Learning Combination Features with L_1 Regularization

Daisuke Okanohara† **Jun'ichi Tsujii**†‡§
†Department of Computer Science, University of Tokyo
Hongo 7-3-1, Bunkyo-ku, Tokyo, Japan
‡School of Informatics, University of Manchester
§NaCTeM (National Center for Text Mining)
{hillbig,tsujii}@is.s.u-tokyo.ac.jp

Abstract

When linear classifiers cannot successfully classify data, we often add *combination features*, which are products of several original features. The searching for effective combination features, namely *feature engineering*, requires domain-specific knowledge and hard work. We present herein an efficient algorithm for learning an L_1 regularized logistic regression model with combination features. We propose to use the grafting algorithm with efficient computation of gradients. This enables us to find optimal weights efficiently without enumerating all combination features. By using L_1 regularization, the result we obtain is very compact and achieves very efficient inference. In experiments with NLP tasks, we show that the proposed method can extract effective combination features, and achieve high performance with very few features.

1 Introduction

A linear classifier is a fundamental tool for many NLP applications, including logistic regression models (LR), in that its score is based on a linear combination of features and their weights,. Although a linear classifier is very simple, it can achieve high performance on many NLP tasks, partly because many problems are described with very high-dimensional data, and high dimensional weight vectors are effective in discriminating among examples.

However, when an original problem cannot be handled linearly, *combination features* are often added to the feature set, where combination features are products of several original features. Examples of combination features are, word pairs in document classification, or part-of-speech pairs of head and modifier words in a dependency analysis task. However, the task of determining effective combination features, namely *feature engineering*, requires domain-specific knowledge and hard work.

Such a non-linear phenomenon can be implicitly captured by using the kernel trick. However, its computational cost is very high, not only during training but also at inference time. Moreover, the model is not interpretable, in that effective features are not represented explicitly. Many kernels methods assume an L_2 regularizer, in that many features are equally relevant to the tasks (Ng, 2004).

There have been several studies to find efficient ways to obtain (combination) features. In the context of boosting, Kudo (2004) have proposed a method to extract complex features that is similar to the item set mining algorithm. In the context of L_1 regularization. Dudík (2007), Gao (2006), and Tsuda (2007) have also proposed methods by which effective features are extracted from huge sets of feature candidates. However, their methods are still very computationally expensive, and we cannot directly apply this kind of method to a large-scale NLP problem.

In the present paper, we propose a novel algorithm for learning of an L_1 regularized LR with combination features. In our algorithm, we can exclusively extract effective combination features without enumerating all of the candidate features. Our method relies on a grafting algorithm (Perkins and Theeiler, 2003), which incrementally adds features like boosting, but it can converge to the global optimum.

We use L_1 regularization because we can obtain a sparse parameter vector, for which many of the parameter values are exactly zero. In other words, learning with L_1 regularization naturally has an intrinsic effect of feature selection, which results in an

Proceedings of NAACL HLT 2009: Short Papers, pages 97–100,
Boulder, Colorado, June 2009.

efficient and interpretable inference with almost the same performance as L_2 regularization (Gao et al., 2007).

The heart of our algorithm is a way to find a feature that has the largest gradient value of likelihood from among the huge set of candidates. To solve this problem, we propose an example-wise algorithm with filtering. This algorithm is very simple and easy to implement, but effective in practice.

We applied the proposed methods to NLP tasks, and found that our methods can achieve the same high performance as kernel methods, whereas the number of active combination features is relatively small, such as several thousands.

2 Preliminaries

2.1 Logistic Regression Model

In this paper, we consider a multi-class logistic regression model (LR). For an input x, and an output label $y \in \mathcal{Y}$, we define a feature vector $\phi(x, y) \in R^m$.

Then in LR, the probability for a label y, given an input x, is defined as follows:

$$p(y|x; \boldsymbol{w}) = \frac{1}{Z(x, \boldsymbol{w})} \exp\left(\boldsymbol{w}^T \phi(x, y)\right), \quad (1)$$

where $\boldsymbol{w} \in R^m$ is a weight vector[1] corresponding to each input dimension, and $Z(x, \boldsymbol{w}) = \sum_y \exp(\boldsymbol{w}^T \phi(x, y))$ is the partition function.

