Efficient Multiclass Implementations of L1-Regularized Maximum Entropy

Patrick Haffner Steven Phillips
AT&T Labs – Research
180 Park Avenue, Florham Park, NJ 07932
{*haffner, phillips*}*@research.att.com*

Rob Schapire
Princeton University
35 Olden Street, Princeton, NJ 08544
schapire@cs.princeton.edu

Abstract

This paper discusses the application of L1-regularized maximum entropy modeling or *SL1-Max* [9] to multiclass categorization problems. A new modification to the SL1-Max fast sequential learning algorithm is proposed to handle *conditional* distributions. Furthermore, unlike most previous studies, the present research goes beyond a single type of conditional distribution. It describes and compares a variety of modeling assumptions about the class distribution (independent or exclusive) and various types of joint or conditional distributions. It results in a new methodology for combining binary regularized classifiers to achieve multiclass categorization. In this context, Maximum Entropy can be considered as a generic and efficient regularized classification tool that matches or outperforms the state-of-the art represented by AdaBoost and SVMs.

1 Introduction

A new form of maximum entropy (maxent) with a *sequential* updating procedure and *L1* regularization (SL1-Max) was recently introduced [9] as a probability distribution estimation technique. This study adapts SL1-Max to classification problems. It demonstrates a regularized linear classification algorithm that bears striking similarities with large margin classifiers such as AdaBoost [19, 7].

Conditional maxent models [4] (also known as conditional exponential or logistic regression models) were previously applied to classification problems in text classification [16]. These models were shown [10] to be a generalization of Support Vector Machines (SVMs) [20] or a modification of AdaBoost normalized to form a conditional distribution [13]. The three aforementioned references employed the L2 type of regularization. L1 regularization was proposed for logistic regression [15], which is a particular case of maximum entropy. The application of conditional maxent to part-of-speech tagging or machine translation problems [17] can also be seen as a classification problem, where the number of classes is very large. Solutions dealing with the computational and memory usage issues arising from this large number of classes were proposed for translation applications. Most of these studies focus on *specific* applications. We could not find studies of maxent as a generic classification algorithm that can be applied to a wide range of problems.

This situation can be contrasted to the literature on large margin classifiers, where extant studies [2] cover their adaptation to multiclass problems. Large margin classifiers were initially demonstrated on binary classification problems and extended to multiclass classification through various schemes combining these binary classifiers. The simplest scheme is to train binary classifiers to distinguish the examples belonging to one class from the examples not belonging to this class. This approach is usually referred to in the literature

as *1-vs-other* or *1-vs-all*. Many other combination schemes are possible, in particular *1-vs-1* where each classifier is trained to separate one class from another. More general combination schemes include error correcting output codes (ECOC) [8] and hierarchies of classifiers.

Our goal is to include maxent among the regularized classification algorithms one would routinely consider, and implement it in a software package that would be as easy to use as SVMs and Adaboost packages. The expected advantage of maxent over other classification algorithms is its flexibility, both in terms of choice of distribution and modeling assumptions. SL1-Max provides the ideal starting point for this work: this algorithm estimates maxent model parameters in a fast sequential manner, and supports an effective and well understood L1 regularization scheme (which leads to sparser solutions than L2 regularization). The new contributions in this paper are the following. We adapt SL1-Max to conditional distributions, which requires the derivation on a new bound on the decrease in the loss. We compare the joint and the class-conditional distributions to the conditional distribution traditionally considered in the literature. We introduce "non-class" maxent models to reduce multiclass problems to a set of binary problems (to our knowledge, such techniques have only been used in question answering systems [18]). We show, through experiments, that maxent statistical interpretation leads to a new methodology for selecting the optimal multiclass approach for a given application.

Section 2 introduces the notation to handle classification problems. In Section 3, we adapt the SL1-Max algorithm to estimate parameters of the joint, class-conditional and conditional distributions. In Section 4, we generalize these techniques to the multi-label case. After a discussion about the implementation in Section 5, comparative experiments are provided in Section 6.

2 Definitions and notation

Our sample space covers input-label associative pairs $(\mathbf{x}, c) \in X \times \{1, \ldots, l\}$. Our goal is to determine, for a given input $\mathbf{x}$, the most likely class label c^* which maximizes the unknown conditional distribution $c^* = \operatorname{argmax}_c p(c|\mathbf{x})$. For simplicity, this paper initially focuses on classification where each input is associated with a single label. From a statistical viewpoint, this means that classes are *exclusive* (i.e. they cannot occur simultaneously). Models where multiple labels are allowed will be considered later.

The application of Bayes Rules writes

$$p(c|\mathbf{x}) = \frac{p(\mathbf{x}, c)}{p(\mathbf{x})} = p(c)\frac{p(\mathbf{x}|c)}{p(\mathbf{x})}. \tag{1}$$

As $p(\mathbf{x})$ does not impact the choice of the class, we can choose which distribution we want to estimate: *joint* with $p(\mathbf{x}, c)$, *conditional* with $p(c|\mathbf{x})$, or *class-conditional* with $p(\mathbf{x}|c)$. The rest of this section introduces notation to manipulate these distributions in a consistent fashion.

