Multi-Class Confidence Weighted Algorithms

Koby Crammer[*] **Mark Dredze**[†] **Alex Kulesza**[*]
[*]Department of Computer and Information Science
University of Pennsylvania
Philadelphia, PA 19104
{crammer,kulesza}@cis.upenn.edu

[†]Human Language Technology Center of Excellence
Johns Hopkins University
Baltimore, MD 21211
mdredze@cs.jhu.edu

Abstract

The recently introduced online confidence-weighted (CW) learning algorithm for binary classification performs well on many binary NLP tasks. However, for multi-class problems CW learning updates and inference cannot be computed analytically or solved as convex optimization problems as they are in the binary case. We derive learning algorithms for the multi-class CW setting and provide extensive evaluation using nine NLP datasets, including three derived from the recently released New York Times corpus. Our best algorithm outperforms state-of-the-art online and batch methods on eight of the nine tasks. We also show that the confidence information maintained during learning yields useful probabilistic information at test time.

1 Introduction

Online learning algorithms such as the Perceptron process one example at a time, yielding simple and fast updates. They generally make few statistical assumptions about the data and are often used for natural language problems, where high dimensional feature representations, e.g., bags-of-words, demand efficiency. Most online algorithms, however, do not take into account the unique properties of such data, where many features are extremely rare and a few are very frequent.

Dredze, Crammer and Pereira (Dredze et al., 2008; Crammer et al., 2008) recently introduced confidence weighted (CW) online learning for binary prediction problems. CW learning explicitly models classifier weight uncertainty using a multivariate Gaussian distribution over weight vectors. The learner makes online updates based on its confidence in the current parameters, making larger changes in the weights of infrequently observed features. Empirical evaluation has demonstrated the advantages of this approach for a number of binary natural language processing (NLP) problems.

In this work, we develop and test multi-class confidence weighted online learning algorithms. For binary problems, the update rule is a simple convex optimization problem and inference is analytically computable. However, neither is true in the multi-class setting. We discuss several efficient online learning updates. These update rules can involve one, some, or all of the competing (incorrect) labels. We then perform an extensive evaluation of our algorithms using nine multi-class NLP classification problems, including three derived from the recently released New York Times corpus (Sandhaus, 2008). To the best of our knowledge, this is the first learning evaluation on these data. Our best algorithm outperforms state-of-the-art online algorithms and batch algorithms on eight of the nine datasets.

Surprisingly, we find that a simple algorithm in which updates consider only a single competing label often performs as well as or better than multi-constraint variants if it makes multiple passes over the data. This is especially promising for large datasets, where the efficiency of the update can be important. In the true online setting, where only one iteration is possible, multi-constraint algorithms yield better performance.

Finally, we demonstrate that the label distributions induced by the Gaussian parameter distributions resulting from our methods have interesting properties, such as higher entropy, compared to those from maximum entropy models. Improved label distributions may be useful in a variety of learning settings.

2 Problem Setting

In the multi-class setting, instances from an input space $\mathcal{X}$ take labels from a finite set $\mathcal{Y}$, $|\mathcal{Y}| = K$.

We use a standard approach (Collins, 2002) for generalizing binary classification and assume a feature function $\boldsymbol{f}(\boldsymbol{x}, y) \in \mathbb{R}^d$ mapping instances $\boldsymbol{x} \in \mathcal{X}$ and labels $y \in \mathcal{Y}$ into a common space.

We work in the online framework, where learning is performed in rounds. On each round the learner receives an input $\boldsymbol{x}_i$, makes a prediction $\hat{y}_i$ according to its current rule, and then learns the true label y_i. The learner uses the new example $(\boldsymbol{x}_i, y_i)$ to modify its prediction rule. Its goal is to minimize the total number of rounds with incorrect predictions, $|\{i : y_i \neq \hat{y}_i\}|$.

In this work we focus on linear models parameterized by weights $\boldsymbol{w}$ and utilizing prediction functions of the form $h_{\boldsymbol{w}}(\boldsymbol{x}) = \arg\max_z \boldsymbol{w} \cdot \boldsymbol{f}(\boldsymbol{x}, z)$. Note that since we can choose $\boldsymbol{f}(\boldsymbol{x}, y)$ to be the vectorized Cartesian product of an input feature function $\boldsymbol{g}(\boldsymbol{x})$ and y, this setup generalizes the use of unique weight vectors for each element of $\mathcal{Y}$.

3 Confidence Weighted Learning

Dredze, Crammer, and Pereira (2008) introduced online confidence weighted (CW) learning for binary classification, where $\mathcal{X} = \mathbb{R}^d$ and $\mathcal{Y} = \{\pm 1\}$. Rather than using a single parameter vector $\boldsymbol{w}$, CW maintains a distribution over parameters $\mathcal{N}(\boldsymbol{\mu}, \Sigma)$, where $\mathcal{N}(\boldsymbol{\mu}, \Sigma)$ the multivariate normal distribution with mean $\boldsymbol{\mu} \in \mathbb{R}^d$ and covariance matrix $\Sigma \in \mathbb{R}^{d \times d}$. Given an input instance $\boldsymbol{x}$, a Gibbs classifier draws a weight vector $\boldsymbol{w}$ from the distribution and then makes a prediction according to the sign of $\boldsymbol{w} \cdot \boldsymbol{x}$.