We estimate the parameter $\boldsymbol{w}$ by a maximum likelihood estimation (MLE) with L_1 regularization using training examples $\{(x_1, y_1), \ldots, (x_n, y_n)\}$:

$$\boldsymbol{w}^* = \arg\min_{\boldsymbol{w}} -L(\boldsymbol{w}) + C \sum_i |w_i| \quad (2)$$

$$L(\boldsymbol{w}) = \sum_{i=1\ldots n} \log p(y_i|x_i; \boldsymbol{w})$$

where $C > 0$ is the trade-off parameter between the likelihood term and the regularization term. This estimation is a convex optimization problem.

2.2 Grafting

To maximize the effect of L_1 regularization, we use the grafting algorithm (Perkins and Theeiler, 2003); namely, we begin with the empty feature set, and incrementally add effective features to the current problem. Note that although this is similar to the boosting algorithm for learning, the obtained result is always optimal. We explain the grafting algorithm here again for the sake of clarity.

The grafting algorithm is summarized in Algorithm 1.

In this algorithm we retain two variables; $\boldsymbol{w}$ stores the current weight vector, and H stores the set of features with a non-zero weight. Initially, we set $\boldsymbol{w} = \mathbf{0}$, and $H = \{\}$. At each iteration, the feature is selected that has the largest absolute value of the gradient of the likelihood. Let $v_k = \frac{\partial L(\boldsymbol{w})}{\partial w_k}$ be the gradient value of the likelihood of a feature k. By following the definition, the value v_k can be calculated as follows,

$$v_k = \sum_{i,y} \alpha_{i,y} \phi_k(x_i, y), \quad (3)$$

where $\alpha_{i,y} = I(y_i = y) - p(y_i|x_i; \boldsymbol{w})$ and $I(a)$ is 1 if a is true and 0 otherwise.

Then, we add $k^* = \arg\max_k |v_k|$ to H and optimize (2) with regard to H only. The solution $\boldsymbol{w}$ that is obtained is used in the next search. The iteration is continued until $|v_k^*| < C$.

We briefly explain why we can find the optimal weight by this algorithm. Suppose that we optimize (2) with all features, and initialize the weights using the results obtained from the grafting algorithm. Since all gradients of likelihoods satisfy $|v_k| \leq C$, and the regularization term pushes the weight toward 0 by C, any changes of the weight vector cannot increase the objective value in (2). Since (2) is the convex optimization problem, the local optimum is always the global optimum, and therefore this is the global optimum for (2)

The point is that, given an efficient method to estimate v_k^* without the enumeration of all features, we can solve the optimization in time proportional to the active feature, regardless of the number of candidate features. We will discuss this in the next section.

3 Extraction of Combination Features

This section presents an algorithm to compute, for combination features, the feature v_k^* that has the largest absolute value of the gradient.

We propose an element-wise extraction method, where we make use of the sparseness of the training data.

In this paper, we assume that the values of the combination features are less than or equal to the original ones. This assumption is typical; for example, it is made in the case where we use binary values for original and combination features.

[1] A bias term b is often considered by adding an additional dimension to $\phi(x, y)$

Algorithm 1 Grafting

Input: training data (x_i, y_i) $(i = 1, \cdots, n)$ and parameter C
$H = \{\}, \boldsymbol{w} = \mathbf{0}$
loop
  $\boldsymbol{v} = \frac{\partial L(\boldsymbol{w})}{\partial \boldsymbol{w}}$ ($L(\boldsymbol{w})$ is the log likelihood term)
  $k^* = \arg\max_k |v_k|$ (The result of Algorithm 2)
  if $|\boldsymbol{v}_{k^*}| < C$ **then** break
  $H = H \cup k^*$
  Optimize $\boldsymbol{w}$ with regards to H
end loop
Output $\boldsymbol{w}$ and H

First, we sort the examples in the order of their $\sum_y |\alpha_{i,y}|$ values. Then, we look at the examples one by one. Let us assume that r examples have been examined so far. Let us define

$$\boldsymbol{t} = \sum_{i \le r, y} \alpha_{i,y} \phi(x_i, y) \quad (4)$$

$$\boldsymbol{t}^- = \sum_{i > r, y} \alpha^-_{i,y} \phi(x_i, y) \quad \boldsymbol{t}^+ = \sum_{i > r, y} \alpha^+_{i,y} \phi(x_i, y)$$

where $\alpha^-_{i,y} = \min(\alpha_{i,y}, 0)$ and $\alpha^+_{i,y} = \max(\alpha_{i,y}, 0)$.