In maxent, training data is used to impose constraints on the distribution. Each constraint expresses a characteristic of the training data that must be learned in the estimated distribution p. Typical constraints require features to have the same expected value as in the training data. Features are real valued functions of the input and of the class $f(\mathbf{x}, c)$. To represent the average of a feature f over a distribution p, we use the following notation:

Joint: The expected value of f under p is

$$p[f] = \sum_{\mathbf{x},c} p(\mathbf{x}, c) f(\mathbf{x}, c).$$

Conditional: For a given training example $\mathbf{x}_i$, $p_i(c) = p(c|\mathbf{x}_i)$ and

$$p_i[f] = \sum_c p(c|\mathbf{x}_i) f(\mathbf{x}_i, c) = \sum_c p_i(c) f(\mathbf{x}_i, c).$$

Class-conditional: For a given class c, $p_c(\mathbf{x}) = p(\mathbf{x}|c)$ and

$$p_c[f] = \sum_{\mathbf{x}} p(\mathbf{x}|c) f(\mathbf{x}, c) = \sum_{\mathbf{x}} p_c(\mathbf{x}) f(\mathbf{x}, c).$$

The training data is a set of input-label pairs $(\mathbf{x}_1, c_1), \ldots, (\mathbf{x}_m, c_m)$, and is a subset of the sample space $\{\mathbf{x}_1, \ldots, \mathbf{x}_m\} \times \{1, \ldots, l\}$. The empirical distributions over this training set are defined as follows:

$$\begin{aligned}
\tilde{p}(\mathbf{x}, c) &= \frac{1}{m} \left|\{1 \le i \le m : \mathbf{x}_i = \mathbf{x} \text{ and } c_i = c\}\right| \\
\tilde{p}_c(\mathbf{x}) &= \frac{\left|\{1 \le i \le m : \mathbf{x}_i = \mathbf{x} \text{ and } c_i = c\}\right|}{\left|\{1 \le i \le m : c_i = c\}\right|} \\
\tilde{p}(c) &= \frac{1}{m} \left|\{1 \le i \le m : c_i = c\}\right|
\end{aligned}$$

All maxent models are based on the computation of a linear score over the features, represented by the inner product between the feature vector and the weight vector $\boldsymbol{\lambda}$. In the classification case, the feature vector $\mathbf{f}(\mathbf{x}, c)$ is defined over all input-label pairs $(\mathbf{x}, c)$. This pair is scored with the inner product $\boldsymbol{\lambda} \cdot \mathbf{f}(\mathbf{x}, c)$ and compared to other pairs $(\mathbf{x}, d)$ with $d \neq c$. A subtlety that arises in the application of maxent to classification problems is the need to multiply each feature as many times as there are classes. Suppose the input $\mathbf{x}$ is a list of n values $v_1(\mathbf{x}), \ldots, v_n(\mathbf{x})$, the class dependent features are defined as follows:

$$f_{d,j}(\mathbf{x}, c) = \begin{cases} v_j(\mathbf{x}) & \text{if } c = d \\ 0 & \text{otherwise} \end{cases} \tag{2}$$

With this representation,the inner product between the feature vector $\mathbf{f}(\mathbf{x}, c)$ and the parameter vector $\boldsymbol{\lambda}$ simplifies as

$$\boldsymbol{\lambda} \cdot \mathbf{f}(\mathbf{x}, c) = \sum_{d,j} \lambda_{d,j} f_{d,j}(\mathbf{x}, c) = \sum_j \lambda_{c,j} v_j(\mathbf{x}) = \boldsymbol{\lambda}_c \cdot \mathbf{v}(\mathbf{x}),$$

where $\boldsymbol{\lambda}_c$ is the subset of parameters specific to class c.

3 Trying Different Distributions

This section provides SL1-Max solutions for the estimation of the *joint*, *class-conditional* and *conditional* distributions. It also shows some limitations of these solutions that will be overcome in the next section. The first subsection focuses on joint distribution $p(\mathbf{x}, c)$.

3.1 Estimating joint distributions

Maximum Entropy restricts the trained model distribution p so that each feature has the same expected and empirical means. Our notation summarizes this constraint as $p[f_{d,j}] = \tilde{p}[f_{d,j}]$. In the regularized case, this constraint is softened to have the form $|p[f_{d,j}] - \tilde{p}[f_{d,j}]| \le \beta_{d,j}$, where $\beta_{d,j}$ is a regularization parameter.

Within these constraints, we are looking for the distribution which is the closest to the uniform distribution, by maximizing the entropy $H(p) = -\sum_{i,c} p(\mathbf{x}_i, c) \ln p(\mathbf{x}_i, c)$. This corresponds to the convex program:

$$\mathcal{P}_1 : \quad \max_p H(p) \text{ subject to} \begin{cases} \sum_{i,c} p(\mathbf{x}_i, c_i) = 1 \\ \forall d, j : |p[f_{d,j}] - \tilde{p}[f_{d,j}]| \leq \beta_{d,j} \end{cases}$$

The dual program maximizes the likelihood over the exponential distributions.

$$\mathcal{Q}_1(\boldsymbol{\lambda}) : \quad \min_{\boldsymbol{\lambda}} L_{\tilde{p}}^{\boldsymbol{\beta}}(\boldsymbol{\lambda}) \text{ with} \begin{cases} L_{\tilde{p}}^{\boldsymbol{\beta}}(\boldsymbol{\lambda}) = \underbrace{-\tilde{p}[\ln q_{\boldsymbol{\lambda}}]}_{Likelihood} + \underbrace{\sum_{d,j} \beta_{d,j} |\lambda_{d,j}|}_{Regularization} \\ q_{\boldsymbol{\lambda}}(\mathbf{x}, c) = \frac{1}{Z_{\boldsymbol{\lambda}}} e^{\boldsymbol{\lambda} \cdot \mathbf{f}(\mathbf{x}, c)} \\ Z_{\boldsymbol{\lambda}} = \sum_c \sum_i e^{\boldsymbol{\lambda} \cdot \mathbf{f}(\mathbf{x}_i, c)} \end{cases}$$

