Learning a Product of Experts with Elitist Lasso

Mengqiu Wang
Computer Science Department
Stanford University
Stanford, CA 94305, USA
mengqiu@cs.stanford.edu

Christopher D. Manning
Computer Science Department
Stanford University
Stanford, CA 94305, USA
manning@cs.stanford.edu

Abstract

Discriminative models such as logistic regression profit from the ability to incorporate arbitrary rich features; however, complex dependencies among overlapping features can often result in weight undertraining. One popular method that attempts to mitigate this problem is logarithmic opinion pools (LOP), which is a specialized form of product of experts model that automatically adjusts the weighting among experts. A major problem with LOP is that it requires significant amounts of domain expertise in designing effective experts. We propose a novel method that learns to induce experts — not just the weighting between them — through the use of a mixed $\ell_2\ell_1$ norm as previously seen in *elitist lasso*. Unlike its more popular sibling $\ell_1\ell_2$ norm (used in group lasso), which seeks feature sparsity at the group-level, $\ell_2\ell_1$ norm encourages sparsity within feature groups. We demonstrate how this property can be leveraged as a competition mechanism to induce groups of diverse experts, and introduce a new formulation of elitist lasso MaxEnt in the FOBOS optimization framework (Duchi and Singer, 2009). Results on Named Entity Recognition task suggest that this method gives consistent improvements over a standard logistic regression model, and is more effective than conventional induction schemes for experts.

1 Introduction

Conditionally trained discriminative models like logistic regression (a.k.a., MaxEnt models) gain power from their ability to incorporate a large number of arbitrarily overlapping features. But such models also exhibit complex feature dependencies and therefore are susceptible to the problem of weight undertraining — the contributions of certain features are overlooked because of features they co-occur with (Sutton et al., 2007; Hinton et al., 2012).

One popular method that attempts to mitigate this problem is logarithmic opinion pools (LOP) (Heskes, 1998; Smith et al., 2005; Sutton et al., 2007). LOP works in a similar fashion to a product of experts model, in which a number of experts are individually assembled first, and then their predictions are combined multiplicatively to create an ensemble model. Experts are generated by first partitioning the feature space into subsets, then an independent model (expert) is trained on each subset of features (Sutton et al., 2007). Under the assumption that strongly correlated features are partitioned into separate subsets, this method effectively forces the models to learn how to make predictions with each subset of the features independently. It also helps in scenarios where some strong features stop other features from getting weight, a problem known as feature co-adaptation. The theoretical justification for favoring a product ensemble over an additive ensemble is that inference with a product of log-linear models is significantly easier than inference with a sum. Sutton et al. (2007) also suggests that product ensembles give better results than additive mixtures on sequence labeling tasks.

Smith et al. (2005) and Sutton et al. (2007) showed that the quality of experts and diversity among them have a direct impact on the final performance of the ensemble. Designing effective and diverse experts was left as an art, and requires

International Joint Conference on Natural Language Processing, pages 525–533,
Nagoya, Japan, 14-18 October 2013.

a great deal of domain expertise.

In this paper, we directly learn to induce diverse experts by using a mixed $\ell_2\ell_1$ norm known as *elitist lasso* in the context of generalized linear models (Kowalski and Torrésani, 2009). We design a novel competition mechanism to encourage experts to use non-overlapping feature sets, effectively learning a partition of the feature space. Efficient optimization of a maximum conditional likelihood objective with respect to $\ell_2\ell_1$ norm is non-trivial and has not been previously studied. We propose a novel formulation to incorporate $\ell_2\ell_1$ norm in the FOBOS optimization framework (Duchi and Singer, 2009). Experiments on Named Entity Recognition task suggest that our proposed method gives consistent improvements over a baseline MaxEnt model and conventional LOP experts. In particular, our method gives the most improvements in recognizing entity types for out-of-vocabulary words.

2 MaxEnt Model

Given an input sequence $\mathbf{x}$ (e.g., words in a sentence), a MaxEnt model defines the conditional probability of an output variable y (e.g., named-entity tag of a word) as follows:

$$P(y = v_i|\mathbf{x}) = \frac{\exp\left(\sum_{k=1}^{K} \theta_k(v_i) f_k(\mathbf{x})\right)}{\sum_{j=1}^{V} \exp\left(\sum_{k=1}^{K} \theta_k(v_j) f_k(\mathbf{x})\right)}$$

where $f_k(\mathbf{x})$ is the kth input feature, K is the total number of input features, and $\theta_k(y)$ is the weight parameter associated with feature f_k for output class v_i. We denote the output classes of y to be $\{v_1, v_2, \cdots, v_V\}$, where V is the total number of output classes.

