LARGE MARGIN ESTIMATION OF N-GRAM LANGUAGE MODELS FOR SPEECH RECOGNITION VIA LINEAR PROGRAMMING

Vladimir Magdin, Hui Jiang

Department of Computer Science and Engineering, York University
4700 Keele Street, Toronto, Ontario M3J 1P3, CANADA
Email: {vmagdin, hj}@cse.yorku.ca

ABSTRACT

We present a novel discriminative training algorithm for n-gram language models for use in large vocabulary continuous speech recognition. The algorithm uses large margin estimation (LME) to build an objective function for maximizing the minimum margin between correct transcriptions and their competing hypotheses, which are encoded as word graphs generated from the Viterbi decoding process. The nonlinear LME objective function is approximated by a linear EM-style auxiliary function that leads to a linear programming problem, which is efficiently solved by convex optimization algorithms. Experimental results have shown that the proposed discriminative training method can outperform the conventional discounting-based maximum likelihood estimation methods. A relative reduction in word error rate of over 2.5% has been observed on the SPINE1 speech recognition task.

Index Terms— Large Margin Estimation (LME), n-gram Language Modeling, Linear Programming, LVCSR

1. INTRODUCTION

Statistical n-gram language models play an important role in automatic speech recognition (ASR), because they constrain the potentially-vast search space of hypotheses. N-gram language models are obtained via maximum likelihood estimation (MLE) from large bodies of text. However, even in a text corpus of millions of words, many n-grams never appear. Smoothing and discounting techniques, like the Katz back-off model [1], are used to approximate unseen n-grams’ probabilities. Language model obtained via traditional discounting-based MLE are typically suboptimal for a specific task, since the probabilities of many n-grams are crudely approximated by smoothing. To address this issue, this paper will introduce a novel discriminative training algorithm to adjust a language model for the purpose of improving speech recognition performance (word error rate).

Discriminative training of acoustic models has been studied extensively, and has become an integral part of modern state-of-the-art ASR systems. One class of algorithms use maximum mutual information estimation (MMIE), first applied to ASR by Bahl et al. [2], which involves constructing an objective function that tries to maximize the mutual information between a training word sequence and its corresponding sequence of acoustic features. Minimum classification error (MCE) [3] is another class of discriminative training approaches, which aim to optimize a differentiable objective function directly related to the recognition rate. Another discriminative training approach is the so-called large margin HMMs [4], which applies the principles of large margin classifiers from machine learning to maximizing the multi-class separation margin between utterance transcriptions and their competing hypotheses.

Discriminative training of language models has received less attention, compared to discriminative training of acoustic models. Stolcke et al. [5] constructed a separate “anti-LM” from hypotheses that are confusable with the correct transcriptions. Chen et al. [6] directly adjusted n-gram counts based on a discriminative objective function. Kuo et al. [7] applied the MCE algorithm to language model training via the GPD algorithm. Roark et al. [8] described discriminative n-gram modeling, a multi-pass algorithm based on global n-gram language models and two parameter estimation methods based on the perceptron algorithm, and regularized conditional maximum likelihood. This two-pass procedure was built upon by Zhou and Meng [9] to create a single-pass discriminative language modeling technique.

In our previous work, we proposed a novel discriminative training algorithm for n-gram language models [10]. The key idea behind the method is to formulate a discriminative objective function of all n-gram LM parameters in the logarithm space, including all conditional probabilities and back-off parameters. Then, the discriminative objective function is optimized with respect to all n-gram LM parameters subject to some constraints. We formulated the objective function using MMIE based on word graphs generated from the Viterbi decoding. Under some approximation conditions, it is shown that the above MMIE training can be converted into a linear program (LP) problem which can be solved efficiently even for very large-scale model size.

In this paper, we present an extension of our MMIE-based algorithm by formulating the discriminative training problem using large margin estimation (LME), which is based on the advances in the large margin classification vein of research in machine learning. The LME objective function is defined in Section 2. The necessary approximations and conversion of LME of n-gram LM into a linear program are presented in Section 3. Experimental results are shown in Section 4.

