayes nets: for example, in CHM, the process of learning the posteriors, given the conditionals, converges to a global optimum via a straightforward iterative optimization procedure, whatever the structure of the graph (in particular, even if there are loops), provided only that there exists a path to each unlabeled point from at least one labeled point. In this regard, CHM is similar to dependency networks (Heckerman et al. [2001]). In CHM, as in Laplacian SSL, each node corresponds to a random variable, but unlike the original Laplacian SSL, the arcs are directed , and each arc carries a matrix which models a conditional probability[2]. The matrices can even be rectangular, which corresponds to the posteriors at different nodes corresponding to different numbers of classes for that random variable. We will also investigate learning the conditional probability matrices themselves from data. In this paper we will consider CHM models for classification, but the same ideas could be extended for regression, clustering, ranking, etc.

Directed Graphs

1.1.1 Conditional Harmonic Mixing: Motivation

CHM is a highly redundant model, in that for a 'perfect' CHM model of a given problem, the posterior for a given node can be computed from the posterior at any adjacent node, together with the conditional probability matrix on the arc joining them. However this is an idealization: CHM handles this by asking that the posterior at a given node be that distribution such that the number of bits needed to describe the distributions predicted at that node, by the adjacent nodes, is minimized. This

2. Recently, Zhou et al. [2005b] have extended Laplacian SSL to the case of directed arcs.

Kullback Leibler Divergence

is accomplished by minimizing a KL divergence (see below). Building on an idea proposed in Zhu et al. [2003a], CHM can also be used to improve the accuracy of another, given base classifier.

In the graphical approaches to semi supervised learning of Zhu and Ghahramani [2002], Zhu et al. [2003a], Zhou et al. [2003], the underlying intuition is that the function should vary smoothly across the graph, so that closely clustered points tend to be assigned similar function values (the "clustering assumption"). This leads to the use of undirected arcs in the graph, since the graph is used essentially to model the density. However there is a second intuition that we wish to add, and that is of the propagation of information. Consider the graph shown in Figure 1, left panel (in this chapter, filled (unfilled) circles represent labeled (unlabeled) points), where both arcs have the same weight. In the harmonic solutions of Zhu and Ghahramani [2002], Zhu et al. [2003a], Zhou et al. [2003], the state of node 2 is the weighted average of its neighbours. However in this particular graph, it seems strange to have node 2 care about the state of node 3, since from the information propagation point of view, all of the label information propagates out from node 1, and all label information about node 3 has already passed through node 2. By making the arcs directed, as in the right panel, this problem can be addressed with no loss of generality, since an undirected arc between nodes A and B can be simulated by adding arcs A→B and B→A.

Figure 1.1 Directional arcs for information flow.

A second reason for using directed arcs is that the relations between points can themselves be asymmetric (even if both are unlabeled). For example, in KNN, if point A is the nearest neighbour of point B, point B need not be that of point A. Such asymmetrical relations can be captured with directed arcs.

Harmonic Solutions

CHM shares with Laplacian SSL the desirable property that its solutions are harmonic, and unique, and can be computed iteratively and efficiently. It shares with Bayesian graphical models the desirable property that it is a probabilistic model from the ground up. We end this Section with a simple but useful observation, but first we must introduce some notation. Suppose nodes i and j ($i, j \in \{1, \ldots, N\}$) are connected by a directed arc from i to j (throughout, we will index nodes by i, j, k and vector indices by a, b). We will represent the posterior at any node k as the vector $P(X_k = M_a) \equiv P_k$ (so that P_k is a vector indexed by a), and the conditional on the arc as $P(X_j|X_i, G) \equiv P_{ji}$ (so that P_{ji} is a matrix indexed by the class index at node j and the class index at node i). Then the computation of i's prediction of the posterior at j is just the matrix vector multiply $P_{ji}P_i \doteq \sum_b (P_{ji})_{ab}(P_i)_b$. Note that all conditional matrices are also conditioned on the training data, the graph

structure, and other factors, which we denote collectively by G. We emphasize that the number of classes at different nodes can differ, in which case the conditional matrices joining them will be rectangular. Note also that the P_{ji} are column stochastic matrices. Similarly we will call any vector whose components are a nonnegative partition of unity a stochastic vector. Then we have the following

Observation: Given any two stochastic vectors P_i and P_j, there always exists a conditional probability matrix P_{ij} such that $P_i = P_{ij}P_j$.

This follows trivially from the choice $(P_{ij})_{ab} = (P_i)_a \quad \forall b$, and just corresponds to the special case that the probability vectors P_i and P_j are independent. This shows that CHM is able, in principle, to model any set of posteriors on the nodes, and that some form of regularization will therefore be needed if we expect, for example, to learn nontrivial matrices P_{ij} given a set of posteriors P_i. We will impose this regularization by partitioning the N_a arcs in the graph into a small number n of equivalence classes, where $n \ll N_a$, such that arcs in a given equivalence class are to have the same P_{ij}. In this paper, we will use nearest neighbour relations to determine the equivalence classes.

1.1.2 Related Work

Transductive Learning

Zhu and Ghahramani [2002], Zhu et al. [2003a], Zhou et al. [2003] introduce Laplacian SSL for transductive learning using graphs. Each link is given a scalar weight that measures the similarity between the data points attached to the nodes at that link's endpoints, and each node has a scalar value. The objective function is a weighted sum of squared differences in function values between pairs of nodes, with positive weights; minimizing this encourages the modeled function to vary slowly across nodes. The solution is a harmonic function (Doyle and Snell [1984]) in which each function value is the weighted sum of neighbouring values. The function is thresholded to make the classification decision. In contrast to Laplacian SSL, at the solution, for non-diagonal conditional probability matrices, the CHM conditional harmonic property generates extra additive terms in the function on the nodes, which are not present in the Gaussian field solution (which is homogeneous in the function values). The two methods coincide only when the random variables at all nodes correspond to just two classes, when the conditional probability matrices in CHM are two by two unit matrices, where all the weights in the Gaussian random field are unity, and where all nodes are joined by directed arcs in both directions. Finally, one concrete practical difference is that CHM can handle one-sided classification problems, where training data from only one class is available, by using conditional posterior matrices other than unit matrices. In this case, the objective function in Zhu et al. [2003a] is minimized by attaching the same label to all the unlabeled data. However either method can handle one-sided classification problems by leveraging results from an existing one-sided classifier; we will explore this method below.

Harmonic Function

Zhu et al. [2003b] embed the label propagation work in a probabilistic framework

Gaussian Process

by showing that the model can be viewed in terms of Gaussian processes. However to establish the connection, extra assumptions and approximations are required: the inverse covariance matrix must be regularized, an extra set of unobserved random variables, which give rise to the labels via a sigmoid conditional, are introduced, and the posteriors must be approximated (the authors use the Laplace approximation). CHM, by contrast, is inherently a probabilistic model.

Directed graph models for semi-supervised learning were also considered in Zhou et al. [2005b]. However in that work, the kinds of graphs considered were specific to a web-like application, with nodes split into hubs and authorities, and with the fundamental assumption that the similarity of two nodes in the graph is defined by their co-linkage (either from parents or children). Again, the aim of the model is to require that the modeled function vary slowly across 'similar' nodes, so the notion of information propagation described above does not play a direct role; the model is also not a probabilistic one. More recently, semi-supervised learning on directed graphs was also studied from a more general point of view in Zhou et al. [2005a].

Bayes Nets

Finally we emphasize the main differences between CHM and probabilistic graphical models such as Bayes nets and Markov random fields. Belief nets and MRF's use the graph structure to encode conditional independence assumptions about the random variables in the model, whereas CHM uses the (redundant) graph structure to model both the flow of information from the training data, and the smoothness assumptions on the functions being modeled. Evaluating belief nets (for example, using belief propagation) in the presence of loops in the graph gets complicated quickly, whereas as we shall see, CHM converges under general conditions, even in the presence of loops. However both approaches share the fact that they are probabilistic models.