It is helpful to consider the connection between a MaxEnt model and a Linear Network model commonly seen in the neural network literature, by re-expressing this objective using a *softmax* activation function. The softmax function is defined as:

$$\text{softmax}\left(q(y)\right) = \frac{\exp\left(q(y)\right)}{\sum_{y'} \exp\left(q(y')\right)}$$

in our case $q(y) = \sum_{k=1}^{K} \theta_k(y) f_k(\mathbf{x})$, and we have:

$$P(y|\mathbf{x}) = \text{softmax}\left(\sum_{k=1}^{K} \theta_k(y) f_k(\mathbf{x})\right)$$

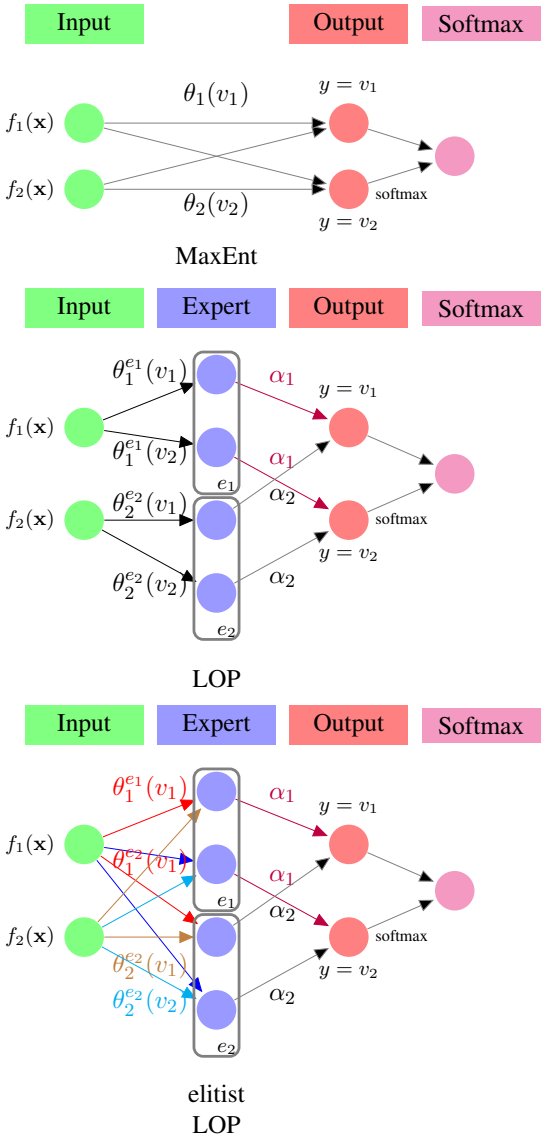

Figure 1: The top part shows a regular MaxEnt model with 2 features and 2 output classes. The middle part shows a LOP with two experts. α_1 and α_2 are the expert mixing coefficients to be learned in the LOP model. The bottom part shows an elitist LOP expert induction model with two experts. For each feature weight parameter θ, the superscript denotes which feature group it belongs to (e.g., θ^{e_1} belongs to expert e_1); the subscript denotes which input feature it belongs to (e.g., θ_1 means it applies to input feature $f_1(\mathbf{x})$); and the value in parentheses denotes which output class this feature is associated with (e.g., $\theta(v_1)$ applies to output class $y = v_1$). Features that belong to the same group are shown in the same color.

We visualize this using a neural network style diagram in the top part of Figure 1. In this diagram, the value at each node is computed as the sum of all incoming edge weights (parameters in the model) multiplied by the values of nodes on the originating end of edges. In this example, $K = 2$ and we have two input features ($f_1(\mathbf{x})$ and $f_2(\mathbf{x})$); there are two possible value assignments for y (v_1 and v_2). Correspondingly, there are four weight parameters ($\theta_i(v_j)$, for $i = 1, 2, j = 1, 2$) as shown by the directed edge going from "Input" layer nodes to "Output" layer nodes. We apply a softmax function to the "Output" layer to produce the probability assignment of each output class.

3 Weight Undertraining

The problem of weight undertraining occurs in discriminatively trained classifiers when some strongly indicative features occur during training but not at test time. Because of the presence of such strong features during training, weaker features that occur at both training and test time do not receive enough weight, and thus impede the classifier from reaching its full potential at test time. This phenomenon is also known as feature co-adaptation (Hinton et al., 2012). Sutton et al. (2007) gave an excellent demonstration of this problem using a synthetic logistic regression model. In their example, the output function is the softmax of the linear sum of a set of weights x_1 to x_n, where each of the x_i acts as a "weak" predictor. A "strong" feature which takes the form of $x' = \sum_{i=1}^{n} x_i + \mathcal{N}$ (where $\mathcal{N}$ is an added Gaussian noise) is then added to the classifier during training but removed at test time, which simulates a feature co-adaptation scenario. Simulation results show that classifier accuracy can drop by as much as 40% as a result of weight undertraining.

This problem is difficult to combat because we do not know a priori what set of features will be present at test time. However, a number of methods including feature bagging (Sutton et al., 2007), random feature dropout (Hinton et al., 2012), and logarithmic opinion pools (Heskes, 1998; Smith et al., 2005) have been introduced as ways to alleviate weight undertraining.