This prediction rule is robust if the example is classified correctly with high-probability, that is, for some confidence parameter $.5 \leq \eta < 1$, $\Pr_{\boldsymbol{w}}[y(\boldsymbol{w} \cdot \boldsymbol{x}) \geq 0] \geq \eta$. To learn a binary CW classifier in the online framework, the robustness property is enforced at each iteration while making a minimal update to the parameter distribution in the KL sense:

$$
\begin{aligned}
(\boldsymbol{\mu}_{i+1}, \Sigma_{i+1}) = & \\
& \arg\min_{\boldsymbol{\mu}, \Sigma} \mathrm{D}_{\mathrm{KL}}\left(\mathcal{N}(\boldsymbol{\mu}, \Sigma) \,\|\, \mathcal{N}(\boldsymbol{\mu}_i, \Sigma_i)\right) \\
& \text{s.t. } \Pr_{\boldsymbol{w}}[y_i(\boldsymbol{w} \cdot \boldsymbol{x}_i) \geq 0] \geq \eta
\end{aligned} \tag{1}
$$

Dredze et al. (2008) showed that this optimization can be solved in closed form, yielding the updates

$$\boldsymbol{\mu}_{i+1} = \boldsymbol{\mu}_i + \alpha_i \Sigma_i \boldsymbol{x}_i \tag{2}$$

$$\Sigma_{i+1} = \left(\Sigma_i^{-1} + \beta_i \boldsymbol{x}_i \boldsymbol{x}_i^T\right)^{-1} \tag{3}$$

for appropriate α_i and β_i.

For prediction, they use the Bayesian rule

$$\hat{y} = \arg\max_{z \in \{\pm 1\}} \Pr_{\boldsymbol{w} \sim \mathcal{N}(\boldsymbol{\mu}, \Sigma)}[z(\boldsymbol{x} \cdot \boldsymbol{w}) \geq 0] \ ,$$

which for binary labels is equivalent to using the mean parameters directly, $\hat{y} = \mathrm{sign}(\boldsymbol{\mu} \cdot \boldsymbol{x})$.

4 Multi-Class Confidence Weighted Learning

As in the binary case, we maintain a distribution over weight vectors $\boldsymbol{w} \sim \mathcal{N}(\boldsymbol{\mu}, \Sigma)$. Given an input instance $\boldsymbol{x}$, a Gibbs classifier draws a weight vector $\boldsymbol{w} \sim \mathcal{N}(\boldsymbol{\mu}, \Sigma)$ and then predicts the label with the maximal score, $\arg\max_z (\boldsymbol{w} \cdot \boldsymbol{f}(\boldsymbol{x}, z))$. As in the binary case, we use this prediction rule to define a robustness condition and corresponding learning updates.

We generalize the robustness condition used in Crammer et al. (2008). Following the update on round i, we require that the ith instance is correctly labeled with probability at least $\eta < 1$. Among the distributions that satisfy this condition, we choose the one that has the minimal KL distance from the current distribution. This yields the update

$$
\begin{aligned}
(\boldsymbol{\mu}_{i+1}, \Sigma_{i+1}) = & \\
& \arg\min_{\boldsymbol{\mu}, \Sigma} \mathrm{D}_{\mathrm{KL}}\left(\mathcal{N}(\boldsymbol{\mu}, \Sigma) \,\|\, \mathcal{N}(\boldsymbol{\mu}_i, \Sigma_i)\right) \\
& \text{s.t. } \Pr[y_i \,|\, \boldsymbol{x}_i, \boldsymbol{\mu}, \Sigma] \geq \eta \ ,
\end{aligned} \tag{4}
$$

where

$$
\begin{aligned}
& \Pr[y \,|\, \boldsymbol{x}, \boldsymbol{\mu}, \Sigma] = \\
& \Pr_{\boldsymbol{w} \sim \mathcal{N}(\boldsymbol{\mu}, \Sigma)}\left[y = \arg\max_{z \in \mathcal{Y}} (\boldsymbol{w} \cdot \boldsymbol{f}(\boldsymbol{x}, z))\right] \ .
\end{aligned}
$$

Due to the $\max$ operator in the constraint, this optimization is not convex when $K > 2$, and it does not permit a closed form solution. We therefore develop approximations that can be solved efficiently. We define the following set of events for a general input $\boldsymbol{x}$:

$$
\begin{aligned}
A_{r,s}(\boldsymbol{x}) &\stackrel{\text{def}}{=} \{\boldsymbol{w} : \boldsymbol{w} \cdot \boldsymbol{f}(\boldsymbol{x}, r) \geq \boldsymbol{w} \cdot \boldsymbol{f}(\boldsymbol{x}, s)\} \\
B_r(\boldsymbol{x}) &\stackrel{\text{def}}{=} \{\boldsymbol{w} : \boldsymbol{w} \cdot \boldsymbol{f}(\boldsymbol{x}, r) \geq \boldsymbol{w} \cdot \boldsymbol{f}(\boldsymbol{x}, s) \ \ \forall s\} \\
&= \bigcap_{s \neq r} A_{r,s}(\boldsymbol{x})
\end{aligned}
$$