1.2 Conditional Harmonic Mixing

The structure of the CHM graph will depend on the problem at hand: however all graphs share the weak constraint that for every[3] test node i, there must exist a path in the graph joining i with a training node. We will refer to such nodes as *label-connected*, and to the graph as a whole as label-connected if every test node in the graph is label-connected. A neighbour of a given node i is defined to be any node which is adjacent to node i, where 'adjacent' means that there exists an arc from j to i.

We use the following notation: we assume that the random variable at node i has M_i states (or classes), and that the arc from node i to node j carries a $M_j \times M_i$ conditional probability matrix P_{ji}. We adopt the convention that P_{ji} is the $M_j \times M_i$

3. For readability we use the indices i, j to denote the nodes themselves, since these quantities appear frequently as subscripts. We use the terms 'test node' and 'unlabeled node' interchangeably.

matrix of all zeros if there is no arc from node i to node j. We denote the posterior at node i by the vector $P_i \in \mathbb{R}^{M_i}$, for unlabeled nodes, and by $Q_i \in \mathbb{R}^{M_i}$ for labeled nodes. Denote the set of labeled nodes by $\mathcal{L}$, with $l \doteq |\mathcal{L}|$, and the set of unlabeled nodes by $\mathcal{U}$, with $u \doteq |\mathcal{U}|$, let $\mathcal{M}(i)$ ($\mathcal{N}(i)$) denote the set of indices of labeled (unlabeled) nodes adjacent to node i, and define $\mathcal{I} = \mathcal{M} \cup \mathcal{N}$ with $n(i) \doteq \|\mathcal{I}(i)\|$. Finally, for node i, let $p(i)$ be the number of incoming arcs from adjacent test nodes, and let $q(i)$ be the number of incoming arcs from adjacent train nodes.

1.2.1 The CHM Update Rule

A given node in the graph receives an estimate of its posterior from each of its neighbours. These estimates may not agree. Suppose that the hypothesised distribution at node i is Q_i, and let the estimates from its $n(i)$ neighbours be $P_j,\ j \in \mathcal{I}(i)$, so that $P_j = P_{jk}P_k$ for each $k \in \mathcal{I}(i)$. Given Q_i, the number of bits required to describe the distributions P_j is $\sum_j \{H(P_j) + D(P_j|Q_i)\}$, where H is the entropy and D the KL divergence. Since we wish to use Q_i to describe the combined distributions P_j as closely as possible, we require that this number of bits be minimized. For fixed P_j, this is accomplished by setting $(Q_i)_a = (1/n(i)) \sum_{j=1}^{n(i)} (P_j)_a$. A function on a graph is called harmonic (Doyle and Snell [1984], Zhu et al. [2003a]) if at each internal node the value of the function is the (possibly weighted) mean of the values at its neighbouring points (an internal node, as opposed to a boundary node, is one whose function value is not fixed; below we will just use the terms 'unlabeled node' and 'labeled node' for internal and boundary nodes). Assuming that a solution exists, then at the solution, the posterior at a given node is the weighted mean of the posteriors of its neighbours, where the weights are conditional probability matrices; hence the name 'Conditional Harmonic Mixing'.

Kullback Leibler Divergence

Harmonic Function

1.3 Learning in CHM Models

1.3.1 A Simple Model

It's useful to examine a simple model to fix ideas and to demonstrate a simple convergence proof. Consider the 3-point graph shown in Figure 1.2, with one labeled and two unlabeled nodes, and where to simplify the exposition we take the number of classes at each node to be C.

The consistency conditions arising from the above update rule are:

$$\begin{pmatrix} -1 & \frac{1}{2}P_{12} & \frac{1}{2}P_{13} \\ \frac{1}{2}P_{21} & -1 & \frac{1}{2}P_{23} \end{pmatrix} \begin{pmatrix} P_1 \\ P_2 \\ P_3 \end{pmatrix} = 0 \tag{1.1}$$

where $P_3 = (1, 0, 0, \dots)$ and where the ones in the matrices represent unit matrices. We wish to prove four properties of these equations, for any choice of conditional

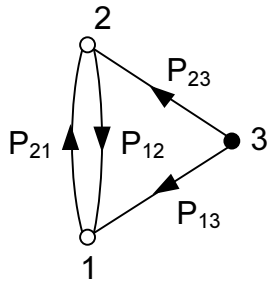

Figure 1.2 A simple 3-point CHM graph.

probability matrices P_{23}, P_{21}, P_{12} and P_{13}: first, that a solution always exists, second, that it is unique, third, that it results in stochastic vectors for the solution P_2 and P_3, and fourth, that Jacobi iterates will converge to it (by solving with Jacobi iterates, we will be able to take advantage of the sparseness of larger problems, as we will see below). Rearranging, we have

$$\begin{pmatrix} 1 & -\frac{1}{2}P_{12} \\ -\frac{1}{2}P_{21} & 1 \end{pmatrix} \begin{pmatrix} P_1 \\ P_2 \end{pmatrix} = \frac{1}{2} \begin{pmatrix} P_{13}P_3 \\ P_{23}P_3 \end{pmatrix} \tag{1.2}$$

The equations will always take this general form, where the matrix on the left is square (but not necessarily symmetric) and of side Cu, and where the left hand side depends only on the unlabeled points (whose posteriors we wish to find) and the right, only on the labeled points. Define:

$$b \doteq \frac{1}{2} \begin{pmatrix} P_{13}P_3 \\ P_{23}P_3 \end{pmatrix}, \quad M \doteq \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}, \quad N \doteq \begin{pmatrix} 0 & \frac{1}{2}P_{12} \\ \frac{1}{2}P_{21} & 0 \end{pmatrix} \tag{1.3}$$

and consider the following iterative algorithm for finding the solution, where $x^{(0)}$ is arbitrary:

$$Mx^{(k+1)} = Nx^{(k)} + b \tag{1.4}$$

With the above definitions, this is a Jacobi iteration (Golub and Van Loan [1996], p. 510), and we have:

Theorem 1 (Golub and Van Loan [1996], Theorem 10.1.1): Suppose $b \in \mathbb{R}^d$ and $\Delta \doteq M - N \in \mathbb{R}^{d \times d}$ is nonsingular. If M is nonsingular and the spectral radius of $M^{-1}N$ satisfies the inequality $\rho(M^{-1}N) < 1$, then the iterates $x^{(k)}$ defined by Eq. (1.4) converge to $x = \Delta^{-1}b$ for any starting vector x.

Stochastic Vectors and Matrices

Since N here is one half times a column-stochastic matrix, its eigenvalues have absolute value at most $\frac{1}{2}$, so $\rho(M^{-1}N) < 1$. Hence for this graph, a solution always exists and is unique. If we start with stochastic vectors everywhere (chosen arbitrarily on the unlabeled nodes), then they will remain stochastic since each

Jacobi iterate maintains this property, and the solution will be stochastic[4]. Note also that the matrix $M - N$ is diagonally dominant, and so has an inverse. However for the general case, N may not be proportional to a column stochastic matrix, and furthermore $M - N$ may not be diagonally dominant; we will need a more general argument.