4 Logarithmic Opinion Pools

The idea behind logarithmic opinion pools (LOP) is that instead of training one classifier, we can train a set of classifiers with non-overlapping or competing feature sets. At test time, we can combine these classifiers and average their results. By creating a diverse set of classifiers and training each of them separately, we can potentially reduce the effect of weight undertraining. For example, when two strongly correlated features are partitioned into different classifiers, they do not overshadow each other. Unlike voting or feature bagging where an additive ensemble of experts is used, LOP derives a joint model by taking a product of experts.[1]

The conditional probability of a LOP ensemble can be expressed as:

$$P_{LOP}(y|\mathbf{x}) = \frac{\prod_{j=1}^{n} [P_j(y|\mathbf{x})]^{\alpha_j}}{\sum_{y'} \prod_{j=1}^{n} [P_j(y|\mathbf{x})]^{\alpha_j}}$$

$$P_j(y|\mathbf{x}) = \text{softmax}\left(\sum_{k=1}^{K} \theta_k^{e_j}(y) f_k(\mathbf{x})\right)$$

where n is the total number of experts, $P_j(y|\mathbf{x})$ is the probability assignment of expert j, and α_j is the mixing coefficient for this expert.

The experts are typically selected so that they capture different signals from the training data (e.g., via feature subsetting). Each expert is trained separately, and their model parameters $\theta_k^{e_j}(y)$ are fixed during LOP combination.

Early work on LOPs either fixed the expert mixing coefficients to some uniformly assigned value (Hinton, 1999), or used some arbitrary handpicked values (Sutton et al., 2007). In both of these two cases, no new parameter values need to be learned, therefore no extra training is required.

Smith et al. (2005) proposed to learn the weights by maximizing log-likelihood of the ensemble product model, and showed that the learned weights work better than uniformly assigned values.

We can visualize the LOP again using a network diagram, as shown in the middle part of Figure 1. Two experts are shown in this diagram, each represented by a plate in the "Expert" layer. This example illustrates an expert generation scheme by partitioning the features into non-overlapping subsets – expert e_1 has feature function $f_1(\mathbf{x})$ while expert e_2 has feature function $f_2(\mathbf{x})$. The weights of the two experts are pre-trained and remain fixed. Training the LOP model involves only learning the α mixing parameters.

[1] A product in the raw probability space is equivalent to a sum in the logarithmic space, and hence the name "logarithmic" opinion pools.

5 Automatic Induction of Experts

A major concern in adopting existing LOP methods is that there are no straightforward guidelines on how to create experts. Different feature partition schemes can result in dramatically different performance (Smith et al., 2005). A successfully designed LOP expert ensemble typically involves non-trivial engineering effort and a significant amount of domain expertise. Furthermore, the improvements shown in one application domain by a well-designed feature partition scheme do not necessarily carry over to other domains.

Therefore, it is our goal to investigate and search for effective methods to automatically induce LOP experts. We would like to find a set of experts that are diverse (i.e., having non-overlapping feature sets) and accurate. The main idea here is that we can train multiple copies of MaxEnt models together to optimize training likelihood, but at the same time we encourage the MaxEnt models to differ as much as possible in their choice of features to employ. To meet this end, we use a $\ell_2\ell_1$ mixed norm to regularize a LOP, and denote the new model as *elitist LOP*.

5.1 Elitist LOP

Before we introduce the $\ell_2\ell_1$ norm, which has not been frequently used in NLP research, let us revisit a closely related but much better known sibling – the $\ell_1\ell_2$ norm used in group lasso.

For a set of model parameters $\boldsymbol{\theta}$, assume we have some feature grouping that partitions the features into G sub-groups. For simplicity, let us further assume that each feature group has the same number of features (M). We denote the index of each feature by g (for "group") and m (for "member"), and use $\boldsymbol{\theta}_{\hat{g}}$ to denote the feature group of index g.

The $\ell_1\ell_2$ mixed norm used in group lasso takes the following form:

$$\|\boldsymbol{\theta}\|_{1,2} = \left(\sum_{g=1}^{G}\left(\sum_{m=1}^{M}|\theta_{g,m}|^2\right)^{1/2}\right) = \|(\|\boldsymbol{\theta}_{\hat{1}}\|_2, \cdots, \|\boldsymbol{\theta}_{\hat{G}}\|_2)\|_1$$

This norm applies ℓ_2 regularization to each group of features, but enforces ℓ_1 regularization at the group level. Since ℓ_1 norm encourages sparse solutions, therefore the effect of the mixed norm is to sparsify features at the group level by eliminating a whole group of features at a time.

It is easy to see how it relates to the $\ell_2\ell_1$ mixed norm, which is given as:

$$\|\boldsymbol{\theta}\|_{2,1} = \left(\sum_{g=1}^{G}\left(\sum_{m=1}^{M}|\theta_{g,m}|\right)^{2}\right)^{1/2} = \|(\|\boldsymbol{\theta}_{\hat{1}}\|_1, \cdots, \|\boldsymbol{\theta}_{\hat{G}}\|_1)\|_2$$

Like $\ell_1\ell_2$, this is also a group-sensitive sparsity-inducing norm, but instead of encouraging sparsity across feature groups, it promotes sparsity within each feature group.