Note that for the joint distribution, the $Z_{\boldsymbol{\lambda}}$ normalization is performed over all classes and all training samples. [9] prove the convergence of a sequential-update algorithm that modifies one weight at a time. This coordinate-wise descent is particularly efficient when dealing with a large number of sparse features. A bound on the decrease in the loss is

$$L_{\tilde{p}}^{\boldsymbol{\beta}}(\boldsymbol{\lambda}') - L_{\tilde{p}}^{\boldsymbol{\beta}}(\boldsymbol{\lambda}) \leq -\delta \tilde{p}[f_{d,j}] + \ln\left(1 + (e^{\delta} - 1) q_{\boldsymbol{\lambda}}[f_{d,j}]\right) + \beta_{d,j} \left(|\lambda_{d,j} + \delta| - |\lambda_{d,j}|\right), \quad (3)$$

with equality if we have binary features. The values of δ that minimize this expression can be obtained in a closed form. Note that this analysis must be repeated for all features j and all classes d.

Efficient implementations of the sequential-update algorithm require storing numerous variables and intermediate computations. For instance, we need to store all the $q_{\boldsymbol{\lambda}}(\mathbf{x}, c)$ and $\boldsymbol{\lambda} \cdot \mathbf{f}(\mathbf{x}_i, c_i)$. The storage requirement in $O(m \times n)$ can be problematic for large-scale problems.

As a matter of fact, we found memory requirements to be the main limitation of this implementation of multiclass SL1-Max. Speedup techniques based on partial pricing strategies [5] have reduced the learning time of SL1-Max and made it manageable.

3.2 Estimating class-conditional distributions

The motivation for using the class-conditional distribution $p(\mathbf{x}|c)$ is that it allows to build one model per class. From Eq.(2), it is easy to see that for $d \neq c$, features $f_{d,j}$ have no impact on the class-conditional distribution p_c we are trying to estimate. As a result, separate optimization problems can be defined for the l classifiers with no interaction between them. For each class, the convex problem $\mathcal{P}_2$ and its convex dual $\mathcal{Q}_2(\boldsymbol{\lambda}_c)$ are:

$$\mathcal{P}_2 : \quad \max_{p_c} H(p_c) \text{ subject to} \begin{cases} \sum_i p_c(\mathbf{x}_i) = 1 \\ \forall j : |p_c[f_{c,j}] - \tilde{p}_c[f_{c,j}]| \leq \beta_{c,j} \end{cases}$$

$$\mathcal{Q}_2(\boldsymbol{\lambda}_c): \quad \min_{\boldsymbol{\lambda}_c} L_{\tilde{p}}^{\boldsymbol{\beta}}(\boldsymbol{\lambda}_c) \text{ with}$$
$$\begin{cases} L_{\tilde{p}}^{\boldsymbol{\beta}}(\boldsymbol{\lambda}_c) = -\tilde{p}_c[\ln q_{\boldsymbol{\lambda}}] + \sum_j \beta_{c,j} |\lambda_{c,j}| \\ q_{\boldsymbol{\lambda}}(\mathbf{x}, c) = \frac{1}{Z_{\boldsymbol{\lambda}}(c)} e^{\boldsymbol{\lambda} \cdot \mathbf{f}(\mathbf{x}, c)} \\ Z_{\boldsymbol{\lambda}}(c) = \sum_i e^{\boldsymbol{\lambda} \cdot \mathbf{f}(\mathbf{x}_i, c)} \end{cases}$$

For each class c, we have an independent optimization problem to solve. For each of these optimization problems, we have a general solution which is just SL1-Max.

A clear advantage of the *class-conditional* over the *joint* distribution approach is that, when optimizing p_c, we do not have to store the variables used to optimize the class-conditional distributions of the other classes. This provides huge savings in memory (i.e the memory requirement is divided by the number of classes).

A drawback of the class-conditional approach is that it does not minimize explicitly the classification error rate. To obtain the recognized class, it relies on the application of the Bayes rules, and thus on the fact that the probability distributions have been properly estimated. Taking the logarithm of $\operatorname{argmax}_c \tilde{p}(c) q_{\boldsymbol{\lambda}}(\mathbf{x}, c)$, this class is [1] $\operatorname{argmax}_c (\boldsymbol{\lambda} \cdot \mathbf{f}(\mathbf{x}, c) - \ln Z_{\boldsymbol{\lambda}}(c) + \ln \tilde{p}(c))$.

3.3 Estimating conditional distributions

In this section, we propose a novel extension to the SL1-Max algorithm to estimate the parameters of the maxent model for the conditional distribution $p(c|\mathbf{x})$. In the literature [16, 17], conditional maxent is typically the only distribution considered for classification, as it is expected to be the most discriminant. However, its optimization turns out to be more complex, so we present it last.

In the case conditional distributions, the main challenge is that, for each training sample i, we want to estimate one separate distribution over the classes $\tilde{p}_i$. At the same time, the constraints apply to the entire training set and tie up these distributions. If we trained each distribution separately for each sample i, constraints would be $\forall d, j : |p_i[f_{d,j}] - \tilde{p}_i[f_{d,j}]| \leq \beta_{d,i,j}$. This would result in $m \times n \times l$ learnable parameters, with obvious overfitting. On the other hand, summing these constraints over the examples produces $\left| \frac{1}{m} \sum_i p_i[f_{d,j}] - \tilde{p}[f_{d,j}] \right| \leq \beta_{d,j}$. This formulation was used before [16], but we added regularization and a different sequential-update algorithm.