1.3.2 A General Convergence Proof

At the CHM solution, for each node i, we have the consistency conditions

$$P_i - \frac{1}{p(i)+q(i)}\left(\sum_{j\in\mathcal{N}(i)} P_{ij}P_j\right) = \frac{1}{p(i)+q(i)}\left(\sum_{j\in\mathcal{M}(i)} P_{ij}Q_j\right) \tag{1.5}$$

where the right hand side is defined to be zero if $\mathcal{M}(i) = \emptyset$. Let $p = \sum_{i\in\mathcal{U}} M_i$, and define a block matrix $A \in \mathbb{R}^{p\times p}$ with ones along the diagonal and whose off-diagonal elements, which are either zero matrices or are the matrices $\frac{1}{p(i)+q(i)}P_{ij}$, are chosen so that Eq. (1.5) can be written as

$$AP = Q \tag{1.6}$$

where $P,\ Q \in \mathbb{R}^p$. Note that the right hand side of Eq. (1.6) is determined by the training data, that all conditional probability matrices associated with unlabeled-unlabeled arcs are in A, and that all conditional probability matrices associated with labeled-unlabeled arcs are in Q. Thus Eq. (1.5) corresponds to the i'th row of Eq. (1.6) and encapsulates M_i equations. Define the k'th Jacobi iterate by

Jacobi Iterations

$$P_i^{(k)} = \frac{1}{p(i)+q(i)}\left(\sum_{j\in N(i)} P_{ij}P_j^{(k-1)}\right) + \frac{1}{p(i)+q(i)}\left(\sum_{j\in M(i)} P_{ij}Q_j\right) \tag{1.7}$$

Referring to Theorem 1, we see that in this case, $A \doteq M - N$ where $M = \mathbf{I}$ and $N_{ij} \doteq \frac{1}{p(i)+q(i)}P_{ij}$ (recall that we define P_{ij} to be the matrix of all zeros, if there is no arc from node j to node i), and b is the second term on the right hand side of Eq. (1.7). Then the k'th Jacobi iterate takes the form $MP^{(k)} = NP^{(k-1)} + b$. We can now state:

Theorem 2: Consider a label-connected CHM graph with l labeled nodes. Assume that the vectors at the labeled nodes are fixed and stochastic. Then a solution to the corresponding CHM equations, Eq. (1.6), always exists and is unique. Furthermore at the solution, the vector $P_i^* \in \mathbb{R}^{M_i}$ at the ith unlabeled node is stochastic for all i, and the Jacobi iterates on the graph will always converge to the same solution, regardless of the initial values given to the P_i.

4. In fact this is true even if the initial vectors on the unlabeled nodes are chosen arbitrarily, by Theorem 1, since the solution is unique.

We present the proof in the proof-style advocated by (Lamport [1993]).

Proof:

1. **Assume: the CHM graph is label-connected.**
2. $\rho(N) < 1$.
   2.1. **Proof:** Consider the eigenvalue equation:

$$N\boldsymbol{\mu} = \lambda\boldsymbol{\mu} \tag{1.8}$$

Just as we view N as a block matrix whose i'th, j'th element is the matrix $\frac{1}{p(i)+q(i)}P_{ij}$, similarly view $\boldsymbol{\mu}$ as a vector whose i'th element is a vector of dimension M_i. Then let $\boldsymbol{\mu}_i$ be that component of $\boldsymbol{\mu}$ whose L_1 norm is largest (or any such component if there are more than one) and consider the corresponding rows of Eq. (1.8), which encapsulates the M_i equations

$$\frac{1}{p(i)+q(i)}\left(\sum_{j\in N(i)} P_{ij}\boldsymbol{\mu}_j\right) = \lambda\boldsymbol{\mu}_i \tag{1.9}$$

2.1.1 **Assume:** $q(i) > 0$ **and** $i : \|\boldsymbol{\mu}_i\|_1 \geq \|\boldsymbol{\mu}_j\|_1 \ \forall j$**.**

2.1.1.1 Since P_{ij} is column stochastic, it has unit L_1 norm. Thus

$$\begin{aligned}\left\|\sum_{j\in N(i)} P_{ij}\boldsymbol{\mu}_j\right\|_1 &\leq \sum_{j\in N(i)} \left\|P_{ij}\boldsymbol{\mu}_j\right\|_1 \\ &\leq \sum_{j\in N(i)} \|P_{ij}\|_1 \left\|\boldsymbol{\mu}_j\right\|_1 \\ &= \sum_{j\in N(i)} \|\boldsymbol{\mu}_j\|_1 \\ &\leq p(i)\|\boldsymbol{\mu}_i\|_1\end{aligned}$$

where $\|\cdot\|_1$ denotes the L_1 norm, and where the second line follows from an inequality satisfied by all p norms (Golub and Van Loan [1996]). Since by assumption $q(i) \geq 1$, taking the L_1 norm of both sides of Eq. (1.9) gives $|\lambda| < 1$.

2.1.2 **Assume:** $q(i) = 0$ **and** $i : \|\boldsymbol{\mu}_i\|_1 \geq \|\boldsymbol{\mu}_j\|_1 \ \forall j$**.**

2.1.2.1 The argument of 2.1.1.1, but with $q(i) = 0$, gives $|\lambda| \leq 1$, and if $|\lambda| = 1$, then each $\boldsymbol{\mu}_j$ appearing in the sum must have L_1 norm

equal to $\|\boldsymbol{\mu}_i\|_1$, since for $|\lambda| = 1$,

$$\begin{aligned} \|\boldsymbol{\mu}_i\|_1 &= \frac{1}{p(i)} \left\| \sum_{j \in N(i)} P_{ij} \boldsymbol{\mu}_j \right\|_1 \\ &\leq \frac{1}{p(i)} \sum_{j \in N(i)} \|P_{ij} \boldsymbol{\mu}_j\|_1 \\ &\leq \frac{1}{p(i)} \sum_{j \in N(i)} \|\boldsymbol{\mu}_j\|_1 \quad \leq \quad \|\boldsymbol{\mu}_i\|_1 \end{aligned}$$

where the last step follows from the assumption that $\boldsymbol{\mu}_i$ has largest L_1 norm. Thus for each $j \in N(i)$, we can repeat the above argument with $\boldsymbol{\mu}_j$ on the right hand side of Eq. (1.9), and the argument can then be recursively repeated for each $k \in N(j)$, until Eq. (1.9) has been constructed for every node for which a path exists to node i. However since the graph is label-connected, that set of nodes will include a test node which is adjacent to a train node. The previous argument, which assumed that $q > 0$, then shows that $|\lambda| < 1$. Thus, in general for any label-connected CHM graph, $|\lambda| < 1$, and so $\rho(N) < 1$.

3. A is nonsingular.

   3.1 **Proof:** Since $\rho(N) < 1$, the eigenvalues of N all lie strictly within the unit circle centered on the origin in the complex plane $\mathcal{C}$. Since $N = \mathbf{1} - A$ (cf. Eq. (1.6), if $\boldsymbol{e}$ is an eigenvector of A with eigenvalue λ, then it is an eigenvector of N with eigenvalue $1 - \lambda$, and so since $1 - \lambda$ lies strictly within the unit circle centered on the origin in $\mathcal{C}$, λ itself lies strictly within the unit circle centered on the point $\{1, 0\} \in \mathcal{C}$, so $\lambda \neq 0$. Hence none of A's eigenvalues vanish, and A is nonsingular.

4. A solution to the CHM equations exists and is unique.

   4.1 **Proof:** Since A is nonsingular, $AP = Q$ has unique solution $P = A^{-1}Q$.

5. At the solution, the random vector $P_i \in \mathbb{R}^{M_i}$ at each unlabeled node is stochastic, regardless of its initial value.

   5.1 **Proof:** For all unlabeled nodes, choose $P_i^{(0)}$ to be that stochastic vector whose first component is 1 and whose remaining components vanish. Then from Eqn. (1.7), by construction $P_i^{(k)}$ is stochastic for all k. Hence from Theorem 1 and steps 2 and 3 above, the Jacobi iterates will converge to a unique solution, and at that solution the P_i will be stochastic for all $i \in \mathcal{N}$. Finally, by Theorem 1, the same (unique) solution will be found regardless of the initial values of the P_i.

■

We emphasize the following points:

Conditional Probability Matrices

1. The theorem makes no assumptions on the conditional probability matrices,

beyond the requirement that they be column stochastic. In particular, it does not assume that the conditional probability matrices on the graph are consistent, that is, that there exists a joint probability from which all conditionals (or even any subset of them) could be derived by performing appropriate marginalizations. The CHM algorithm can therefore be applied using measured estimates of the conditional probability matrices, for which no precise joint exists.

2. In general A is not symmetric (and need not be row- or column-diagonally dominant).

3. No structure is imposed on the graph beyond its being label-connected. In particular, the graph can contain loops.

4. The numbers of classes at each node can differ, in which case the conditional probability matrices will be rectangular.