We illustrate how we apply $\ell_2\ell_1$ norm to induce diverse experts in the bottom diagram in Figure 1. Similar to the case of LOP, we create two "Expert" layer plates, one for each expert (e_1 and e_2). But unlike traditional LOP, where each expert only gets a subset of the input features (e.g., $f_1(\mathbf{x}) \mapsto e_1$ and $f_2(\mathbf{x}) \mapsto e_2$), each input feature is fully connected to each expert plate (i.e., there are four edges with the same expert superscript from the "Input" layer going into each plate in the "Expert" layer, shown in different colors).

We group every pair of feature weights that originate from the same input feature and end at the same output class into a separate feature group (e.g., $\theta_1^{e_1}(v_1)$ and $\theta_1^{e_2}(v_1)$ belongs to one group, while $\theta_2^{e_1}(v_1)$ and $\theta_2^{e_2}(v_1)$ belongs to another group, etc.). In total we have four feature groups, each shown in a different color.

When we apply $\ell_2\ell_1$ norm to this feature grouping while learning the parameter weights in the LOP model, the regularization will push most of the features that belong to the same group towards 0. But since we are optimizing a likelihood objective, if a particular input feature is indeed predictive, the model will also try to assign some weights to the weight parameters associated with this feature. These two opposite forces create a novel competition mechanism among features that belong to the same group.

This competition mechanism encourages the experts to use different sets of non-overlapping features, thus achieving the diversity effect we want. But because we are also optimizing the likelihood of the ensemble, each expert is encouraged to use parameters that are as predictive as possible. Learning all the experts together can also be seen as a form of *feature sharing*, as suggested by Ando and Zhang (2005) for multi-task learning.

5.2 FOBOS with $\ell_2\ell_1$ norm

In our automatic expert induction model with $\ell_2\ell_1$ norm, the overall objective is given as:

$$\mathcal{L} = \operatorname*{argmin}_{\boldsymbol{\theta}} \Bigg(-\log\Big(\frac{\prod_{j=1}^{n}[P_j(y|\mathbf{x})]^{\alpha_j}}{\sum_{y'}\prod_{j=1}^{n}[P_j(y|\mathbf{x})]^{\alpha_j}}\Big) + \lambda\|(\|\boldsymbol{\theta}_{\hat{1}}\|_1, \cdots, \|\boldsymbol{\theta}_{\hat{G}}\|_1)\|_2 \Bigg)$$

where λ is a parameter that controls the regularization strength.

The feature groups are given as:

$$\begin{aligned}
\boldsymbol{\theta}_{\hat{1}} &= \{\theta_1^{e_1}(v_1), \theta_1^{e_2}(v_1), \cdots, \theta_1^{e^n}(v_1)\} \\
\boldsymbol{\theta}_{\hat{2}} &= \{\theta_2^{e_1}(v_1), \theta_2^{e_2}(v_1), \cdots, \theta_2^{e^n}(v_1)\} \\
&\cdots \\
\boldsymbol{\theta}_{\hat{K}} &= \{\theta_K^{e_1}(v_1), \theta_K^{e_2}(v_1), \cdots, \theta_K^{e^n}(v_1)\} \\
\boldsymbol{\theta}_{\hat{K+1}} &= \{\theta_1^{e_1}(v_2), \theta_1^{e_2}(v_2), \cdots, \theta_1^{e^n}(v_2)\} \\
\boldsymbol{\theta}_{\hat{K+2}} &= \{\theta_2^{e_1}(v_2), \theta_2^{e_2}(v_2), \cdots, \theta_2^{e^n}(v_2)\} \\
&\cdots \\
\boldsymbol{\theta}_{\hat{G}} &= \{\theta_K^{e_1}(v_V), \theta_K^{e_2}(v_V), \cdots, \theta_K^{e^n}(v_V)\}
\end{aligned}$$

There is a total of $G = K \times V$ feature groups, and each feature group has $M = n$ features.

The key difference here from LOP is that the $\boldsymbol{\theta}$ parameters are not pre-trained and fixed, but jointly learned with respect to $\ell_2\ell_1$ regularization. In the LOP case, learning the mixing coefficients α_i ($i = \{1, \ldots, n\}$) can be done by taking the gradients of α_i with respect to the training objective and directly plugging it into a gradient-based optimization framework, such as the *limited memory variable metric* (LMVM) used by Smith et al. (2005) or Stochastic Gradient Descent. However, learning the $\boldsymbol{\theta}$ parameters in our case is not as straightforward, since the objective is non-differentiable at many points due to the $\ell_2\ell_1$ norm. We cannot simply throw in an off-the-shelf gradient-based optimizer.