We assume the probability that $\boldsymbol{w} \cdot \boldsymbol{f}(\boldsymbol{x}, r) = \boldsymbol{w} \cdot \boldsymbol{f}(\boldsymbol{x}, s)$ for some $s \neq r$ is zero, which

holds for non-trivial distribution parameters and feature vectors. We rewrite the prediction $\hat{y} = \arg\max_r \Pr[B_r(\boldsymbol{x})]$, and the constraint from Eq. (4) becomes

$$\Pr[B_{y_i}(\boldsymbol{x})] \geq \eta \,. \tag{5}$$

We focus now on approximating the event $B_{y_i}(\boldsymbol{x})$ in terms of events $A_{y_i,r}$. We rely on the fact that the level sets of $\Pr[A_{y_i,r}]$ are convex in $\boldsymbol{\mu}$ and Σ. This leads to convex constraints of the form $\Pr[A_{y_i,r}] \geq \gamma$.

Outer Bound: Since $B_r(\boldsymbol{x}) \subseteq A_{r,s}(\boldsymbol{x})$, it holds trivially that $\Pr[B_{y_i}(\boldsymbol{x})] \geq \eta \Rightarrow \Pr[A_{y_i,r}] \geq \eta, \forall r \neq y_i$. Thus we can replace the constraint $\Pr[B_{y_i}(\boldsymbol{x})] \geq \eta$ with $\Pr[A_{y_i,r}] \geq \eta$ to achieve an outer bound. We can simultaneously apply *all* of the pairwise constraints to achieve a tighter bound:

$$\Pr[A_{y_i,r}] \geq \eta \quad \forall r \neq y_i$$

This yields a convex approximation to Eq. (4) that may improve the objective value at the cost of violating the constraint. In the context of learning, this means that the new parameter distribution will be close to the previous one, but may not achieve the desired confidence on the current example. This makes the updates more conservative.

Inner Bound: We can also consider an inner bound. Note that $B_{y_i}(\boldsymbol{x})^c = (\cap_r A_{y_i,r}(\boldsymbol{x}))^c = \cup_r A_{y_i,r}(\boldsymbol{x})^c$, thus the constraint $\Pr[B_{y_i}(\boldsymbol{x})] \geq \eta$ is equivalent to

$$\Pr[\cup_r A_{y_i,r}(\boldsymbol{x})^c] \leq 1 - \eta \,,$$

and by the union bound, this follows whenever

$$\sum_r \Pr[A_{y_i,r}(\boldsymbol{x})^c] \leq 1 - \eta \,.$$

We can achieve this by choosing non-negative $\zeta_r \geq 0$, $\sum_r \zeta_r = 1$, and constraining

$$\Pr[A_{y_i,r}(\boldsymbol{x})] \geq 1 - (1-\eta)\,\zeta_r \quad \text{for} \quad r \neq y_i \,.$$

This formulation yields an inner bound on the original constraint, guaranteeing its satisfaction while possibly increasing the objective. In the context of learning, this is a more aggressive update, ensuring that the current example is robustly classified even if doing so requires a larger change to the parameter distribution.

Algorithm 1 Multi-Class CW Online Algorithm

Input: Confidence parameter η
Feature function $\boldsymbol{f}(\boldsymbol{x}, y) \in \mathbb{R}^d$
Initialize: $\boldsymbol{\mu}_1 = \mathbf{0}$, $\Sigma_1 = I$
for $i = 1, 2 \ldots$ **do**
Receive $\boldsymbol{x}_i \in \mathcal{X}$
Predict ranking of labels $\hat{y}_1, \hat{y}_2, \ldots$
Receive $y_i \in \mathcal{Y}$
Set $\boldsymbol{\mu}_{i+1}, \Sigma_{i+1}$ by approximately solving Eq. (4) using one of the following:
Single-constraint update (Sec. 5.1)
Exact many-constraint update (Sec. 5.2)
Seq. many-constraint approx. (Sec. 5.2)
Parallel many-constraint approx. (Sec. 5.2)
end for
Output: Final $\boldsymbol{\mu}$ and Σ

Discussion: The two approximations are quite similar in form. Both replace the constraint $\Pr[B_{y_i}(\boldsymbol{x})] \geq \eta$ with one or more constraints of the form

$$\Pr[A_{y_i,r}(\boldsymbol{x})] \geq \eta_r \,. \tag{6}$$

To achieve an outer bound we choose $\eta_r = \eta$ for any set of $r \neq y_i$. To achieve an inner bound we use all $K-1$ possible constraints, setting $\eta_r = 1 - (1-\eta)\,\zeta_r$ for suitable ζ_r. A simple choice is $\zeta_r = 1/(K-1)$.

In practice, η is a learning parameter whose value will be optimized for each task. In this case, the outer bound (when all constraints are included) and inner bound (when $\zeta_r = 1/(K-1)$) can be seen as equivalent, since for any fixed value of $\eta^{(\text{in})}$ for the inner bound we can choose

$$\eta^{(\text{out})} = 1 - \frac{1-\eta^{(\text{in})}}{K-1} \,,$$

for the outer bound and the resulting η_r will be equal. By optimizing η we automatically tune the approximation to achieve the best compromise between the inner and outer bounds. In the following, we will therefore assume $\eta_r = \eta$.