5. The model handles probabilistic class labels, that is, the Q_i can be arbitrary stochastic vectors.

Gauss-Seidel

6. To improve convergence, Gauss-Seidel iterations should be used, instead of Jacobi iterations. For Gauss-Seidel iterations, the error tends to zero like $\rho(M^{-1}N)^k$ (Golub and Van Loan [1996], p. 514).

1.4 Incorporating Prior Knowledge

Suppose that we are given the outputs of a given classifier on the dataset. The classifier was trained on the available labeled examples, but the amount of training data is limited and we wish to use SSL to improve the results. We can adopt an idea proposed in Zhu et al. [2003a], and for each node in the graph, attach an additional, labeled node, whose label is the posterior predicted for that datapoint. In fact CHM allows us to combine several classifiers in this way. This mechanism has the additional advantage of regularizing the CHM smoothing: the model can apply more, or less, weight to the original classifier outputs, by adjusting the conditionals on the arcs. Furthermore, for graphs that fall into several components, some of which are not label-connected, this method results in sensible predictions for the disconnected subgraphs; the CHM relaxation can be performed even for subgraphs containing no labeled data, since the base classifier still makes predictions for those nodes. In the context of CHM, for brevity we call this procedure of leveraging a base classifier over a graph 'lifting'. We will explore this approach empirically below.

Lifting

1.5 Learning the Conditionals

We are still faced with the problem of finding the conditional matrices P_{ij}. Here we propose one method for solving this, which we explore empirically below. Consider again the simple CHM model shown in Figure 1.2, and to simplify the exposition,

assume that the number of classes at each node is two, and in addition require that $P_l \doteq P_{13} = P_{23}$ and that $P_u \doteq P_{12} = P_{21}$ (l, u denoting labeled, unlabeled respectively). We can parameterize the matrices as:

$$P_l = \begin{pmatrix} 1\text{-}v_1 & v_2 \\ v_1 & 1\text{-}v_2 \end{pmatrix}, \quad P_u = \begin{pmatrix} 1\text{-}v_3 & v_4 \\ v_3 & 1\text{-}v_4 \end{pmatrix} \tag{1.10}$$

where $0 \leq v_i \leq 1 \ \forall i$. Now suppose that *the posteriors on every node in Figure 1.2 are given*, and denote components by e.g. $[P_{1a}, P_{1b}]$. In that case, Eq. (1.1) may be rewritten as:

$$\begin{bmatrix} -P_{3a} & P_{3b} & -P_{2a} & P_{2b} \\ P_{3a} & -P_{3b} & P_{2a} & -P_{2b} \\ -P_{3a} & P_{3b} & -P_{1a} & P_{1b} \\ P_{3a} & -P_{3b} & P_{1a} & -P_{1b} \end{bmatrix} \begin{bmatrix} v_1 \\ v_2 \\ v_3 \\ v_4 \end{bmatrix} = \begin{bmatrix} 2P_{1a} - P_{2a} - P_{3a} \\ 2P_{1b} - P_{2b} - P_{3b} \\ 2P_{2a} - P_{1a} - P_{3a} \\ 2P_{2b} - P_{1b} - P_{3b} \end{bmatrix} \tag{1.11}$$

which we summarize as $A\mathbf{v} = \mathbf{z}$. The matrix A in general need not be square, and if it is, it may be singular (as it is in this example), and even if it is nonsingular, computing $\mathbf{v}$ by inverting A is not guaranteed to give components v_i that lie in the interval $[0, 1]$. Thus instead we solve the quadratic programming problem:

$$\arg\min_{\mathbf{v}} \|A\mathbf{v} - \mathbf{z}\|^2 \ \text{ subject to } \ 0 \leq v_i \leq 1 \ \forall i \tag{1.12}$$

The posteriors P_i can simply be the outputs of a given classifier on the problem, if the classifier outputs are well-calibrated probabilities, or thresholded vectors (whose elements are 0 or 1) for arbitrary classifiers. To summarize: given some estimate of the posteriors on every node, the conditional probability matrices on the arcs can be learned by solving a quadratic programming problem.

1.6 Model Averaging

If sufficient labeled data is available, then a validation set can be used to determine the optimal graph architecture (i.e. to which neighbours each point should connect). However often labeled data is scarce, and in fact semi-supervised learning is really aimed at this case - that is, when labeled data is very scarce, but unlabeled data is plentiful. Thus in general for SSL methods it is highly desirable to find a way around having to use validation sets to choose either the model or its parameters. In this paper we will use model averaging: that is, for a given graph, given a classifier, use CHM to lift its results; then, do this for a variety of graphs, and simply average the posteriors assigned by CHM to each node, across all graphs. This, in combination with learning the conditionals, makes CHM a largely parameter-free approach (once a general algorithm for constructing the graphs has been chosen), although training using many graphs may be more computationally expensive than using a validation set to choose one.

1.7 Experiments

We applied CHM to the problem of text categorization, and to five of the benchmark classification tasks provided with this book.

1.7.1 Reuters-I Data Set

We applied CHM to the problem of the categorization of news articles in the Reuters-I data set Lewis [1997], with the ModApte split of 9,603 training files and 3,744 testing files. Each news article is assigned zero or more labels. Each label is considered to be an independent classification. We train and test on the 10 most common labels in the data set, which generates 10 separate binary classification problems.

One-sided, Two-sided Classification

We ran two kinds of classification experiment: one-sided (where only positive labeled examples are given), and two-sided (where labeled examples of both classes are given). The one-sided task is interesting because for some applications, it is much easier to obtain labeled data of one class, than of the other, and few algorithms can handle training data that contains no examples of one of the classes. For the one-sided problem we investigated using CHM to lift the Rocchio (inductive) classifier outputs, since of the methods we considered, only the Rocchio algorithm (Rocchio [1971]) was appropriate for the one-sided task.

Rocchio Algorithm

For the two-sided problem, we tested two inductive algorithms and one transductive algorithm. The two inductive algorithms were a linear support vector machine (Dumais et al. [1998], Drucker et al. [1999]) and a mixture of multinomial conditional models (Nigam et al. [2000]). For the mixture model, one multinomial model is used to model positive examples, while one or more multinomial models are used to model negative examples. EM is used to fit the mixture model to the negative examples. Hyperparameters (C for the linear SVM and the number of negative multinomials) are set by optimizing the microaveraged F1 score for the labels for Reuters classes 11-15 on the train/test split. These hyperparameters are then used for all ten classes. We also tested Nigam et. al's method for transduction using the multinomial mixture model (Nigam et al. [2000]). In that transductive method, EM is used, not only to learn the negative mixture, but also to infer the labels of the unlabeled data. Nigam et al. [2000] introduce another hyperparameter, which is the fractional weight to assign to each unlabeled case. This weight is also tuned by optimizing the microaveraged F1 score for classes 11-15.

Multinomial Models

For all experiments, we assume that the prior for the task is known. For the lifting arcs (i.e. those arcs joining the extra nodes carrying the classifier posteriors with the unlabeled nodes), we used unit conditional probability matrices, and we mapped the base classifier outputs to $\{0, 1\}$, both for the computation of the learned matrices, and for the CHM computation itself. We did this because the outputs of the base classifiers were far from well-calibrated probabilities (or from quantities that could be mapped to well-calibrated probabilities) as to be expected for very

small training sets.

For the two-sided case, of the three algorithms used, the SVMs were found to give the highest overall accuracy, and so we investigated lifting the SVM outputs with CHM. All of these algorithms make a hard decision about whether a test point is in, or out of, a class. For all algorithms, we choose the threshold for this decision point to reproduce the true number of positives on the entire test set (the 'known prior' assumption).

Note that the validation set was not used to tune the CHM model; it was used only to tune the baseline two-sided classifiers. The motivation for this is that we wish to see if CHM can be used to improve the accuracy of a given (black box) classifier, using only very limited training data.