To alleviate the problem of non-differentiability, the recently proposed FOrward-Backward Splitting (FOBOS) optimization framework (Duchi and Singer, 2009) comes to mind. FOBOS is a (sub-)gradient-based framework that solves the optimization problem iteratively in two steps: in each iteration, we first take a sub-gradient step, and then take an analytical minimization step that incorporates the regularization term, which can often be solved with a closed form solution. Formally, each iteration at timestamp t in FOBOS consists of the following two steps:

$$\boldsymbol{\theta}_{t+\frac{1}{2}} = \boldsymbol{\theta}_t - \eta_t \boldsymbol{g}_t \tag{1}$$

$$\boldsymbol{\theta}_{t+1} = \operatorname*{argmin}_{\boldsymbol{\theta}} \left\{ \frac{1}{2}\|\boldsymbol{\theta} - \boldsymbol{\theta}_{t+\frac{1}{2}}\|^2 + \eta_t \varphi(\boldsymbol{\theta}) \right\} \tag{2}$$

$\boldsymbol{g}_t$ is the subgradient of $-log(P_{LOP}(y|\boldsymbol{x}))$ at θ_t. Step (1) is just a regular gradient-based step, where η_t is the step size and $\boldsymbol{g}_t$ is the sub-gradient vector of parameter vector $\boldsymbol{\theta}$. Step (2) finds a new vector that stays close to the interim vector after step (1), but also has a low penalty score according to the regularization term $\varphi(\boldsymbol{\theta})$ (in our case, $\varphi(\boldsymbol{\theta}) = \|\boldsymbol{\theta}\|_{2,1}$). In the case of ℓ_1 and ℓ_2 regularization, each feature θ_i is independent in step (2), and thus can be solved separately.[2]

Duchi and Singer (2009) gave closed form solutions for solving the minimization problem in step (2) for ℓ_1, ℓ_2, and $\ell_1\ell_2$ norms, but no past research has demonstrated how to solve it for $\ell_2\ell_1$ norm. We leverage the results given by Kowalski and Torrésani (2009) for elitist lasso, and show that parameter $\theta_{t+1,g,m}$ (the m^{th} feature of group g at timestamp $t+1$) can be solved analytically as follows:

$$\theta_{t+1,g,m} = \text{sign}(\theta_{t+\frac{1}{2},g,m})\left(|\theta_{t+\frac{1}{2},g,m}| - \tau_g\right)^+$$

$$\tau_g = \frac{\lambda'}{1+\lambda' M_{\hat{g}}(\lambda')}\left\|x_{t+\frac{1}{2},g,1:M_{\hat{g}}(\lambda')}\right\|_1$$

$\text{sign}(x)$ is the mathematical sign function. For $x \in \mathcal{R}$, we have $x^+ = x$ if $x \geq 0$ and $x^+ = 0$ if $x \leq 0$. $\lambda' = \eta_t\lambda$ is the regularization weight adjusted by the current step size.

Let $\bar{\boldsymbol{\theta}}_{\hat{g}}$ denote a new vector that holds the positive absolute value of $\boldsymbol{\theta}_{\hat{g}}$, sorted in descending order, i.e., $\bar{\theta}_{g,1} \geq \bar{\theta}_{g,2} \geq \cdots \bar{\theta}_{g,M}$. $M_{\hat{g}}(\lambda')$ is a positive integer given by the following definition:

$$\bar{\theta}_{g,M_{\hat{g}}(\lambda')+1} \leq \sum_{m=1}^{M_{\hat{g}}(\lambda')+1}\left(\bar{\theta}_{g,m} - \bar{\theta}_{g,M_{\hat{g}}(\lambda')+1}\right)$$

and

$$\bar{\theta}_{g,M_{\hat{g}}(\lambda')} > \lambda' \sum_{m=1}^{M_{\hat{g}}(\lambda')}\left(\bar{\theta}_{g,m} - \bar{\theta}_{g,M_{\hat{g}}(\lambda')}\right)$$

[2] This property turns out to be of great importance in real-world large scale scenarios, since it allows the optimization of high-dimensional feature vectors to be parallelized.

$x_{t+\frac{1}{2},g,1:M_{\hat{g}}(\lambda')}$ is then the subset of features from 1 to $M_{\hat{g}}(\lambda')$ – in descending order of their absolute value – in group g at timestamp $t + \frac{1}{2}$.

One problem with finding the exact solution of elitist lasso as illustrated above is that computing the value $M_{\hat{g}}(\lambda')$ involves sorting the parameter vector $\boldsymbol{\theta}$, which is a $O(n \log n)$ operation. When the number of features within each group is large, this could be prohibitively expensive. An approximate solution simply replaces $M_{\hat{g}}(\lambda')$ with the size of the group M. Kowalski and Torrésani (2009) found that approximated elitist lasso works nearly as well as the exact version, but is faster and easier to implement. We adopt this approximation in our experiments.

With this, we have described a general-purpose method for solving any convex optimization problem with $\ell_2\ell_1$ norm.

6 Experiments

We evaluate the performance of our proposed method on the task of Named Entity Recognition (NER). The first dataset we evaluate on is the standard CoNLL-2003 English shared task benchmark dataset (Sang and Meulder, 2003), which is a collection of documents from Reuters newswire articles, annotated with four entity types: *Person*, *Location*, *Organization*, and *Miscellaneous*. We adopt the `BIO2` annotation standard. Beginning and intermediate positions of an entity are marked with *B-* and *I-* tags, and non-entities with *O* tag. The training set contains 204K words (14K sentences), the development set contains 51K words (3.3K sentences), and the test set contains 46K words (3.5K sentences). The second dataset is the MUC6/7 set, which contains 255K words for training and 59K words for testing. The MUC data is annotated with 7 entity types. It is missing the *Miscellaneous* entity type, but includes 4 additional entity types that do not occur in CoNLL-2003: *Date*, *Time*, *Money*, and *Percent*.