The two optimization problems are:

$$\mathcal{P}_3: \quad \max_{p_1,\ldots,p_m} \sum_i H(p_i) \text{ subject to}$$
$$\begin{cases} \forall i : \sum_c p_i(c) = 1 \\ \forall d, j : \left| \frac{1}{m} \sum_i p_i[f_{d,j}] - \tilde{p}[f_{d,j}] \right| \leq \beta_{d,j} \end{cases}$$

[1] Note that this relies on a good estimation of the $Z_{\boldsymbol{\lambda}}(c)$ normalization factors, With a joint distribution, there is no need to compute the $Z_{\boldsymbol{\lambda}}$ normalization factor and $\operatorname{argmax}_c q_{\boldsymbol{\lambda}}(\mathbf{x}, c) = \operatorname{argmax}_c \boldsymbol{\lambda} \cdot \mathbf{f}(\mathbf{x}, c)$.

$$\mathcal{Q}_3(\boldsymbol{\lambda}): \quad \min_{\boldsymbol{\lambda}} L_{\tilde{p}}^{\boldsymbol{\beta}}(\boldsymbol{\lambda}) \text{ with}$$
$$\begin{cases} L_{\tilde{p}}^{\boldsymbol{\beta}}(\boldsymbol{\lambda}) = -\frac{1}{m}\sum_i \tilde{p}_i[\ln q_{\boldsymbol{\lambda}}] + \sum_{d,j} \beta_{d,j} \left|\lambda_{d,j}\right| \\ q_{\boldsymbol{\lambda}}(\mathbf{x}, c) = \frac{1}{Z_{\boldsymbol{\lambda}}(\mathbf{x})} e^{\boldsymbol{\lambda}\cdot\mathbf{f}(\mathbf{x},c)} \\ Z_{\boldsymbol{\lambda}}(\mathbf{x}) = \sum_c e^{\boldsymbol{\lambda}\cdot\mathbf{f}(\mathbf{x},c)} \end{cases}$$

The likelihood can be expanded as

$$L_{\tilde{p}}^{\boldsymbol{\beta}}(\boldsymbol{\lambda}) = -\sum_{d,j} \lambda_{d,j}\tilde{p}[f_{d,j}] + \frac{1}{m}\sum_i \ln Z_{\boldsymbol{\lambda}}(\mathbf{x}_i) + \sum_{d,j} \beta_{d,j} \left|\lambda_{d,j}\right|. \quad (4)$$

The novelty in problem $\mathcal{Q}_3(\boldsymbol{\lambda})$ involves having one normalization constant per example. In the development of the likelihood, a single logarithm $\ln Z_{\boldsymbol{\lambda}}$ is replaced by the sum $\frac{1}{m}\sum_i \ln Z_{\boldsymbol{\lambda}}(\mathbf{x}_i)$. To bound $L_{\tilde{p}}^{\boldsymbol{\beta}}(\boldsymbol{\lambda}') - L_{\tilde{p}}^{\boldsymbol{\beta}}(\boldsymbol{\lambda})$, the most difficult step is to bound:

$$\begin{aligned} \Delta &= \frac{1}{m}\sum_i \ln \frac{Z_{\boldsymbol{\lambda}'}(\mathbf{x}_i)}{Z_{\boldsymbol{\lambda}}(\mathbf{x}_i)} && (5) \\ &= \frac{1}{m}\sum_i \ln \left(\sum_{c\neq d} q_{\boldsymbol{\lambda}}(\mathbf{x}_i, c) + q_{\boldsymbol{\lambda}}(\mathbf{x}_i, d) e^{\delta f_{d,j}(\mathbf{x}_i, d)} \right) && (6) \\ &\leq \frac{1}{m}\sum_i \ln \Big(1 + (e^{\delta} - 1) q_{\boldsymbol{\lambda}}(\mathbf{x}_i, d) f_{d,j}(\mathbf{x}_i, d) \Big) && (7) \\ &\leq \ln \Big(1 + (e^{\delta} - 1) q'_{\boldsymbol{\lambda}}[f_{d,j}] \Big) && (8) \end{aligned}$$

where $q'_{\boldsymbol{\lambda}}[f_{d,j}] = \frac{1}{m}\sum_i q_{\boldsymbol{\lambda}}(\mathbf{x}_i, d) f_{d,j}(\mathbf{x}_i, d)$.

Eq.(7) uses

$$e^{\delta f_{d,j}(\mathbf{x}_i, d)} \leq 1 + (e^{\delta} - 1) f_{d,j}(\mathbf{x}_i, d)$$

for $f_{d,j}(\mathbf{x}_i, d) \in [0, 1]$ with equality if $f_{d,j}(\mathbf{x}_i, d) \in \{0, 1\}$. Eq.(8) relies on the convexity of the log function to apply Jensen's inequality.

We have established here a new bound on the decrease in the loss for L1-regularized *conditional* maxent models:

$$L_{\tilde{p}}^{\boldsymbol{\beta}}(\boldsymbol{\lambda}') - L_{\tilde{p}}^{\boldsymbol{\beta}}(\boldsymbol{\lambda}) \leq -\delta\tilde{p}[f_{d,j}] + \ln \Big(1 + (e^{\delta} - 1) q'_{\boldsymbol{\lambda}}[f_{d,j}] \Big) + \beta_{d,j} \left(|\lambda_{d,j} + \delta| - |\lambda_{d,j}| \right). \quad (9)$$

Because of its similarity to the standard SL1-Max bound (Eq.(9)), it allows a simple generalization of the SL1-Max algorithm to conditional distributions by replacing the $q_{\boldsymbol{\lambda}}[f_{d,j}]$ with $q'_{\boldsymbol{\lambda}}[f_{d,j}]$. Our experiments in the case of binary features show that the bound given in Eq.(3) is very tight. It can be used to obtain, in a closed form manner, a value for δ that is close to the optimum.