5 Online Updates

Our algorithms are online and process examples one at a time. Pseudo-code for our approach is given in algorithm 1. We approximate the prediction step by ranking each label y according to the score given by the mean weight vector, $\boldsymbol{\mu} \cdot \boldsymbol{f}(\boldsymbol{x}_i, y)$. Although this approach is Bayes optimal for binary problems (Dredze et al., 2008),

it is an approximation in general. We note that more accurate inference can be performed in the multi-class case by sampling weight vectors from the distribution $\mathcal{N}(\boldsymbol{\mu}, \Sigma)$ or selecting labels sensitive to the variance of prediction; however, in our experiments this did not improve performance and required significantly more computation. We therefore proceed with this simple and effective approximation.

The update rule is given by an approximation of the type described in Sec. 4. All that remains is to choose the constraint set and solve the optimization efficiently. We discuss several schemes for minimizing KL divergence subject to one or more constraints of the form $\Pr[A_{y_i,r}(\boldsymbol{x})] \geq \eta$. We start with a single constraint.

5.1 Single-Constraint Updates

The simplest approach is to select the single constraint $\Pr[A_{y_i,r}(\boldsymbol{x})] \geq \eta$ corresponding to the highest-ranking label $r \neq y_i$. This ensures that, following the update, the true label is more likely to be predicted than the label that was its closest competitor. We refer to this as the $k = 1$ update.

Whenever we have only a single constraint, we can reduce the optimization to one of the closed-form CW updates used for binary classification. Several have been proposed, based on linear approximations (Dredze et al., 2008) and exact formulations (Crammer et al., 2008). For simplicity, we use the `Variance` method from Dredze et al. (2008), which did well in our initial evaluations. This method leads to the following update rules. Note that in practice Σ is projected to a diagonal matrix as part of the update; this is necessary due to the large number of features that we use.

$$\boldsymbol{\mu}_{i+1} = \boldsymbol{\mu}_i + \alpha_i \Sigma_i \boldsymbol{g}_{i,y_i,r} \tag{7}$$

$$\Sigma_{i+1} = \left(\Sigma_i^{-1} + 2\alpha_i \phi \boldsymbol{g}_{i,y_i,r} \boldsymbol{g}_{i,y_i,r}^\top\right)^{-1} \tag{8}$$

$$\boldsymbol{g}_{i,y_i,r} = \boldsymbol{f}(\boldsymbol{x}_i, y_i) - \boldsymbol{f}(\boldsymbol{x}_i, r) \quad \phi = \Phi^{-1}(\eta)$$

The scale α_i is given by $\max(\gamma_i, 0)$, where γ_i is equal to

$$\frac{-(1 + 2\phi m_i) + \sqrt{(1 + 2\phi m_i)^2 - 8\phi(m_i - \phi v_i)}}{4\phi v_i}$$

and

$$m_i = \boldsymbol{\mu}_i \cdot \boldsymbol{g}_{i,y_i,r} \qquad v_i = \boldsymbol{g}_{i,y_i,r}^\top \Sigma_i \boldsymbol{g}_{i,y_i,r} \,.$$

These rules derive directly from Dredze et al. (2008) or Figure 1 in Crammer et al. (2008); we simply substitute $y_i = 1$ and $\boldsymbol{x}_i = \boldsymbol{g}_{i,y_i,r}$.

5.2 Many-Constraints Updates

A more accurate approximation can be obtained by selecting multiple constraints. Analogously, we choose the $k \leq K-1$ constraints corresponding to the labels $r_1, \ldots, r_k \neq y_i$ that achieve the highest predicted ranks. The resulting optimization is convex and can be solved by a standard Hildreth-like algorithm (Censor & Zenios, 1997). We refer to this update as `Exact`. However, `Exact` is expensive to compute, and tends to over-fit in practice (Sec. 6.2). We propose several approximate alternatives.

Sequential Update: The Hildreth algorithm iterates over the constraints, updating with respect to each until convergence is reached. We approximate this solution by making only a single pass:

- Set $\boldsymbol{\mu}_{i,0} = \boldsymbol{\mu}_i$ and $\Sigma_{i,0} = \Sigma_i$.
- For $j = 1, \ldots, k$, set $(\boldsymbol{\mu}_{i,j}, \Sigma_{i,j})$ to the solution of the following optimization:

$$\min_{\boldsymbol{\mu},\Sigma} \mathrm{D}_{\mathrm{KL}}\left(\mathcal{N}(\boldsymbol{\mu}, \Sigma) \,\|\, \mathcal{N}\left(\boldsymbol{\mu}_{i,j-1}, \Sigma_{i,j-1}\right)\right)$$
$$\text{s.t. } \Pr\left[A_{y_i,r_j}(\boldsymbol{x})\right] \geq \eta$$

- Set $\boldsymbol{\mu}_{i+1} = \boldsymbol{\mu}_{i,k}$ and $\Sigma_{i+1} = \Sigma_{i,k}$.