We used a comprehensive set of features from Finkel et al. (2005) for training the MaxEnt model. A total number of 437905 features were generated on the CoNLL-2003 training dataset. The most important features are listed in Table 1.

6.1 Experimental Setup

We tuned the regularization parameters in the ℓ_1, ℓ_2 and $\ell_1\ell_2$ norms on the development dataset via grid search. We used a tuning procedure similar to the one used in Turian et al. (2010) where we evaluate results on the development set after each optimization iteration, and terminate the procedure after not observing a performance increase on the development set in 25 continuous iterations. We found 150 to be a good λ value for ℓ_2, 0.1 for ℓ_1 and 0.1 for $\ell_2\ell_1$. Similarly, we tuned the number of experts for both LOP baselines and the elitist LOP induction scheme. All model parameters were initialized to a random value in $[-0.1, 0.1]$.

– The word, word shape (e.g., whether capitalized, numeric, etc.), and letter n-grams (up to 6gram) at current position
– The word and word shape of the previous and next position
– Previous word shape in conjunction with current word shape
– Disjunctive word set of the previous and next 4 positions
– Capitalization pattern in a 3 word window
– The current word matched against a list of name titles (e.g., Mr., Mrs.)
– Previous two words in conjunction with the word shape of the previous word

Table 1: MaxEnt features.

Results are measured in precision (P), recall (R) and F_1 score. Statistical significance is measured using the paired bootstrap resampling method (Efron and Tibshirani, 1993). We compare our results against a MaxEnt baseline model with both ℓ_1 and ℓ_2 regularization, as well as an automatic expert induction model with ℓ_1 norm.

Two commonly used LOP expert induction schemes were also compared. In the first scheme (*LOP random*), experts are constructed by randomly partitioning the features into n subsets, where n is the number of experts. Each expert therefore has $\frac{1}{n}$th of the full feature set. The second scheme (*LOP sequential*) works in a similar way, but instead of random partition, we create feature subsets sequentially and preserve the relative order of the features. We also list results reported in Smith et al. (2005) on the same dataset for comparison. They experimented with three manually crafted expert induction schemes (*Smith et al. 05* {*simple;positional;label*}) for LOP with a Conditional Random Field (CRF), which is a more powerful sequence model than our MaxEnt baseline.

6.2 Main Results

Results on the CoNLL and MUC dataset are shown in Table 2. The proposed automatic expert induction scheme (row *elitist LOP*) gives consistent and statistically significant improvements over the MaxEnt baselines on both CoNLL and MUC test sets.

In particular, on the CoNLL test set, we ob-

	Models	# of experts	Dev Set Precision	Dev Set Recall	Dev Set F_1	Test Set Precision	Test Set Recall	Test Set F_1
CoNLL	MaxEnt-ℓ_2	1	88.46	87.73	88.09	81.42	81.64	81.53
	MaxEnt-ℓ_1	1	88.46	89.63	89.04	82.20	84.24	83.21†
	LOP random	2	88.75	90.79	89.76	82.73	85.84	84.25†‡
	LOP sequential	2	88.68	90.79	89.72	83.03	86.03	84.50†‡
	LOP-ℓ_1	5	88.76	90.41	89.58	82.90	85.94	84.40†‡
	elitist LOP	3	89.65	91.08	90.36	83.96	86.67	85.29†‡
	Smith et al. 05 simple	2	-	-	90.26	-	-	84.22
	Smith et al. 05 positional	3	-	-	90.35	-	-	84.71
	Smith et al. 05 label	5	-	-	89.30	-	-	83.27
MUC	MaxEnt-ℓ_2	1	91.47	86.20	88.76	89.06	80.44	84.53
	MaxEnt-ℓ_1	1	92.56	85.50	88.89	89.61	79.09	84.02
	LOP random	2	93.77	84.95	89.14	91.05	78.89	84.54
	LOP sequential	2	94.34	84.18	88.97	91.71	77.42	83.96
	LOP-ℓ_1	5	93.25	86.09	89.53	90.70	79.57	84.78‡
	elitist LOP	3	93.47	86.48	89.84	91.22	80.15	85.33†‡

Table 2: Results of NER on CoNLL and MUC dataset. † and ‡ indicate statistically significantly better F_1 scores than the MaxEnt-ℓ_2 and MaxEnt-ℓ_1 baselines, respectively.

	Models	IV Precision	IV Recall	IV F_1	OOV Precision	OOV Recall	OOV F_1
CoNLL	MaxEnt-ℓ_2	89.21	86.81	88.00	85.69	80.61	83.08
	elitist LOP	90.25	89.85	90.05†	86.30	86.67	86.49†
MUC	MaxEnt-ℓ_2	90.21	80.88	85.29	84.90	78.80	81.74
	elitist LOP	92.27	80.02	85.71	87.49	80.62	83.91†

Table 3: OOV and IV results breakdown. † indicates statistically significantly better F_1 scores than the MaxEnt-ℓ_2 baseline.

serve an improvement of 3.7% absolute F_1 over *Maxent-*ℓ_2. The gains are particularly large in recall (by 5% absolute score), although there is also a 2.5% improvement in precision. The two conventional LOP induction schemes also show significant improvements over the MaxEnt baselines on this dataset, with the sequential feature partition scheme working slightly better than the random feature partition. However, elitist LOP outperforms the two LOP schemes by as much as 1% in absolute F_1, and also gives better results than manually crafted experts for CRFs in Smith et al. (2005).