In the case of conditional maxent, it is instructive to compare this algorithm to Improved Iterative Scaling(IIS) [4], which also updates the parameters to maximize a bound on the decrease in the loss. First, the bounds are significantly different. The SL1-Max bound is tighter because it requires only a single λ_j to be modified at a time. Second, while both approaches support closed form solutions under specific conditions, these conditions are very different: the features must be binary in the case of SL1-Max, and they must add up to a constant value in the case of IIS. Finally, to our knowledge, there is no simple modification of IIS to handle L1-regularization.

4 Multi-label categorization

The fundamental modeling assumption we have made so far implies that each example i only carries a single label c_i. However, in many classification problems, a given input can correspond to multiple labels (multi-label).

For simplicity, we assume that there is no form of ranking or preference [1] among the labels. Our sample space covers input-code pairs $(\mathbf{x}, \mathbf{y}) \in X \times \{0,1\}^l$, where $\mathbf{y}$ is a binary output code.

Class-conditional distributions represent the easiest way to deal with multiple labels. We only focus on the estimation of $\tilde{p}_c$ and the fact that there are multiple labels that can be ignored. However, the final classification decision will require a multiplication by $\tilde{p}(c)$ that is not defined as a probability distribution because $\sum_c \tilde{p}(c) > 1$.

This section reviews two other techniques to handle multiple labels that only require minimum modifications of the algorithms proposed so far and show their limitations.

4.1 Duplicating training examples

Assume the training data is a set of input-code pairs $(\mathbf{x}_1, \mathbf{y}_1), \ldots, (\mathbf{x}_m, \mathbf{y}_m)$. Our goal is to project this training set in the smaller input-label sample space $\{\mathbf{x}_1, \ldots, \mathbf{x}_m\} \times \{1, \ldots, l\}$. For each training sample $(\mathbf{x}_i, \mathbf{y}_i)$ in the input-code space , we build K_i samples in the input-label space with $K_i = |\{1 \leq k \leq l : \mathbf{y}_i[k] = 1\}|$ The conditional probability function is:

$$\tilde{p}(\mathbf{y}[k] = 1 | \mathbf{x}_i) = \begin{cases} \frac{1}{K_i} & \text{if } \mathbf{y}_i[k] = 1 \\ 0 & \text{otherwise} \end{cases} \tag{10}$$

The problem with this approach is that the empirical distribution $\tilde{p}$ is reweighted to favor examples with multiple labels with $\tilde{p}(\mathbf{x}_i) = \frac{K_i}{\sum_i K_i}$ (we assume here that $\forall i, j : \mathbf{x}_i \neq \mathbf{x}_j$).

4.2 Using a non-class model

With an output code that represents l binary decisions, a trivial solution is to build l binary classifiers. This is typically what l *1-vs-other* classifiers do, and, in the case of maxent, one may think that the l class-conditional classifiers described in Section 3.2 represent such a solution.

However, the statistical reality is more complex; an independence assumption between each binary output is necessary:

$$p(\mathbf{y}|\mathbf{x}) = p(y^1, \ldots, y^l | \mathbf{x}) = \prod_c p(y^c | \mathbf{x}) \tag{11}$$

Under this assumption, one can estimate each $p(y^c|\mathbf{x})$ independently. For each class c, we introduce the distributions $p_i^c(y) = p^c(y|\mathbf{x}_i) = p(y^c = y|\mathbf{x}_i)$ where $y \in \{0,1\}$ is the the index for a secondary classification problem between examples belonging to class c and the other examples (which are said to be part of c

non-class). The independence assumption can be rewritten as $p_i(\mathbf{y}) = \prod_c p_i^c(y^c)$ so that the overall entropy can be decomposed into one entropy per class:

$$H(p_i) = -\sum_{\mathbf{y}\in\{0,1\}^l} p_i(\mathbf{y}) \ln p_i(\mathbf{y}) = \sum_c H(p_i^c) \tag{12}$$

As the entropy of each of the p^c distribution entropies can be maximized separately, we have l binary maxent models to estimate. Conditional maxent has previously been applied to binary "question answering" problems [18].

The framework defined by problems $\mathcal{P}_3$ and $\mathcal{Q}_3(\boldsymbol{\lambda})$ to produce the set of parameters $\boldsymbol{\lambda}$ can be applied here to the distribution p^c. The transformation of notation is described in the following table:

Distribution	p	p^c
Problem	$\mathcal{Q}_3(\boldsymbol{\lambda})$	$\mathcal{Q}_3(\boldsymbol{\lambda}^c)$
Parameters	$\boldsymbol{\lambda}$	$\boldsymbol{\lambda}^c$
Features	$\mathbf{f}(\mathbf{x}, c)$	$\mathbf{f}^c(\mathbf{x}, y)$
with	$c \in \{1, \ldots, l\}$	$y \in \{0, 1\}$