Parallel Update: As an alternative to the Hildreth algorithm, we consider the simultaneous algorithm of Iusem and Pierro (1987), which finds an exact solution by iterating over the constraints in parallel. As above, we approximate the exact solution by performing only one iteration. The process is as follows.

- For $j = 1, \ldots, k$, set $(\boldsymbol{\mu}_{i,j}, \Sigma_{i,j})$ to the solution of the following optimization:

$$\min_{\boldsymbol{\mu},\Sigma} \mathrm{D}_{\mathrm{KL}}\left(\mathcal{N}(\boldsymbol{\mu}, \Sigma) \,\|\, \mathcal{N}(\boldsymbol{\mu}_i, \Sigma_i)\right)$$
$$\text{s.t. } \Pr\left[A_{y_i,r_j}(\boldsymbol{x})\right] \geq \eta$$

- Let $\boldsymbol{\lambda}$ be a vector, $\lambda_j \geq 0$, $\sum_j \lambda_j = 1$.
- Set $\boldsymbol{\mu}_{i+1} = \sum_j \lambda_j \boldsymbol{\mu}_{i,j}$, $\Sigma_{i+1}^{-1} = \sum_j \lambda_j \Sigma_{i,j}^{-1}$.

In practice we set $\lambda_j = 1/k$ for all j.

6 Experiments

6.1 Datasets

Following the approach of Dredze et al. (2008), we evaluate using five natural language classification tasks over nine datasets that vary in difficulty, size, and label/feature counts. See Table 1 for an overview. Brief descriptions follow.

Task	*Instances*	*Features*	*Labels*	*Bal.*
20 News	18,828	252,115	20	Y
Amazon 7	13,580	686,724	7	Y
Amazon 3	7,000	494,481	3	Y
Enron A	3,000	13,559	10	N
Enron B	3,000	18,065	10	N
NYTD	10,000	108,671	26	N
NYTO	10,000	108,671	34	N
NYTS	10,000	114,316	20	N
Reuters	4,000	23,699	4	N

Table 1: A summary of the nine datasets, including the number of instances, features, and labels, and whether the numbers of examples in each class are balanced.

Task	*Sequential*	*Parallel*	*Exact*
20 News	**92.16**	91.41	88.08
Amazon 7	77.98	**78.35**	77.92
Amazon 3	93.54	**93.81**	93.00
Enron A	**82.40**	81.30	77.07
Enron B	71.80	**72.13**	68.00
NYTD	**83.43**	81.43	80.92
NYTO	**82.02**	78.67	80.60
NYTS	52.96	**54.78**	51.62
Reuters	93.60	**93.97**	93.47

Table 2: A comparison of $k = \infty$ updates. While the two approximations (sequential and parallel) are roughly the same, the exact solution over-fits.

Amazon Amazon product reviews. Using the data of Dredze et al. (2008), we created two domain classification datasets from seven product types (apparel, books, dvds, electronics, kitchen, music, video). *Amazon 7* includes all seven product types and *Amazon 3* includes books, dvds, and music. Feature extraction follows Blitzer et al. (2007) (bigram features and counts).

20 Newsgroups Approximately 20,000 newsgroup messages, partitioned across 20 different newsgroups.[1] This dataset is a popular choice for binary and multi-class text classification as well as unsupervised clustering. We represent each message as a binary bag-of-words.

Enron Automatic sorting of emails into folders.[2] We selected two users with many email folders and messages: `farmer-d` (*Enron A*) and `kaminski-v` (*Enron B*). We used the ten largest folders for each user, excluding non-archival email folders such as "inbox," "deleted items," and "discussion threads." Emails were represented as binary bags-of-words with stop-words removed.

NY Times To the best of our knowledge we are the first to evaluate machine learning methods on the New York Times corpus. The corpus contains 1.8 million articles that appeared from 1987 to 2007 (Sandhaus, 2008). In addition to being one of the largest collections of raw news text, it is possibly the largest collection of publicly released annotated news text, and therefore an ideal corpus for large scale NLP tasks. Among other annotations, each article is labeled with the desk that produced the story (Financial, Sports, etc.) (*NYTD*), the online section to which the article was posted (*NYTO*), and the section in which the article was printed (*NYTS*). Articles were represented as bags-of-words with feature counts (stop-words removed).

Reuters Over 800,000 manually categorized newswire stories (RCV1-v2/ LYRL2004). Each article contains one or more labels describing its general topic, industry, and region. We performed topic classification with the four general topics: corporate, economic, government, and markets. Details on document preparation and feature extraction are given by Lewis et al. (2004).

6.2 Evaluations

We first set out to compare the three update approaches proposed in Sec. 5.2: an exact solution and two approximations (sequential and parallel). Results (Table 2) show that the two approximations perform similarly. For every experiment the CW parameter η and the number of iterations (up to 10) were optimized using a single randomized iteration. However, sequential converges faster, needing an average of 4.33 iterations compared to 7.56 for parallel across all datasets. Therefore, we select sequential for our subsequent experiments.