Then, simple calculus shows that the gradient value for a combination feature k, v_k, for which the original features are $k1$ and $k2$, is bounded below/above thus;

$$\begin{aligned} t_k + t^-_k < \quad & v_k \quad < t_k + t^+_k \quad (5) \\ t_k + \max(t^-_{k1}, t^-_{k2}) < \quad & v_k \quad < t_k + \min(t^+_{k1}, t^+_{k2}). \end{aligned}$$

Intuitively, the upper bound of (5) is the case where the combination feature fires only for the examples with $\alpha_{i,y} \ge 0$, and the lower bound of (5is the case where the combination feature fires only for the examples with $\alpha_{i,y} \le 0$. The second inequality arises from the fact that the value of a combination feature is equal to or less than the values of its original features. Therefore, we examine (5) and check whether or not $|v_k|$ will be larger than C. If not, we can remove the feature safely.

Since the examples are sorted in the order of their $\sum_y |\alpha_{i,y}|$, the bound will become tighter quickly. Therefore, many combination features are filtered out in the early steps. In experiments, the weights for the original features are optimized first, and then the weights for combination features are optimized. This significantly reduces the number of candidates for combination features.

Algorithm 2 Algorithm to return the feature that has the largest gradient value.

Input: training data (x_i, y_i) and its $\alpha_{i,y}$ value $(i = 1, \ldots, n,\ y = 1, \ldots, |\mathscr{Y}|)$, and the parameter C. Examples are sorted with respect to their $\sum_y |\alpha_{i,y}|$ values.
$\boldsymbol{t}^+ = \sum_{i=1}^n \sum_y \max(\alpha_{i,y}, 0)\phi(x, y)$
$\boldsymbol{t}^- = \sum_{i=1}^n \sum_y \min(\alpha_{i,y}, 0)\phi(x, y)$
$\boldsymbol{t} = \mathbf{0}$, $H = \{\}$ // Active Combination Feature
for $i = 1$ to n and $y \in \mathscr{Y}$ **do**
  for all combination features k in x_i **do**
    if $|v_k| > C$ (Check by using Eq.(5)) **then**
      $v_k := v_k + \alpha_{i,y}\phi_k(x_i, y)$
      $H = H \cup k$
    end if
  end for
  $\boldsymbol{t}^+ := \boldsymbol{t}^+ - \max(\alpha_{i,y}, 0)\phi(x_i, y)$
  $\boldsymbol{t}^- := \boldsymbol{t}^- - \min(\alpha_{i,y}, 0)\phi(x_i, y)$
end for
Output: $\arg\max_{k \in H} v_k$

Algorithm 2 presents the details of the overall algorithm for the extraction of effective combination features. Note that many candidate features will be removed just before adding.

4 Experiments

To measure the effectiveness of the proposed method (called L_1-Comb), we conducted experiments on the dependency analysis task, and the document classification task. In all experiments, the parameter C was tuned using the development data set.

In the first experiment, we performed Japanese dependency analysis. We used the Kyoto Text Corpus (Version 3.0), Jan. 1, 3-8 as the training data, Jan. 10 as the development data, and Jan. 9 as the test data so that the result could be compared to those from previous studies (Sassano, 2004)[2]. We used the shift-reduce dependency algorithm (Sassano, 2004). The number of training events was $11,3332$, each of which consisted of two word positions as inputs, and $y = \{0, 1\}$ as an output indicating the dependency relation. For the training data, the number of original features was 78570, and the number of combination features of degrees 2 and 3 was 5787361, and 169430335, respectively. Note that we need not see all of them using our algorithm.