The exponential distribution that solves the dual problem $\mathcal{Q}_3(\boldsymbol{\lambda}^c)$ takes the form:

$$q_{\boldsymbol{\lambda}^c}(\mathbf{x}, y) = \frac{1}{Z_{\boldsymbol{\lambda}^c}(\mathbf{x})} e^{\boldsymbol{\lambda}^c \cdot \mathbf{f}^c(\mathbf{x}, y)}$$

The simplifying assumption of Eq.(2) becomes here:

$$f^c_{y',j}(\mathbf{x}, y) = \begin{cases} v_j(\mathbf{x}) & \text{if } y = y' \\ 0 & \text{otherwise} \end{cases} \tag{13}$$

Thus $\boldsymbol{\lambda}^c \cdot \mathbf{f}^c(\mathbf{x}, y) = \boldsymbol{\lambda}^c_y \cdot \mathbf{v}(\mathbf{x})$ and the probability of observing class c becomes

$$q_{\boldsymbol{\lambda}^c}(\mathbf{x}, y) = \sigma\left((\boldsymbol{\lambda}^c_1 - \boldsymbol{\lambda}^c_0) \cdot \mathbf{v}(\mathbf{x})\right) \tag{14}$$

where $\sigma(x) = \frac{1}{1+e^{-x}}$ is the sigmoid function. Note that for each feature $v_j(\mathbf{x})$, classifier c has two parameters: $\lambda^c_{1,j}$ used in the positive model and $\lambda^c_{0,j}$ used in the negative model. Given a test input $\mathbf{x}$, this approach can be used either to produce the code vector $\mathbf{y}$ such that $q^c_\lambda(\mathbf{x}, y^c) > 0.5$ or the top class $\operatorname{argmax}_c q^c_\lambda(\mathbf{x}, 1)$.

Eq.(14) suggests that, in the binary case, conditional maxent amounts to logistic regression. The use of L1-regularization in logistic regression was recently analyzed [15]. Another technique to optimize the logistic loss that relies on an implicit L1 regularization is AdaBoost with logistic loss [7], which also uses a sequential update procedure similar to SL1-Max.

While the independence assumption is not as straightforward, we can also transpose problem $\mathcal{P}_1$ to the distributions $p^c(\mathbf{x}, y) = p(\mathbf{x}_i, y^c = y)$ and obtain the convex dual $\mathcal{Q}_1(\boldsymbol{\lambda}^c)$.

5 Implementation: it's all about normalization

This section shows that from an implementation viewpoint, normalization is the main differentiator in the algorithms described in this paper.

We have already noted than SL1-Max is strikingly similar to AdaBoost, especially when AdaBoost is described within the Bregman distance framework [7]. As a matter of fact, unregularized conditional maxent was shown [13] to be equivalent to AdaBoost with the additional constraint of $\sum_c p_i(c) = 1$.

Assume Classes	Implement Classifiers	Maxent Distrib.	Optim. problem	Z?
Exclusive	**Tied**	Joint	$\mathcal{Q}_1(\boldsymbol{\lambda})$	No
		Conditional	$\mathcal{Q}_3(\boldsymbol{\lambda})$	No
	Separate	ClassCond	$\mathcal{Q}_2(\boldsymbol{\lambda}_c)$	Yes
Independent		Joint	$\mathcal{Q}_1(\boldsymbol{\lambda}^c)$	No
		Conditional	$\mathcal{Q}_3(\boldsymbol{\lambda}^c)$	No
		Adaboost		No

Table 1: Impact of the model on the implementation. The last column indicates that the computation of the normalization Z is required to perform classification.

Our implementation of SL1-Max capitalized on an earlier implementation of AdaBoost to include a normalization constant. The joint model requires normalization over all the classes and examples. The class-conditional model requires, for a given class, normalization over all the examples. The conditional model requires, for a given example, normalization over all the classes.

Our implementation of multiclass SL1-Max, which derives directly from Sections 3 and 4.2, can be interpreted as a combination of *1-vs-other* classifiers. We have not explored output codes or hierarchical structures, though multiclass SL1-Max offers a promising framework to explore these approaches, where the class-independent hypotheses would be much closer to reality. The most important difference between the various modeling hypotheses is whether the classifiers can be trained separately, or are tied by shared normalization constants. Training classifiers separately can be done on parallel computers or sequentially; in either case, the implementation is more memory efficient in a way which is critical when the number of classes is very large.

There is some merit in training the classifiers together: one can minimize a unique target function and monitor the training process in a variety of ways. The most common stopping criterion is when the classification error minimum is reached on validation data. When training classifiers separately, the absence of a single stopping criterion makes the process much harder to monitor. Table 1 summarizes the merit of each modeling assumption from an implementation viewpoint.

An implementation of SL1-Max that is optimized using *partial pricing* strategies [5] is provided in the (blanked out) software package. When classifiers can be trained separately, an SL1-Max binary classifier is just another *1-vs-other* classifier that can be used instead of an AdaBoost or a SVM classifier. On a given classification learning task, choosing between SL1-Max, AdaBoost and SVM can be done with a single switch, or by automatically using cross-validation data. Systematic experimental comparisons between the three approaches for large scale natural language understanding tasks [6] indicate that SL1-Max is the fastest approach on datasets larger than 100,000 examples, with state-of-the-art accuracy[2].

It would be informative to compare SL1-Max to algorithms considered as the state-of-the art for the estimation of parameters in conditional entropy models. They include Iterative Scaling algorithms, such as IIS [4] and Fast Iterative Scaling(FIS) [11], and gradient algorithms [14]. This study, which would be of considerable interest, is beyond the scope of this paper.