1.7.1.1 Preprocessing, Graph Construction, and Training Data

Each document is pre-processed into a bag of words: that is, only the frequencies with which words appear in a document are used as features; the position of a word in a document is ignored. All words within the text, title, and body of the document are used, except words within the author or dateline, which are excluded. Words within an 11-word stopword list are also excluded. Every word is stemmed using the Porter stemmer (Porter [1980]), and the number of occurrences for each stem is computed for each document ('term frequency', or TF). The vector of TF values is then fed to the multinomial classifiers (which can only accept TF vectors). For all other classifiers and for constructing the CHM graph, we used TF-IDF features. Here, IDF (inverse document frequency) (Sparck-Jones [1972]) is the log of the ratio of the total number of documents to the number of documents in which a stemmed word appears. This log is multiplied by the term frequency to yield a TF-IDF feature. The TF-IDF vector for a document is then normalized to lie on the unit sphere.

Bag-Of-Words, TF, TF-IDF

For CHM, each graph is constructed using a simple nearest neighbour algorithm: an arc is added from node i to node j if node i is the k'th nearest neighbour of node j, for all $k \leq K$, where $K \in \{1, 3, 5, 7, 9, 11, 15, 20, 25, 30, 40, 50\}$, provided node j is not itself a labeled node. The conditional probability matrices for all arcs for a given k are shared; this imposes a form of regularization on the parameters, and embodies the idea that k alone should determine the type of link joining the two nodes. Note that this results in a directed graph which in general has no undirected equivalent (that is, a pair of unlabeled nodes can have an arc going one way but not the other, and labeled nodes only have outgoing arcs). The CHM posteriors at each unlabeled node were then averaged over all twelve graphs to arrive at the prediction. We tested two CHM algorithms: first, using unit conditional probability matrices, and second, using learned matrices.

Nearest Neighbour

For the one-sided task, we used labeled sets of size 1, 2, 5, 10, 20, 50 and 100, for each category. For the two-sided task, we used ten times as much labeled data for each experiment (i.e. labeled sets of size 10, 20, 50, 100, 200, 500 and 1000) to further explore dependence on training set size. The two-sided training sets are

shared amongst all classes: we ensure that at least one positive example for each of the 10 classes is present in all seven of the training sets.

The results are collected below. For the one-sided task, we plot F1 versus training set size, for Rocchio, Rocchio plus CHM with unit matrices, and Rocchio plus CHM for learned matrices in Figure 1.3. It's interesting that, although on this task using unit conditional probability matrices gives better mean results, the learned matrices have lower variance: the results for learned matrices rarely drop below the Rocchio baseline. Results for the two-sided task are collected in Tables 1.1 through 1.7, where we show results for all classifiers and for all categories, as well as the microaveraged results.

The Tables give F1 on the unlabeled subsets only. To determine statistical significance, we treat F1 score as proportions (similar to Yang and Liu [1999]). To be conservative with our confidence intervals, we treat the number of samples in the significance test to be the denominator of the F1 score: the number of false positives plus half the number of errors. Thus, we apply a one-way unbalanced ANOVA to predict correctness of the sample, given a factor which is the algorithm used. For those experiments where a main effect is found by ANOVA to be greater than 99% significant, a *post hoc* test comparing all pairs of algorithms is performed (using the Tukey-Kramer correction for repeated tests). In the Tables, we bold the results that are found by the post hoc test to be better than all of the unbolded algorithms (at a 99% confidence threshold).

For almost all experiments, the SVM gave higher F1 than the multinomial mixture or Nigam et al.'s algorithm. CHM gives better results than all other methods, at a 99% confidence threshold, for the case of 10 labeled points, and microaveraged over all data sets.

1.7.2 Benchmark Data Sets

We also applied CHM to five of the benchmark classification datasets provided with this book, namely datasets 1, 2, 4, 5 and 7. Each dataset contains 1500 points, with either 10 or 100 labeled points, and comes with a 12-fold validation split: all results quoted here are microaveraged over the twelve splits. A Support Vector Machine (SVM) was used as the base classifier in these experiments. Given the limited amount of training data, we chose to use a linear SVM with a very high C parameter (C=1000), which was effectively a hard-margin classifier.

1.7.2.1 Preprocessing, Graph Construction, and Training Data

For each fold of each data set, we trained a linear SVM with several different preprocessing alternatives:

- Raw data — no pre-processing.
- Scaling — after subtracting off the mean, scaling each feature so that they all have unit variance.

- Norming — after subtracting off the mean for each feature, scaling each data point so that they all have unit L_2 norm.
- Sphereing — applying PCA to the raw data, and using the latent coordinates as input.
- Chopping — as in Sphereing, but only choosing the top d latent dimensions, where d is chosen to cover 90% of the variance of the data.

One of these alternatives was chosen for each fold, by minimizing a generalization bound of the resulting SVM. For each fold, we compute

$$\arg\min_i R_i^2 ||\mathbf{w}_i^2|| \tag{1.13}$$

where $\mathbf{w}_i$ is the primal weight vector from the SVM on pre-processing alternative i, and R_i is the radius of the smallest ball that contains the data (after the ith pre-processing alternative). We approximate this radius by finding the distance of the data point that is farther from the mean over the whole data set. This choosing process usually picks one pre-processing method for all folds of a data set, often choosing "norming". However, some data sets (such as data set 4), alternate between "norming" and "sphereing".

We investigated a different graph construction mechanism from that used for the Reuters data. We call the algorithm "Flood Fill". The Flood Fill method was found to give similar results to the basic nearest neighbor method, but resulted in smaller graphs, leading to faster experiments (a typical run, for a given dataset, for both training set sizes, and for all 12 splits of the validation set, took approximately 50 minutes on a 3GHz machine). The Flood Fill method works as follows: choose some fixed positive integer n. Add n directed arcs from each labeled node to its nearest n unlabeled neighbours; all such arcs are assigned flavor = 1. Call the set of nodes reached in this way N_1 (where N_1 does not include the training nodes). For each node in N_1, do the same, allowing arcs to land on unlabeled nodes in N_1; assign all arcs generated in this way flavor = 2. At the ith iteration, arcs are allowed to fall on unlabeled nodes in N_i, but not on nodes $N_j,\ j < i$. The process repeats until either all nodes are reached, or until no further arcs can be added (note that graphs with disconnected pieces are allowed here). Here we smoothed (using model averaging) using values for n of 5, 9, 15, 25 and 50. The flood fill algorithm can create disconnected subgraphs, and since it is not clear how best to combine outputs of graphs with different connectedness, we simply thresholded the value at each node after each smoothing step, before taking the average.

We present the results in Tables 1.8 through Table 1.11. We chose two normalizations: the "normed/sphered/chopped" normalization, using the above bound; or just using the "normed" normalization everywhere, combined with a soft-margin linear SVM classifier (C=10). As in the Reuters experiments, the prior for each dataset is assumed known. The Tables give accuracies on the unlabeled subsets only. We applied a two-way ANOVA to assess the statistical significance of these

results, where the two factors are the fold number and the algorithm number, and the prediction of the ANOVA is the correctness of a sample. For those experiments where a main effect is found by ANOVA to be greater than a 99% significance level, a *post hoc* test comparing all pairs of algorithms is performed (using the Tukey-Kramer correction for repeated tests). Using a 99% ($p < 0.01$) significance level for the *post hoc* comparisons, we find the results shown in the Tables, where again statistical significance is indicated with bold versus normal typeface; the results can be summarized as follows:

- In no case is there a statistically significant difference between the learned conditional matrices, and the unit matrices, for CHM;
- CHM beats the SVM for all conditions for datasets 1 and 2.
- For the case of dataset 4, with normed-only preprocessing, and $l = 100$,SVM beats CHM;
- There is no statistically significant difference between results for dataset 5;
- For dataset 7, SVM beats CHM for $l = 10$, and CHM beats SVM for $l = 100$.