The exact method performs poorly, displaying the lowest performance on almost every dataset. This is unsurprising given similar results for binary CW learning Dredze et al. (2008), where exact updates were shown to over-fit but converged after a single iteration of training. Similarly, our exact implementation converges after an average of 1.25 iterations, much faster than either of the approximations. However, this rapid convergence appears to come at the expense of accuracy. Fig. 1 shows the accuracy on Amazon 7 test data after each training iteration. While both sequential and parallel improve with several iterations, exact de-

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

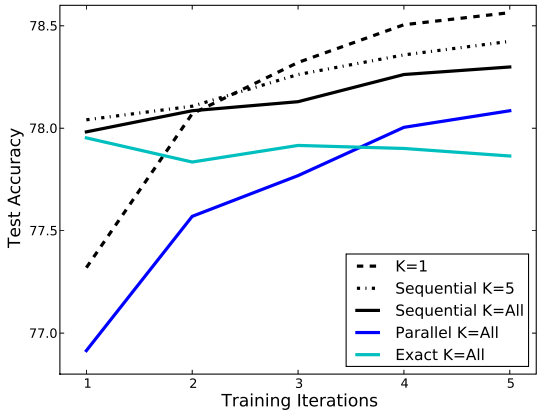

Figure 1: Accuracy on test data after each iteration on the Amazon 7 dataset.

grades after the first iteration, suggesting that it may over-fit to the training data. The approximations appear to smooth learning and produce better performance in the long run.

6.3 Relaxing Many-Constraints

While enforcing many constraints may seem optimal, there are advantages to pruning the constraints as well. It may be time consuming to enforce dozens or hundreds of constraints for tasks with many labels. Structured prediction tasks often involve exponentially many constraints, making pruning mandatory. Furthermore, many real world datasets, especially in NLP, are noisy, and enforcing too many constraints can lead to over-fitting. Therefore, we consider the impact of reducing the constraint set in terms of both reducing run-time and improving accuracy.

We compared using all constraints ($k = \infty$) with using 5 constraints ($k = 5$) for the sequential update method (Table 3). First, we observe that $k = 5$ performs better than $k = \infty$ on nearly every dataset: fewer constraints help avoid over-fitting and once again, simpler is better. Additionally, $k = 5$ converges faster than $k = \infty$ in an average of 2.22 iterations compared with 4.33 iterations. Therefore, reducing the number of constraints improves both speed and accuracy. In comparing $k = 5$ with the further reduced $k = 1$ results, we observe the latter improves on seven of the nine methods. This surprising result suggests that CW learning can perform well even without considering more than a single constraint per example. However, $k = 1$ exceeds the performance of multiple constraints only through repeated training iterations. $k = 5$ CW learning converges faster — 2.22 iterations compared with 6.67 for $k = 1$ — a desirable property in many resource restricted settings. (In the true online setting, only a single iteration may be possible.) Fig. 1 plots the performance of $k = 1$ and $k = 5$ CW on test data after each training iteration. While $k = 1$ does better in the long run, it lags behind $k = 5$ for several iterations. In fact, after a single training iteration, $k = 5$ outperforms $k = 1$ on eight out of nine datasets. Thus, there is again a tradeoff between faster convergence ($k = 5$) and increased accuracy ($k = 1$). While the $k = 5$ update takes longer per iteration, the time required for the approximate solutions grows only linearly in the number of constraints. The evaluation in Fig. 1 required 3 seconds for the first iteration of $k = 1$, 10 seconds for $k = 5$ and 11 seconds for one iteration of all 7 constraints. These differences are insignificant compared to the cost of performing multiple iterations over a large dataset. We note that, while both approximate methods took about the same amount of time, the exact solution took over 4 minutes for its first iteration.

Finally, we compare CW methods with several baselines in Table 3. Online baselines include Top-1 Perceptron (Collins, 2002), Top-1 Passive-Aggressive (PA), and k-best PA (Crammer & Singer, 2003; McDonald et al., 2004). Batch algorithms include Maximum Entropy (default configuration in McCallum (2002)) and support vector machines (LibSVM (Chang & Lin, 2001) for one-against-one classification and multiclass (MC) (Crammer & Singer, 2001)). Classifier parameters (C for PA/SVM and maxent's Gaussian prior) and number of iterations (up to 10) for the online methods were optimized using a single randomized iteration. On eight of the nine datasets, CW improves over all baselines. In general, CW provides faster and more accurate multiclass predictions.

7 Error and Probabilistic Output

Our focus so far has been on accuracy and speed. However, there are other important considerations for selecting learning algorithms. Maximum entropy and other probabilistic classification algorithms are sometimes favored for their probability scores, which can be useful for integration with other learning systems. However, practition-

		PA		CW			SVM		
Task	*Perceptron*	*K=1*	*K=5*	*K=1*	*K=5*	*K=∞*	*1 vs. 1*	*MC*	*Maxent*
20 News	81.07	88.59	88.60	$**$**92.90**	$**$92.78	$**$92.16	85.18	90.33	88.94
Amazon 7	74.93	76.55	76.72	$**$**78.70**	$**$78.04	$**$77.98	75.11	76.60	76.40
Amazon 3	92.26	92.47	93.29	†94.01	$**$**94.29**	93.54	92.83	93.60	93.60
Enron A	74.23	79.27	80.77	††**83.83**	†82.23	†82.40	80.23	82.60	82.80
Enron B	66.30	69.93	68.90	$**$**73.57**	$**$72.27	$**$71.80	65.97	71.87	69.47
NYTD	80.67	83.12	81.31	$**$**84.57**	$*$83.94	83.43	82.95	82.00	83.54
NYTO	78.47	81.93	81.22	†**82.72**	†82.55	82.02	82.13	81.01	82.53
NYTS	50.80	**56.19**	55.04	54.67	54.26	52.96	55.81	56.74	53.82
Reuters	92.10	93.12	93.30	93.60	**93.67**	93.60	92.97	93.32	93.40