The two key factors that contribute to the remarkable learning speed of SL1-Max have not been, to our knowledge, applied to most algorithms in the iterative scaling family. First, SL1-Max is based on a

[2]They are only outperformed by SVMs with polynomial kernels, which are not a computationally practical because of the large number of support vectors

	Reuters	WebKB	SuperTags
Multi-label?	Yes	No	No
Train size	9603	3150	950028
Test size	3299	4199	46451
Num. of labels	90	4	4726
Num. of features	22758	25229	95516
Features/sample	126.7	129.1	18.8

Table 2: Key characteristics of the three datasets used in the experiments. The last line gives the average number of non-zero features per training vector.

pricing strategy: modify the single parameter which causes the greatest decrease in the objective function. The addition of *partial* pricing can make the search for the parameter considerably faster. Second, the L1 regularization adds some slack in the constraints and makes them easier to satisfy early in the optimization process.

Results on the WebKB text classification task show that SL1-Max takes less than 10 seconds to learn 3150 examples with 25,000 features, which compares favorably to more than 100 seconds when using FIS or IIS on a reduced set of 300 features [11]. (we assume comparable Pentium CPUs with a 2GHz clock).

6 Experiments and Discussions

The first two datasets are small enough to allow us to run the methods $\mathcal{Q}_1(\boldsymbol{\lambda})$ and $\mathcal{Q}_3(\boldsymbol{\lambda})$, which are compared to $\mathcal{Q}_2(\boldsymbol{\lambda}_c)$, $\mathcal{Q}_1(\boldsymbol{\lambda}^c)$, and $\mathcal{Q}_3(\boldsymbol{\lambda}^c)$. The Reuters-21758[3] dataset contains stories collected from Reuters newswire in 1987. We used the ModApte split between 9603 train stories and 3299 test stories. This is a multi-label problem, where the number of labels per story ranges from 0 to 15. The WebKB[4] dataset contains web pages gathered from university computer science departments. We selected the same categories as [16]: *student*, *faculty*, *courses* and *projects*. The 4199 samples are split between training and testing using a 4-fold cross-validation.

The third dataset, which demonstrates the scaling ability of SL1-Max, is much larger; and only the methods $\mathcal{Q}_2(\boldsymbol{\lambda}_c)$, $\mathcal{Q}_1(\boldsymbol{\lambda}^c)$, and $\mathcal{Q}_3(\boldsymbol{\lambda}^c)$ can be applied. It consists of a set of SuperTags. SuperTags are extensions of part-of-speech tags that encode morpho-syntactic constraints [3]) and are derived from the phrase-structure annotated Penn TreeBank. The characteristics of the three datasets are summarized in Table 2.

Table 3 compares the five different multiclass SL1-Max models considered in Sections 3 and 4.2. The SL1-Max regularization parameter is set to $\beta = 0.5$. AdaBoost with logistic loss and linear SVMs are provided as a baseline (note that our implementation of AdaBoost can be considered as a class-independent model with separate classifiers). The best error rate we obtain on WebKB (7.1%) and the best F-Measure we obtain on Reuters (86.7%) compare favorably to the literature [16, 12]. The training speed of Adaboost and SL1-Max are very similar and can be optimized using the same techniques. They are not reported here as a detailed comparison is reported elsewhere [5].

How good are class-conditional models? The most computationally efficient model is the *class-conditional* model $\mathcal{Q}_2(\boldsymbol{\lambda}_c)$. Table 4, which compares the computational efficiency of the $\mathcal{Q}_2(\boldsymbol{\lambda}_c)$, $\mathcal{Q}_1(\boldsymbol{\lambda}^c)$,

[3] http://www.daviddlewis.com/resources/testcollections/reuters21578

[4] http://www-2.cs.cmu.edu∼webkb

	Reuters	WebKB	SuperTags
AdaBoost	14.9/86.7	7.10	12.0
Linear SVM	15.3/86.6	7.57	
$\mathcal{Q}_1(\boldsymbol{\lambda})$	16.9/84.0	7.95	
$\mathcal{Q}_2(\boldsymbol{\lambda}_c)$	17.4/83.9	8.19	11.2
$\mathcal{Q}_3(\boldsymbol{\lambda})$	16.6/79.6	7.76	
$\mathcal{Q}_1(\boldsymbol{\lambda}^c)$	15.0/86.5	7.50	11.9
$\mathcal{Q}_3(\boldsymbol{\lambda}^c)$	14.1/86.4	7.45	11.1

Table 3: Error rates on the 3 datasets. For Reuters, which is multi-label, the first number is the top-class error rate (an example is considered an error if the highest scoring class given by the classifier is not part of the target labels) and the second number is the micro-averaged optimal F-measure.

	# parameters (thousands)		Train time (hours)	
β	0.5	0.9	0.5	0.9
$\mathcal{Q}_1(\boldsymbol{\lambda}^c)$	53	35	15.65	16.09
$\mathcal{Q}_2(\boldsymbol{\lambda}_c)$	53	34	14.92	7.58
$\mathcal{Q}_3(\boldsymbol{\lambda}^c)$	41	24	46.65	46.40

Table 4: Number of non-zero model parameters and training time for the SuperTags set as a function of the regularizer β and the optimization method. On this large set, β has little impact on accuracy, and mostly affects speed and sparsity.

and $\mathcal{Q}_3(\boldsymbol{\lambda}^c)$ models, shows that it has the smallest training time and the smallest number of parameters. However, its error rate on smaller dataset is higher due to the estimation of the Z normalization constant.