1.7.2.2 Discussion

This work demonstrates that CHM can be used to improve the performance of the best available classifier, on several datasets, when labeled data is limited. However the improvement is not uniform; for some datasets we observed that adding more smoothing (arcs) improved accuracy, while for others increased smoothing caused accuracy to drop. A method to accurately predict the required amount of smoothing, for a given problem, would boost the CHM accuracies significantly. We attempted to overcome this behaviour by model averaging, that is, averaging over different graphs, but this is a crude way to address the problem. Also in this paper we only discussed two simple heuristics for constructing the graphs; it would be useful to explore more sophisticated methods, for example, methods that compute a local metric (see, for example, Xing et al. [2003]). Other techniques for choosing the conditionals, for example using leave-one-out on the labeled set, or using a subgraph that is close to the labeled data, may also be fruitful to explore.

1.8 Conclusions

We have presented Conditional Harmonic Mixing (CHM), a graphical model that can be used for semi-supervised learning. CHM combines and improves upon earlier work in semi-supervised learning in several ways. First, unlike Bayes networks, CHM can model and learn using conditional probability distributions that do not have a consistent joint. This freedom allows us to learn and infer using simple linear algebra. Second, unlike Laplacian SSL, CHM can model asymmetric influences between random variables. Indeed, our random variables can have different cardinal-

ities: CHM is not limited to simply modeling harmonic functions. Finally, CHM can use a purely inductive algorithm to provide prior knowledge to the semi-supervised learning, which leads to superior performance on one-sided and two-sided empirical benchmarks. As the experiments show, one key open question for research is how to construct graphs that can take full advantage of semi-supervised learning with CHM: for a given dataset, for some choice of graphs the improvement is significant, while for other choices of graph for the same dataset, applying CHM can even reduce accuracy. This was addressed in the present work by simply using model averaging over graphs; it seems likely that better methods are possible.

Acknowledgements

We thank David Heckerman and Chris Meek for useful discussions.

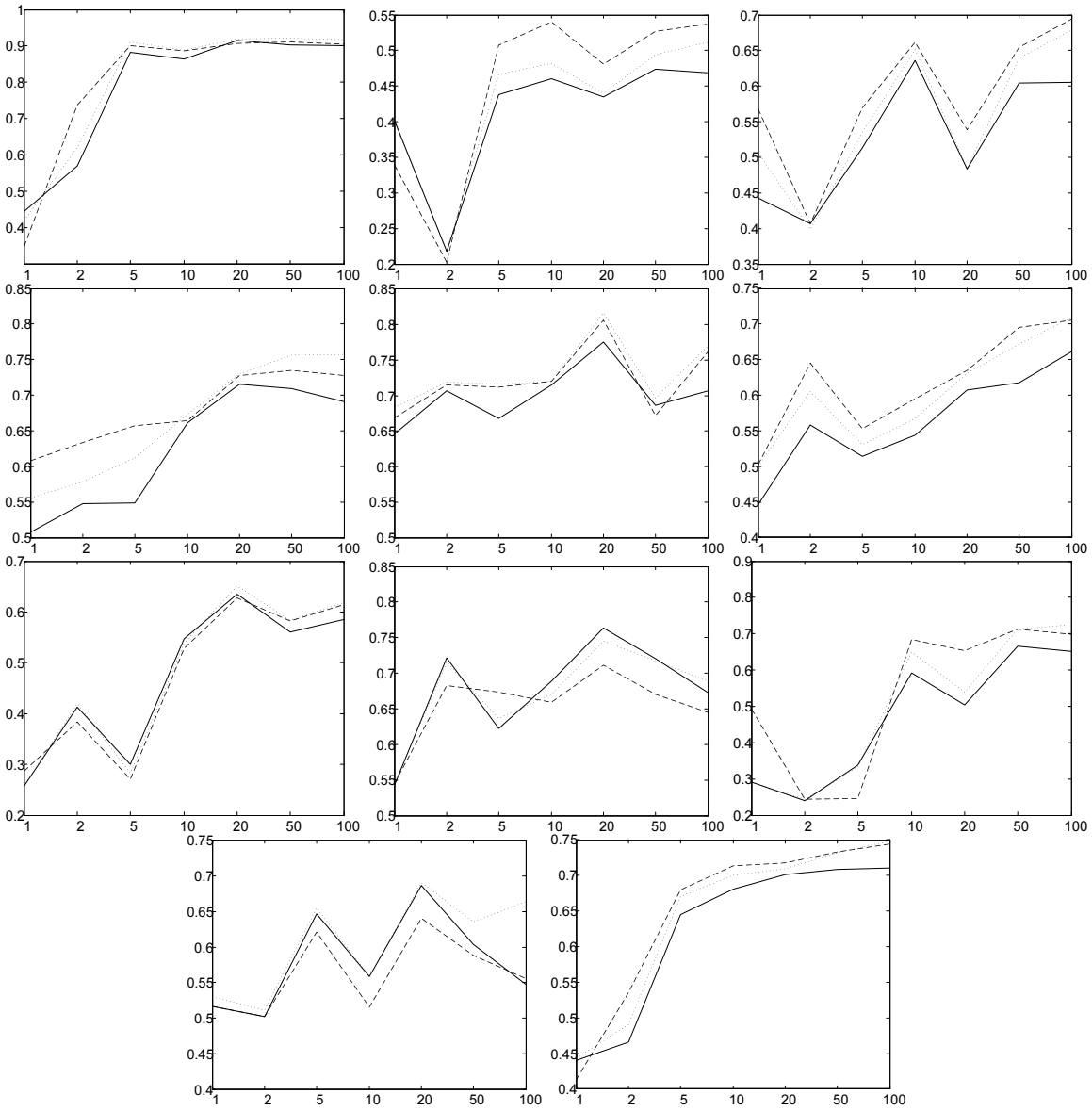

Figure 1.3 Results for the Rocchio classifiers (solid), Rocchio lifted with CHM, unit conditional matrices (dashed), and Rocchio lifted with learned conditional matrices (dotted). The y axis is F1, the x axis, training set size. Graphs are arranged left to right in order of increasing category. Most graphs have the y axis range chosen to be 0.35 for comparison. The last graph (bottom right) is the microaveraged results over all ten categories.

Category	Multinomial Mixture	Nigam	SVM	SVM/CHM Unit	SVM/CHM Learned
1	0.411	0.411	0.446	0.472	0.457
2	0.477	0.477	0.520	**0.603**	**0.592**
3	0.308	0.308	**0.463**	**0.520**	**0.509**
4	0.428	0.428	0.507	**0.628**	**0.538**
5	0.246	0.246	**0.466**	**0.515**	**0.482**
6	**0.249**	**0.249**	0.151	0.111	0.128
7	0.099	0.099	**0.474**	**0.478**	**0.507**
8	0.223	0.223	**0.454**	**0.504**	**0.472**
9	0.347	0.347	0.242	0.274	0.253
10	0.110	0.110	**0.233**	**0.271**	**0.233**
Microaverage	0.374	0.374	0.446	**0.491**	**0.475**

Table 1.1 F1 for top ten categories + microaverage F1, for training set size = 10

Category	Multinomial Mixture	Nigam	SVM	SVM/CHM Unit	SVM/CHM Learned
1	0.851	0.869	**0.903**	**0.899**	**0.908**
2	0.663	**0.797**	0.704	0.723	0.735
3	0.302	**0.401**	**0.453**	**0.516**	**0.497**
4	0.555	0.571	0.572	0.653	0.609
5	0.385	0.170	**0.477**	**0.563**	**0.527**
6	**0.285**	0.153	0.148	0.103	0.126
7	0.138	0.132	**0.488**	**0.484**	**0.507**
8	0.227	0.344	**0.507**	**0.521**	**0.525**
9	**0.407**	0.063	0.228	0.270	0.249
10	0.148	0.284	0.275	0.305	0.280
Microaverage	0.614	0.639	**0.678**	**0.694**	**0.696**

Table 1.2 F1 for top ten categories + microaverage F1, for training set size = 20