Table 3: A comparison of CW learning ($k = 1, 5, \infty$ with sequential updates) with several baseline algorithms. CW learning achieves the best performance eight out of nine times. Statistical significance (McNemar) is measured against *all* baselines ($*$ indicates 0.05 and $**$ 0.001) or against online baselines († indicates 0.05 and †† 0.001).

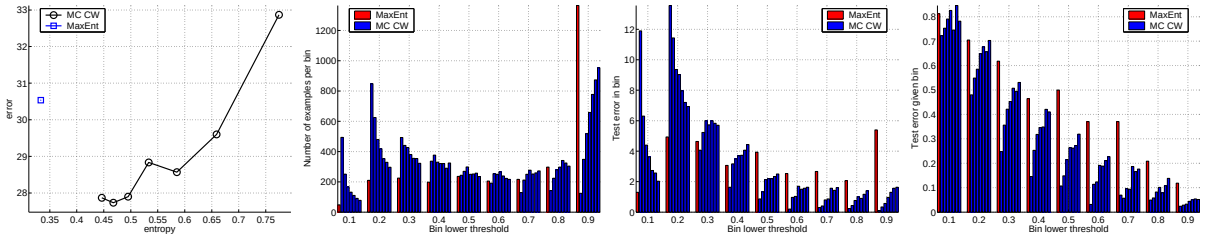

Figure 2: **First panel:** Error versus prediction entropy on Enron B. As CW converges (right to left) error and entropy are reduced. **Second panel:** Number of test examples per prediction probability bin. The red bars correspond to maxent and the blue bars to CW, with increasing numbers of epochs from left to right. **Third panel:** The contribution of each bin to the total test error. **Fourth panel:** Test error conditioned on prediction probability.

ers have observed that maxent probabilities can have low entropy and be unreliable for estimating prediction confidence (Malkin & Bilmes, 2008). Since CW also produces label probabilities — and does so in a conceptually distinct way — we investigate in this section some empirical properties of the label distributions induced by CW's parameter distributions and compare them with those of maxent.

We trained maxent and CW $k = 1$ classifiers on the Enron B dataset, optimizing parameters as before (maxent's Gaussian prior and CW's η). We estimated the label distributions from our CW classifiers after each iteration and on every test example $\boldsymbol{x}$ by Gibbs sampling weight vectors $\boldsymbol{w} \sim \mathcal{N}(\boldsymbol{\mu}, \Sigma)$, and for each label y counting the fraction of weight vectors for which $y = \arg\max_z \boldsymbol{w} \cdot \boldsymbol{f}(\boldsymbol{x}, z)$. Normalizing these counts yields the label distributions $\Pr[y|\boldsymbol{x}]$. We denote by $\hat{y}$ the predicted label for a given $\boldsymbol{x}$, and refer to $\Pr[\hat{y}|\boldsymbol{x}]$ as the prediction probability.

The leftmost panel of Fig. 2 plots each method's prediction error against the normalized entropy of the label distribution $-\left(\frac{1}{m}\sum_i \sum_z \Pr[z|\boldsymbol{x}_i] \log\left(\Pr[z|\boldsymbol{x}_i]\right)\right) / \log(K)$. Each CW iteration (moving from right to left in the plot) reduces both error and entropy. From our maxent results we make the common observation that maxent distributions have (ironically) low entropy. In contrast, while CW accuracy exceeds maxent after its second iteration, normalized entropy remains high. Higher entropy suggests a distribution over labels that is less peaked and potentially more informative than those from maxent. We found that the average probability assigned to a correct prediction was 0.75 for CW versus 0.83 for maxent and for an incorrect prediction was 0.44 for CW versus 0.56 for maxent.

Next, we investigate how these probabilities relate to label accuracy. In the remaining panels, we binned examples according to their prediction probabilities $\Pr[\hat{y}|\boldsymbol{x}] = \max_y \Pr[y|\boldsymbol{x}]$. The second panel of Fig. 2 shows the numbers of test examples with $\Pr[\hat{y}|\boldsymbol{x}] \in [\theta, \theta + 0.1)$ for $\theta = 0.1, 0.2 \ldots 0.9$. (Note that since there are 10

classes in this problem, we must have $\Pr[\hat{y}|\boldsymbol{x}] \geq 0.1$.) The red (leftmost) bar corresponds to the maximum entropy classifier, and the blue bars correspond, from left to right, to CW after each successive training epoch.

From the plot we observe that the maxent classifier assigns prediction probability greater than 0.9 to more than 1,200 test examples out of 3,000. Only 50 examples predicted by maxent fall in the lowest bin, and the rest of examples are distributed nearly uniformly across the remaining bins. The large number of examples with very high prediction probability explains the low entropy observed for the maximum entropy classifier.