Class-exclusive vs. independent assumptions: The "Reuters" column of Table 3 indicates that making a class-exclusive assumption when it is not justified (e.g. the Reuters data is multi-label) leads to a significant loss in performance. By contrast, the class-independent assumption is never true, but a combination of binary maxent classifiers, which relies on this assumption, consistently improves performance. It also greatly improves training speed by allowing parallelization and a small memory footprint. A combination of binary classifier yields excellent *classification accuracy* regardless of the size of the problem. However, comparisons on a "Question Answering" problem [18] suggest that they may not perform well for *ranking* tasks. Future work on multi-label tasks will also assess the ranking performance with specific error measures [1].

Pros and cons of conditional models: For pure classification, the fully discriminant *conditional* models ($\mathcal{Q}_3(\boldsymbol{\lambda})$ and $\mathcal{Q}_3(\boldsymbol{\lambda}^c)$) yield the best results. This may justify the exclusive use of conditional models in all previous studies of multiclass maxent. Table 4 shows another reason to prefer conditional models: they are *sparser* and require fewer model parameters (since they only focus on performing class discrimination). Conditional models have two major drawbacks. First, as their outputs are normalized separately for each example, they tend to be poor confidence estimators. Metrics based on the comparison of the output to a varying threshold tend to fare poorly, For instance, in Table 3, the conditional model $\mathcal{Q}_3(\boldsymbol{\lambda})$ yield the lowest F-measure for Reuters (79.6). Second, as shown in Table 4, they can require more training time.

7 Conclusions

We have shown that a sequential maxent algorithm (SL1-Max) can be applied to many classification problems with performances which are comparable to Adaboost and SVMs. An important (and apparently under-appreciated) advantage of maxent for classification problems appears to be its remarkable flexibility in terms of modeling assumptions. In future work, this flexibility will be used to optimize maxent for problems where ranking or rejection performances are critical, and to which traditional classification methods are problematic to adapt.

References

[1] Fabio Aiolli and Alessandro Sperduti. Learning preferences for multiclass problems. In *Advances in Neural Information Processing Systems 17*. MIT Press, Cambridge, MA, 2005.

[2] E.L. Allwein, R.E. Schapire, and Y. Singer. Reducing Multiclass to Binary: A Unifying Approach for Margin Classifiers. In *Proceedings of ICML'00*, 2000.

[3] S. Bangalore and A. K. Joshi. Supertagging: An approach to almost parsing. *Computational Linguistics*, 25(2), 1999.

[4] Adam L. Berger, Stephen A. Della Pietra, and Vincent J. Della Pietra. A Maximum Entropy Approach to Natural Language Processing. *Computational Linguistics*, 22(1):39–71, 1996.

[5] blanked. Accelerating Sequential Learning with Partial Pricing. In *Submitted to ICML*, 2005.

[6] blanked. Scaling Large Margin Classifiers for Spoken Language Understanding. In *In review*, 2005.

[7] Michael Collins, Robert E. Schapire, and Yoram Singer. Logistic Regression, AdaBoost and Bregman Distances. In *Proceeddingd of COLT'00*, pages 158–169, Stanford, CA, 2000.

[8] Koby Crammer and Yoram Singer. On the Learnability and Design of Output Codes for Multiclass Problems. In *Proceedings of COLT'00*, pages 35–46, 2000.

[9] Miroslav Dudik, Steven Phillips, and Robert E. Schapire. Performance Guarantees for Regularized Maximum Entropy Density Estimation. In *Proceedings of COLT'04*, Banff, Canada, 2004. Springer Verlag.

[10] T. Jaakkola, M. Meila, and T. Jebara. Maximum Entropy Discrimination. In *Technical Report AITR-1668*. MIT, 1999.

[11] R. Jin, R. Yan, and J. Zhang. A Faster Iterative Scaling Algorithm For Conditional Exponential Model. In *The Twentieth International Conference on Machine Learning (ICML-2003)*, August 2003.

[12] Thorsten Joachims. Text categorization with support vector machines: learning with many relevant features. In *Proceedings of ECML-98*. Springer Verlag, 1998.

[13] G. Lebanon and J. Lafferty. Boosting and maximum likelihood for exponential models. In *Advances in Neural Information Processing Systems*, 2002.

[14] Robert Malouf. A comparison of algorithms for maximum entropy parameter estimation. In *Proceedings of CoNLL-2002*, pages 49–55. Taipei, Taiwan, 2002.

[15] Andrew Y. Ng. Feature selection, L1 vs. L2 regularization, and rotational invariance. In *Proceedings of ICML'04*, Banff, Canada, 2004. Omnipress.

[16] K. Nigam, J. Lafferty, and A. McCallum. Using maximum entropy for text classification. In *IJCAI-99 Workshop on Machine Learning for Information Filtering*, pages 61–67, 1999.

[17] Franz Josef Och and Hermann Ney. Discriminative Training and Maximum Entropy Models for Statistical Machine Translation. In *ACL 2002: Proc. of the 40th Annual Meeting of the Association for Computational Linguistics*, pages 295–302, July 2002.

[18] Deepak Ravichandran, Eduard Hovy, and Franz Josef Och. Statistical QA - Classifier vs Re-ranker: What's the difference? In *Proceedings of the ACL Workshop on Multilingual Summarization and Question Answering–Machine Learning and Beyond*. Sapporo,Japan, 2003.

[19] Robert E. Schapire and Yoram Singer. Improved boosting algorithms using confidence-rated predictions. *Machine Learning*, 37(3):297–336, 1999.

[20] Vladimir N. Vapnik. *Statistical Learning Theory*. John Wiley & Sons, 1998.