Category	Multinomial Mixture	Nigam	SVM	SVM/CHM Unit	SVM/CHM Learned
1	0.906	0.917	0.935	0.914	0.923
2	0.655	**0.711**	**0.735**	**0.741**	**0.749**
3	0.438	0.410	**0.579**	**0.681**	**0.633**
4	0.493	0.512	0.585	0.661	0.626
5	0.268	0.405	**0.666**	**0.697**	**0.708**
6	0.341	0.374	**0.514**	**0.545**	**0.535**
7	0.436	0.404	0.356	0.423	0.379
8	0.394	0.298	0.468	0.532	0.493
9	0.133	**0.274**	**0.256**	**0.288**	**0.270**
10	0.350	0.312	0.444	0.444	0.444
Microaverage	0.652	0.677	**0.734**	**0.748**	**0.744**

Table 1.3 F1 for top ten categories + microaverage F1, for training set size = 50

Category	Multinomial Mixture	Nigam	SVM	SVM/CHM Unit	SVM/CHM Learned
1	**0.917**	0.840	**0.939**	**0.912**	**0.921**
2	0.770	**0.863**	0.798	0.777	0.791
3	0.397	**0.596**	**0.535**	**0.599**	**0.559**
4	0.637	0.576	0.668	0.674	0.681
5	0.494	0.606	**0.728**	**0.772**	**0.773**
6	0.350	0.333	**0.522**	**0.573**	**0.542**
7	0.485	0.471	0.571	0.579	0.573
8	0.466	0.384	**0.680**	**0.658**	**0.673**
9	0.313	0.335	**0.489**	**0.641**	**0.595**
10	0.333	0.231	0.410	0.410	0.410
Microaverage	0.712	0.715	**0.778**	**0.776**	**0.778**

Table 1.4 F1 for top ten categories + microaverage F1, for training set size = 100

Category	Multinomial Mixture	Nigam	SVM	SVM/CHM Unit	SVM/CHM Learned
1	0.921	0.925	**0.950**	0.916	0.923
2	0.799	0.777	0.829	0.817	0.826
3	0.576	0.542	0.587	0.591	0.583
4	0.586	0.628	**0.729**	**0.737**	**0.734**
5	0.618	0.533	**0.754**	**0.782**	**0.788**
6	0.496	0.463	**0.696**	**0.723**	**0.715**
7	0.574	0.493	0.642	0.552	0.595
8	0.361	0.429	**0.721**	**0.721**	**0.729**
9	0.406	0.371	0.519	0.693	0.580
10	0.275	0.352	0.421	0.421	0.421
Microaverage	0.747	0.736	**0.813**	**0.801**	**0.804**

Table 1.5 F1 for top ten categories + microaverage F1, for training set size = 200

Category	Multinomial Mixture	Nigam	SVM	SVM/CHM Unit	SVM/CHM Learned
1	0.935	0.935	**0.957**	0.920	0.930
2	0.781	0.781	**0.869**	**0.835**	**0.847**
3	0.594	0.594	**0.691**	**0.687**	**0.683**
4	0.714	0.714	**0.816**	**0.814**	**0.820**
5	0.696	0.696	**0.806**	**0.821**	**0.824**
6	0.486	0.486	**0.704**	**0.723**	**0.716**
7	0.600	0.600	0.681	0.657	0.679
8	0.565	0.565	**0.827**	**0.775**	**0.801**
9	0.693	0.693	0.704	0.715	0.726
10	0.439	0.439	0.618	0.583	0.610
Microaverage	0.781	0.781	**0.856**	**0.831**	**0.840**

Table 1.6 F1 for top ten categories + microaverage F1, for training set size = 500

Category	Multinomial Mixture	Nigam	SVM	SVM/CHM Unit	SVM/CHM Learned
1	0.938	0.940	0.949	0.926	0.933
2	0.843	0.854	0.888	0.854	0.863
3	0.666	0.626	0.702	0.701	0.693
4	0.691	0.638	**0.859**	**0.827**	**0.833**
5	0.803	0.793	0.828	0.807	0.818
6	0.569	0.565	**0.724**	**0.716**	**0.720**
7	0.682	0.611	0.691	0.679	0.693
8	0.627	0.540	**0.841**	**0.762**	**0.802**
9	0.721	0.760	0.806	0.775	0.783
10	0.398	0.385	**0.738**	**0.719**	**0.724**
Microaverage	0.816	0.807	**0.869**	**0.844**	**0.852**

Table 1.7 F1 for top ten categories + microaverage F1, for training set size = 1000

DataSet (10)	SVM	SVM/CHM Unit	SVM/CHM Learned
1	0.803	**0.860**	**0.859**
2	0.751	**0.779**	**0.776**
4	0.543	0.530	0.530
5	0.605	0.613	0.612
7	**0.589**	0.567	0.566

Table 1.8 Accuracy for labeled sets of size 10, using normed/sphered/chopped preprocessing.

DataSet (100)	SVM	SVM/CHM Unit	SVM/CHM Learned
1	0.922	**0.966**	**0.966**
2	0.788	**0.816**	**0.813**
4	0.705	0.699	0.702
5	0.747	0.755	0.756
7	0.725	**0.755**	**0.755**

Table 1.9 Accuracy for labeled sets of size 100, using normed/sphered/chopped preprocessing.

DataSet (10)	SVM	SVM/CHM Unit	SVM/CHM Learned
1	0.803	**0.860**	**0.859**
2	0.760	**0.799**	**0.795**
4	0.541	0.525	0.527
5	0.605	0.613	0.612
7	**0.589**	0.567	0.566

Table 1.10 Accuracy for labeled sets of size 10, using normed preprocessing only.

DataSet (100)	SVM	SVM/CHM Unit	SVM/CHM Learned
1	0.922	**0.966**	**0.966**
2	0.870	**0.939**	**0.935**
4	**0.692**	0.600	0.613
5	0.747	0.755	0.756
7	0.725	**0.755**	**0.755**

Table 1.11 Accuracy for labeled sets of size 100, using normed preprocessing only.

References

Y. S. Abu-Mostafa. Machines that learn from hints. *Scientific American*, 272(4):64–69, 1995.

A. K. Agrawala. Learning with a probabilistic teacher. *IEEE Transactions on Information Theory*, 16:373–379, 1970.

C.J.C. Burges. Geometric Methods for Feature Selection and Dimensional Reduction. In L. Rokach and O. Maimon, editors, *Data Mining and Knowledge Discovery Handbook: A Complete Guide for Practitioners and Researchers*. Kluwer Academic, 2005.

O. Dekel, C.D. Manning, and Y. Singer. Log-linear models for label-ranking. In *Advances in Neural Information Processing Systems 16*. MIT Press, 2004.

P.G. Doyle and J.L. Snell. *Random Walks and Electric Networks*. The Mathematical Associatino of America, 1984.

H. Drucker, D. Wu, and V. Vapnik. Support vector machines for spam categorization. *IEEE Transactions on Neural Networks*, 10(5):1048–1054, 1999.

S.T. Dumais, J. Platt, D. Heckerman, and M. Sahami. Inductive learning algorithms and representations for text categorization. In *Proc. ACM International Conference on Information and Knowledge Management*, pages 148–155, 1998.

S. C. Fralick. Learning to recognize patterns wothout a teacher. *IEEE Transactions on Information Theory*, 13:57–64, 1967.

G.H. Golub and C.F. Van Loan. *Matrix Computations*. Johns Hopkins, third edition, 1996.

D. Heckerman, D.M. Chickering, C. Meek, R. Rounthwaite, and C. Kadie. Dependency networks for inference, collaborative filtering, and data visualization. *Journal of Machine Learning Research*, 1:49–75, 2001.

M.I. Jordan, editor. *Learning in Graphical Models*. MIT Press, 1999.

L. Lamport. How to write a proof. *American Mathematical Monthly*, 102(7):600–608, 1993.

D.D. Lewis. The reuters-21578 data set. `http://www.daviddlewis.com/resources/testcollections/reuters21578/`, 1997.

K. Nigam, A. McCallum, S. Thrun, and T. Mitchell. Text classification from labeled and unlabeled documents using EM. *Machine Learning*, 39(2/3):103–134, 2000.