In contrast, the CW classifier shows the opposite behavior after one epoch of training (the leftmost blue bar), assigning low prediction probability (less than 0.3) to more than 1,200 examples and prediction probability of at least 0.9 to only 100 examples. As CW makes additional passes over the training data, its prediction confidence increases and shifts toward more peaked distributions. After seven epochs fewer than 100 examples have low prediction probability and almost 1,000 have high prediction probability. Nonetheless, we note that this distribution is still less skewed than that of the maximum entropy classifier.

Given the frequency of high probability maxent predictions, it seems likely that many of the high probability maxent labels will be wrong. This is demonstrated in the third panel, which shows the contribution of each bin to the total test error. Each bar reflects the number of mistakes per bin divided by the size of the complete test set (3,000). Thus, the sum of the heights of the corresponding bars in each bin is proportional to test error. Much of the error of the maxent classifier comes not only from the low-probability bins, due to their inaccuracy, but also from the highest bin, due to its very high population. In contrast, the CW classifiers see very little error contribution from the high-probability bins. As training progresses, we see again that the CW classifiers move in the direction of the maxent classifier but remain essentially unimodal.

Finally, the rightmost panel shows the conditional test error given bin identity, or the fraction of test examples from each bin where the prediction was incorrect. This is the pointwise ratio between corresponding values of the previous two histograms. For both methods, there is a monotonically decreasing trend in error as prediction probability increases; that is, the higher the value of the prediction probability, the more likely that the prediction it provides is correct. As CW is trained, we see an increase in the conditional test error, yet the overall error decreases (not shown). This suggests that as CW is trained and its overall accuracy improves, there are more examples with high prediction probability, and the cost for this is a relative increase in the conditional test error per bin. The maxent classifier produces an extremely large number of test examples with very high prediction probabilities, which yields relatively high conditional test error. In nearly all cases, the conditional error values for the CW classifiers are smaller than the corresponding values for maximum entropy. These observations suggest that CW assigns probabilities more conservatively than maxent does, and that the (fewer) high confidence predictions it makes are of a higher quality. This is a potentially valuable property, e.g., for system combination.

8 Conclusion

We have proposed a series of approximations for multi-class confidence weighted learning, where the simple analytical solutions of binary CW learning do not apply. Our best CW method outperforms online and batch baselines on eight of nine NLP tasks, and is highly scalable due to the use of a single optimization constraint. Alternatively, our multi-constraint algorithms provide improved performance for systems that can afford only a single pass through the training data, as in the true online setting. This result stands in contrast to previously observed behaviors in non-CW settings (McDonald et al., 2004). Additionally, we found improvements in both label entropy and accuracy as compared to a maximum entropy classifier. We plan to extend these ideas to structured problems with exponentially many labels and develop methods that efficiently model label correlations. An implementation of CW multi-class algorithms is available upon request from the authors.

References

Blitzer, J., Dredze, M., & Pereira, F. (2007). Biographies, bollywood, boom-boxes and blenders: Domain adaptation for sentiment classification. *Association for Computational Linguistics (ACL)*.

Censor, Y., & Zenios, S. (1997). *Parallel optimization: Theory, algorithms, and applications*. Oxford University Press, New York, NY, USA.

Chang, C.-C., & Lin, C.-J. (2001). *LIBSVM: a library for support vector machines*. Software available at `http://www.csie.ntu.edu.tw/~cjlin/libsvm`.

Collins, M. (2002). Discriminative training methods for hidden markov models: Theory and experiments with perceptron algorithms. *Empirical Methods in Natural Language Processing (EMNLP)*.

Crammer, K., Dredze, M., & Pereira, F. (2008). Exact confidence-weighted learning. *Advances in Neural Information Processing Systems 22*.

Crammer, K., & Singer, Y. (2001). On the algorithmic implementation of multiclass kernel-based vector machines. *Jornal of Machine Learning Research*, *2*, 265–292.

Crammer, K., & Singer, Y. (2003). Ultraconservative online algorithms for multiclass problems. *Jornal of Machine Learning Research (JMLR)*, *3*, 951–991.

Dredze, M., Crammer, K., & Pereira, F. (2008). Confidence-weighted linear classification. *International Conference on Machine Learning (ICML)*.

Iusem, A., & Pierro, A. D. (1987). A simultaneous iterative method for computing projections on polyhedra. *SIAM J. Control and Optimization*, *25*.

Lewis, D. D., Yang, Y., Rose, T. G., & Li, F. (2004). Rcv1: A new benchmark collection for text categorization research. *Journal of Machine Learning Research (JMLR)*, *5*, 361–397.

Malkin, J., & Bilmes, J. (2008). Ratio semi-definite classifiers. *IEEE Int. Conf. on Acoustics, Speech, and Signal Processing*.

McCallum, A. (2002). MALLET: A machine learning for language toolkit. `http://mallet.cs.umass.edu`.

McDonald, R., Crammer, K., & Pereira, F. (2004). Large margin online learning algorithms for scalable structured classification. *NIPS Workshop on Structured Outputs*.

Sandhaus, E. (2008). The new york times annotated corpus. Linguistic Data Consortium, Philadelphia.