On the MUC test set, while elitist LOP still outperforms the MaxEnt baselines, the LOP schemes do not help or in some cases hurt performance. This suggests the lack of robustness of LOP which was discussed in Section 5. The automatically learned LOP experts are more robust on this data set, and gives a 0.8% improvement measured in absolute F_1 score over the *MaxEnt-*ℓ_2 baseline.

The elitist LOP model is more expressive than MaxEnt since it has more parameters to be combined. To understand whether performance gain is obtained by the expressiveness or by the regularization of $\ell_2\ell_1$ norm, we compare against an elitist LOP model with just ℓ_1 regularization (*LOP-*ℓ_1). Results show that the $\ell_2\ell_1$ norm is a better regularizer for avoiding overfitting with these models. We also see a significant gain in *LOP-*ℓ_1 over *MaxEnt-*ℓ_1, suggesting that the expressiveness of the model also helps.

6.3 Out-of-vocabulary vs. In-vocabulary

To further understand *how* our method improves over the MaxEnt baseline, we break down the test results into two subsets: words that were seen in the training dataset (in-vocabulary, or IV), versus words that were not (out-of-vocabulary, or OOV). We removed the entity boundary tags (e.g. "I-ORG" becomes "ORG") so that each word token is evaluated separately. Dividing IV and OOV subsets can be done by checking each word against a lexicon compiled from training dataset.

If our automatic expert induction LOP method does successfully mitigate the weight undertraining problem, we would expect to see an improvement in OOV recognition. The result of this analysis is shown in Table 3.

On the CoNLL dataset, our method gives significant improvements over MaxEnt in both OOV and IV category. In particular, improvement from

OOV subset (3.4% in F_1) is larger than the improvement from IV subset (2% in F_1). This matches that our hypothesis that our method mitigates the weight undertraining problem and thus gives a stronger boost in OOV recognition.

This effect is even more pronounced on the MUC dataset. The improvement in IV subset in this case is actually quite modest (0.5%), but we get a significant 2.2% improvement from the OOV subset.

7 Related Work

Beyond the several aforementioned works that address the issue of weight undertraining, another recent work that looks at this problem is Wang and Manning (2013). They examined the objective function of dropout training prescribed by Hinton et al. (2012), and proposed an approximation of dropout by directly sampling from a Gaussian approximation. The resulting algorithm is orders of magnitude faster than the iterative algorithm given by Hinton et al. (2012). It will be interesting to compare our proposed method with this method in the future.

The use of group-sparsity norms in NLP research is relatively rare. Martins et al. (2011) proposed the use of structure-inducing norms, in particular $\ell_1\ell_2$ norm (group lasso), for learning the structure of classifiers. A key observation is that during testing, most of the runtime is consumed in the feature instantiation stage. Since $\ell_1\ell_2$ norm discards whole groups of features at a time, there are fewer feature templates that need to be instantiated, therefore it could give a significant runtime speedup.

Our use of $\ell_2\ell_1$ norm takes a different flavor in that we explore the internal structure of the model. There is, however, one recent paper that also makes use of $\ell_2\ell_1$ norm for a similar reason. Das and Smith (2012) employed $\ell_2\ell_1$ norm for learning sparse structure in a network formed by lexical predicates and semantic frames. Their results show that $\ell_2\ell_1$ norm yields much more compact models than ℓ_1 or ℓ_2 norms, with superior performance in learning to expand lexicons.

However, the optimization problem in Das and Smith (2012) was much simpler since all parameters can only take on positive values, and therefore they could directly use a gradient-descent method with specialized edge condition checks. In our work, we have shown a general-purpose method in the FOBOS framework for optimizing any convex function with $\ell_2\ell_1$ norm.

A related ℓ_1-ℓ_2 mixed norm is called *elastic net* (Zou and Hastie, 2005). It was proposed to overcome a problem of lasso (i.e., ℓ_1 regularization), which occurs when there is a group of correlated predictors, and lasso tends to pick one and ignore all the others. Elastic net takes the form of a sum of a ℓ_1 and a ℓ_2 norm. It was used in Lavergne et al. (2010) for learning large-scale Conditional Random Fields.

Another popular approach that also explores diversity in model predictions is system combination, which is related to bagging predictors (Breiman, 1996). For example, Sang et al. (2000) reported that by combing systems using different tagging representation and diverse classifier models (e.g., support vector machine, logistic regression, naive Bayes), they were able to achieve significantly higher accuracy than any individual classifier alone. This approach has also been applied extensively in Machine Translation (DeNero et al., 2010) and Speech Recognition (Mikolov et al., 2011).