M. Porter. An algorithm for suffix stripping. *Program*, 14(3):130–137, 1980.

J. Rocchio. Relevance feedback in information retrieval. In G. Salton, editor, *The SMART Retrieval System: Experiments in Automatic Document Processing*, pages 313–323. Prentice-Hall, 1971.

H. J. Scudder. Probability of error of some adaptive pattern-recognition machines. *IEEE Transactions on Information Theory*, 11:363–371, 1965.

K. Sparck-Jones. A statistical interpretation of term specificity and its application in retrieval. *J. of Documentation*, 28(1):11–21, 1972.

E.P. Xing, A.Y. Ng, M.I. Jordan, and S. Russell. Distance metric learning, with application to clustering with side information. In *Advances in Neural Information Processing Systems*, volume 13. The MIT Press, 2003.

Y. Yang and X. Liu. A re-examination of text categorization methods. In *Proc. 21st International ACM SIGIR Conf.*, pages 42–49, 1999.

D. Zhou, O. Bousquet, T. Lal, J. Weston, and B. Schölkopf. Learning with local and global consistency. In *NIPS*, volume 16, 2003.

D. Zhou, J. Huang, and B. Schölkopf. Learning from Labeled and Unlabeled Data on a Directed Graph. In Luc De Raedt and Stefan Wrobel, editors, *Proceedings of the 22nd International*

Conference on Machine Learning, 2005a.

D. Zhou, B. Schölkopf, and T. Hofmann. Semi-supervised learning on directed graphs. In *Advances in Neural Information Processing Systems 18*. MIT Press, 2005b.

X. Zhu and Z. Ghahramani. Learning from labeled and unlabeled data with label propagation. Technical report, Carnegie Mellon University, 2002.

X. Zhu, Z. Ghahramani, and J. Lafferty. Semi-supervised learning using gaussian fields and harmonic functions. In *ICML*, 2003a.

X. Zhu, J. Lafferty, and Z. Ghahramani. Semi-supervised learning: from Gaussian fields to Gaussian processes. Technical Report CMU-CS-03-175, Carnegie Mellon University, 2003b.

Notation and Symbols

Sets of Numbers

$\mathbb{N}$	the set of natural numbers, $\mathbb{N} = \{1, 2, \dots\}$
$\mathbb{R}$	the set of reals
$[n]$	compact notation for $\{1, \dots, n\}$
$x \in [a, b]$	interval $a \leq x \leq b$
$x \in (a, b]$	interval $a < x \leq b$
$x \in (a, b)$	interval $a < x < b$
$\lvert C \rvert$	cardinality of a set C (for finite sets, the number of elements)

Data

$\mathcal{X}$	the input domain
d	(used if $\mathcal{X}$ is a vector space) dimension of $\mathcal{X}$
M	number of classes (for classification)
l, u	number of labeled, unlabeled training examples
n	total number of examples, $n = l + u$.
i, j	indices, often running over $[l]$ or $[n]$
x_i	input patterns $x_i \in \mathcal{X}$
y_i	classes $y_i \in [M]$ (for regression: target values $y_i \in \mathbb{R}$)
X	a sample of input patterns, $X = (x_1, \dots, x_n)$
Y	a sample of output targets, $Y = (y_1, \dots, y_n)$
X_l	labeled part of X, $X_l = (x_1, \dots, x_l)$
Y_l	labeled part of Y, $Y_l = (y_1, \dots, y_l)$
X_u	unlabeled part of X, $X_u = (x_{l+1}, \dots, x_{l+u})$
Y_u	unlabeled part of Y, $Y_u = (y_{l+1}, \dots, y_{l+u})$

Kernels

$\mathcal{H}$	feature space induced by a kernel
Φ	feature map, $\Phi : \mathcal{X} \to \mathcal{H}$
k	(positive definite) kernel
K	kernel matrix or Gram matrix, $K_{ij} = k(x_i, x_j)$

Vectors, Matrices and Norms

$\mathbf{1}$	vector with all entries equal to one
$\mathbf{I}$	identity matrix
$A^\top$	transposed matrix (or vector)
A^{-1}	inverse matrix (in some cases, pseudo-inverse)
$\mathbf{tr}\,(A)$	trace of a matrix
$\mathbf{det}\,(A)$	determinant of a matrix
$\langle \mathbf{x}, \mathbf{x}' \rangle$	dot product between $\mathbf{x}$ and $\mathbf{x}'$
$\|\cdot\|$	2-norm, $\|\mathbf{x}\| := \sqrt{\langle \mathbf{x}, \mathbf{x} \rangle}$
$\|\cdot\|_p$	p-norm , $\|\mathbf{x}\|_p := \left(\sum_{i=1}^N |x_i|^p\right)^{1/p}$, $N \in \mathbb{N} \cup \{\infty\}$
$\|\cdot\|_\infty$	∞-norm , $\|\mathbf{x}\|_\infty := \sup_{i=1}^N |x_i|$, $N \in \mathbb{N} \cup \{\infty\}$

Functions

ln	logarithm to base e
$\log_2$	logarithm to base 2
f	a function, often from $\mathcal{X}$ or $[n]$ to $\mathbb{R}$, $\mathbb{R}^M$ or $[M]$
$\mathcal{F}$	a family of functions
$L_p(\mathcal{X})$	function spaces, $1 \leq p \leq \infty$

Probability

$\mathrm{P}\{\cdot\}$	probability of a logical formula
$\mathrm{P}(C)$	probability of a set (event) C
$p(x)$	density evaluated at $x \in \mathcal{X}$
$\mathbf{E}\,[\cdot]$	expectation of a random variable
$\mathbf{Var}\,[\cdot]$	variance of a random variable
$\mathcal{N}(\mu, \sigma^2)$	normal distribution with mean μ and variance σ^2

Graphs

$\mathfrak{g}$	graph $\mathfrak{g} = (V, E)$ with nodes V and edges E
$\mathcal{G}$	set of graphs
$\mathbf{W}$	weighted adjacency matrix of a graph ($\mathbf{W}_{ij} \neq 0 \Leftrightarrow (i,j) \in E$)
$\mathbf{D}$	(diagonal) degree matrix of a graph, $\mathbf{D}_{ii} = \sum_j W_{ij}$
$\mathfrak{L}$	normalized graph Laplacian, $\mathfrak{L} = \mathbf{I} - \mathbf{D}^{-1/2}\mathbf{W}\mathbf{D}^{-1/2}$
L	un-normalized graph Laplacian, $L = \mathbf{D} - \mathbf{W}$

SVM-related

$\rho_f(x, y)$	margin of function f on the example (x, y), i.e., $y \cdot f(x)$
ρ_f	margin of f on the training set, i.e., $\min_{i=1}^m \rho_f(x_i, y_i)$
h	VC dimension
C	regularization parameter in front of the empirical risk term
λ	regularization parameter in front of the regularizer
$\mathbf{w}$	weight vector
b	constant offset (or threshold)
α_i	Lagrange multiplier or expansion coefficient
β_i	Lagrange multiplier
$\boldsymbol{\alpha}, \boldsymbol{\beta}$	vectors of Lagrange multipliers
ξ_i	slack variables
$\boldsymbol{\xi}$	vector of all slack variables
Q	Hessian of a quadratic program

Miscellaneous

I_A	characteristic (or indicator) function on a set A i.e., $I_A(x) = 1$ if $x \in A$ and 0 otherwise
δ_{ij}	Kronecker δ ($\delta_{ij} = 1$ if $i = j, 0$ otherwise)
δ_x	Dirac δ, satisfying $\int \delta_x(y) f(y) dy = f(x)$
$O(g(n))$	a function $f(n)$ is said to be $O(g(n))$ if there exist constants $C > 0$ and $n_0 \in \mathbb{N}$ such that $\lvert f(n)\rvert \leq Cg(n)$ for all $n \geq n_0$
$o(g(n))$	a function $f(n)$ is said to be $o(g(n))$ if there exist constants $c > 0$ and $n_0 \in \mathbb{N}$ such that $\lvert f(n)\rvert \geq cg(n)$ for all $n \geq n_0$
rhs/lhs	shorthand for "right/left hand side"
■	the end of a proof