[2]The data set is different from that in the CoNLL shared task. This data set is more difficult.

Table 1: The performance of the Japanese dependency task on the Test set. The active features column shows the number of nonzero weight features.

	DEP. ACC. (%)	TRAIN TIME (S)	ACTIVE FEAT.
L_1-COMB	89.03	605	78002
L_1-ORIG	88.50	35	29166
SVM 3-POLY	88.72	35720	(KERNEL)
L_2-COMB3	89.52	22197	91477782
AVE. PERCE.	87.23	5	45089

In all experiments, combination features of degrees 2 and 3 (the products of two or three original features) were used.

We compared our methods using LR with L_1 regularization using original features (L_1-Original), SVM with a 3rd-polynomial Kernel, LR with L_2 regularization using combination features with up to 3 combinations (L_2-Comb3), and an averaged perceptron with original features (Ave. Perceptron).

Table 1 shows the result of the Japanese dependency task. The accuracy result indicates that the accuracy was improved with automatically extracted combination features. In the column of active features, the number of active features is listed. This indicates that L_1 regularization automatically selects very few effective features. Note that, in training, L_1-Comb used around 100 MB, while L_2-Comb3 used more than 30 GB. The most time consuming part for L_1-Comb was the optimization of the L_1-LR problem.

Examples of extracted combination features include POS pairs of head and modifiers, such as *Head/Noun*-Modifier/Noun, and combinations of distance features with the POS of head.

For the second experiment, we performed the document classification task using the Tech-TC-300 data set (Davidov et al., 2004)[3]. We used the tf-idf scores as feature values. We did not filter out any words beforehand. The Tech-TC-300 data set consists of 295 binary classification tasks. We divided each document set into a training and a test set. The ratio of the test set to the training set was $1:4$. The average number of features for tasks was $25,389$.

Table 2 shows the results for L_1-LR with combination features and SVM with linear kernel[4]. The results indicate that the combination features are effective.

[3]http://techtc.cs.technion.ac.il/techtc300/techtc300.html

[4]SVM with polynomial kernel did not achieve significant improvement

Table 2: Document classification results for the Tech-TC-300 data set. The column F_2 shows the average of F_2 scores for each method of classification.

	F_2
L_1-COMB	0.949
L_1-ORIG	0.917
SVM (LINEAR KERNEL)	0.896

5 Conclusion

We have presented a method to extract effective combination features for the L_1 regularized logistic regression model. We have shown that a simple filtering technique is effective for enumerating effective combination features in the grafting algorithm, even for large-scale problems. Experimental results show that a L_1 regularized logistic regression model with combination features can achieve comparable or better results than those from other methods, and its result is very compact and easy to interpret. We plan to extend our method to include more complex features, and apply it to structured output learning.

References

Davidov, D., E. Gabrilovich, and S. Markovitch. 2004. Parameterized generation of labeled datasets for text categorization based on a hierarchical directory. In *Proc. of SIGIR*.

Dudík, Miroslav, Steven J. Phillips, and Robert E. Schapire. 2007. Maximum entropy density estimation with generalized regularization and an application to species distribution modeling. *JMLR*, 8:1217–1260.

Gao, J., H. Suzuki, and B. Yu. 2006. Approximation lasso methods for language modeling. In *Proc. of ACL/COLING*.

Gao, J., G. Andrew, M. Johnson, and K. Toutanova. 2007. A comparative study of parameter estimation methods for statistical natural language processing. In *Proc. of ACL*, pages 824–831.

Kudo, T. and Y. Matsumoto. 2004. A boosting algorithm for classification of semi-structured text. In *Proc. of EMNLP*.

Ng, A. 2004. Feature selection, l1 vs. l2 regularization, and rotational invariance. In *NIPS*.

Perkins, S. and J. Theeiler. 2003. Online feature selection using grafting. *ICML*.

Saigo, H., T. Uno, and K. Tsuda. 2007. Mining complex genotypic features for predicting HIV-1 drug resistance. *Bioinformatics*, 23:2455–2462.

Sassano, Manabu. 2004. Linear-time dependency analysis for japanese. In *Proc. of COLING*.