8 Conclusion

Our results show that previous automatic expert definition methods used in logarithmic opinion pools lack robustness across different tasks and data sets. Instead of manually defining good (i.e., diverse) experts, we demonstrated an effective way to induce experts automatically, by using a sparsity-inducing mixed $\ell_1\ell_2$ norm inspired by elitist lasso. We proposed a novel formulation of the optimization problem with $\ell_2\ell_1$ norm in the FOBOS framework. Our method gives consistent and significant improvements over a MaxEnt baseline with fine-tuned ℓ_2 regularization on two NER datasets. The gains are most evident in recognizing entity types for out-of-vocabulary words.

Acknowledgments

The authors would like to thank Rob Voigt and the three anonymous reviewers for their valuable comments and suggestions. We gratefully acknowledge the support of the DARPA Broad Operational Language Translation (BOLT) program through IBM. Any opinions, findings, and conclusion or recommendations expressed in this material are those of the authors and do not necessarily reflect the view of the DARPA or the US government.

References

Rie K. Ando and Tong Zhang. 2005. A framework for learning predictive structures from multiple tasks and unlabeled data. *Journal of Machine Learning Research*, 6:1817–1853.

Leo Breiman. 1996. Bagging predictors. *Machine Learning*, 24:123–140.

Dipanjan Das and Noah A. Smith. 2012. Graph-based lexicon expansion with sparsity-inducing penalties. In *Proceedings of the 2012 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT)*.

John DeNero, Shankar Kumar, Ciprian Chelba, and Franz Och. 2010. Model combination for machine translation. In *Proceedings of the 11th Annual Conference of the North American Chapter of the Association for Computational Linguistics (NAACL)*.

John Duchi and Yoram Singer. 2009. Efficient online and batch learning using forward backward splitting. *Journal of Machine Learning Research*, 10:2899–2934.

Bradley Efron and Robert J. Tibshirani. 1993. *An Introduction to the Bootstrap*. Chapman & Hall, New York.

Jenny R. Finkel, Trond Grenager, and Christopher D. Manning. 2005. Incorporating non-local information into information extraction systems by Gibbs sampling. In *Proceedings of the 43nd Annual Meeting of the Association for Computational Linguistics (ACL)*.

Tom Heskes. 1998. Selecting weighting factors in logarithmic opinion pools. In *Proceedings of Advances in Neural Information Processing Systems (NIPS) 10*.

Geoffery Hinton, Nitish Srivastava, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2012. Improving neural networks by preventing co-adaptation of feature detectors. *CoRR*, abs/1207.0580.

Geoffery Hinton. 1999. Products of experts. In *Proceedings of the Ninth International Conference on Artificial Neural Networks (ICANN)*.

Matthieu Kowalski and Bruno Torrésani. 2009. Sparsity and persistence: mixed norms provide simple signal models with dependent coefficients. *Signal, Image and Video Processing*, 3(3):251–264.

Thomas Lavergne, Olivier Cappé, and François Yvon. 2010. Practical very large scale CRFs. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics (ACL)*.

André F. T. Martins, Noah A. Smith, Pedro M. Q. Aguiar, and Mário A. T. Figueiredo. 2011. Structured sparsity in structured prediction. In *Proceedings of the Conference on Empirical Methods on Natural Language Processing (EMNLP)*.

Tomas Mikolov, Anoop Deoras, Stefan Kombrink, Lukas Burget, and Jan Cernocky. 2011. Empirical evaluation and combination of advanced language modeling techniques. In *Proceedings 12th Annual Conference of the International Speech Communication Association (INTERSPEECH)*.

Erik F. Tjong Kim Sang and Fien De Meulder. 2003. Introduction to the CoNLL-2003 shared task: language-independent named entity recognition. In *Proceedings of the 7th Conference on Natural language learning (CoNLL)*.

Erik F. Tjong Kim Sang, Walter Daelemans, Hervé Déjean, Rob Koeling, Yuval Krymolowski, Vasin Punyakanok, and Dan Roth. 2000. Applying system combination to base noun phrase identication. In *Proceedings of the 18th International Conference on Computational Linguistics (COLING)*.

Andrew Smith, Trevor Cohn, and Miles Osborne. 2005. Logarithmic opinion pools for conditional random fields. In *Proceedings of the 43th Annual Meeting of the Association for Computational Linguistics (ACL)*.

Charles Sutton, Michael Sindelar, and Andrew McCallum. 2007. Reducing weight undertraining in structured discriminative learning. In *Proceedings of Human Language Technologies: The Annual Conference of the North American Chapter of the Association for Computational Linguistics (NAACL-HLT)*.

Joseph Turian, Lev Ratinov, and Yoshua Bengio. 2010. Word representations: A simple and general method for semi-supervised learning. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics (ACL)*.

Sida I. Wang and Christopher D. Manning. 2013. Fast dropout training. In *Proceedings of the 30th International Conference on Machine Learning (ICML)*.

Hui Zou and Trevor Hastie. 2005. Regularization and variable selection via the elastic net. *Journal of the Royal Statistical Society, Series B*, 67:301–320.