2. DISCRIMINATIVE TRAINING USING LME

In ASR, recognition is performed via the maximum a posteriori (MAP) decision rule [11]. For a speech utterance X, the decoder tries to find a sequence of words, W, such that:
\[
\hat{W} = \arg \max_{W} P_{\Lambda}(X|W) = \arg \max_{W} P_{\Lambda}(X) \cdot P_{\Lambda}(W)
\]
\[
= \arg \max_{W} P_{\Lambda}(X|\lambda_{W}) \cdot P_{\Lambda}(W)
\]
(1)

where \(P(X|W)\) is modeled via an HMM-based acoustic model, \(\lambda_{W}\), and the \(P(W)\) is modeled via an n-gram language model \(\Lambda\).
Without loss of generality, a tri-gram language model is assumed. To facilitate the following presentation, all parameters in a tri-gram language model are represented in the logarithm space. As a result, the entire tri-gram language model \( \Lambda \) can be defined as follows:

\[
\Lambda = \{ \lambda_i, \eta_j, \mu_k, \phi_i, \psi_m \} \quad i \in P_1, j \in P_2, k \in P_1, l \in Q_2, m \in Q_1 \quad (2)
\]

where \( \lambda_i, \eta_j, \mu_k, \phi_i, \psi_m \) denote conditional probabilities (in log domain) for tri-grams, bi-grams, uni-grams, and back-off weights (in log domain) for bi-grams, and uni-grams, respectively; \( P_1 \) represents all \( m \)-grams in \( \Lambda \), and \( Q_1 \) represents all cases which require an \( m \)-gram to back-off to an \((m + 1)\)-gram. In this way, the log likelihood function of the above tri-gram language model can be easily represented as a linear function of all parameters of \( \Lambda \) in equation (2) as shown in the following sections.

Following the research developments in the machine learning community, large margin methods have been become increasingly popular in many different fields [12]. Large margin classifiers aim to maximize the margin (or the confidence level) of classification, instead of minimizing the classification error. Large margin estimation of HMM parameter for ASR was proposed by Jiang et. al [13], [4], and can be similarly applied to language model training. In this work, competing hypotheses are encoded as word graphs [14], which are directed acyclical graphs (DAGs), which compactly store a large number of hypotheses. The following decision margin, \( d(\Lambda) \), is defined for each utterance:

\[
d(W|\Lambda) = \max_{W' \in \mathcal{A}(W)} \ln[\Pr(W'|\Lambda) \cdot \mathcal{A}(W')] - \ln[\Pr(W|\Lambda) \cdot \mathcal{A}(W)] \quad (3)
\]

where \( \mathcal{A}(W) \) is a constant term related to the acoustic score of sentence \( W \). The maximization in Equation (3) over all paths in a word graph is exponential in complexity, since explicitly finding the path with the highest probability involves enumerating all possible paths. This maximization can be approximated with a log-summation, the so-called soft-max operation. Using the relaxation techniques presented in the next section, this summation will be computed efficiently. Therefore, the essence of LME training of a language model lies in optimizing \( n \)-gram parameters in a way that maximizes the minimum margin over the entire data set:

\[
d(W_n|\Lambda) = \ln[\Pr(W_n|\Lambda) \cdot \mathcal{A}(W_n)] - \ln \sum_{W' \in \mathcal{A}(W)} \Pr(W'|\Lambda) \cdot \mathcal{A}(W') \quad (4)
\]

The LME procedure attempts to find the minimum margin from the support set, \( S = \{ W : 0 \le d(W_n|\Lambda) \le \gamma \} \), which only contains utterances that have a positive decision margin. The positive constant \( \gamma \) limits the size of the support set for performance considerations, because only the data points which are close to the decision margin play a role in determining its location. Following [15], the incorrectly-classified utterances with a negative margin form the so-called error set \( \mathcal{E} = \{ W : d(W_i|\Lambda) \le 0 \} \). The error set is incorporated into LME training as follows:

\[
\arg\max_{\Lambda} \left[ \min_{W_n \in S} d(W_n|\Lambda) - \epsilon \cdot \frac{1}{|\mathcal{E}|} \sum_{W_i \in \mathcal{E}} d(W_i|\Lambda) \right] \quad (5)
\]

where \( \epsilon \) is a positive multiplier that controls the degree of contribution of the error set to the training problem.

3. SOLVING LME WITH LINEAR PROGRAMMING

In order to solve the maximization problem in Equation (5), this section describes a lower-bound expectation-based approximation to the log-sum term that is based on the well-known Jensen inequality, similar to the auxiliary function used in the Expectation-Maximization algorithm. Consequently, we are able to convert the LME of \( n \)-gram LM into a linear programming problem, which can be solved efficiently using readily-available convex optimization tools.

3.1. Correct Transcription

The term related to the correct transcription of an utterance is modeled by a simple Markov Chain back-off \( n \)-gram model. This term can be expressed as a linear function of the individual language model parameters (\( n \)-gram probabilities, back-off weights), via the following set of transformations:

\[
\ln[\Pr(W'|\Lambda) \cdot \mathcal{A}(W)] = \sum_{t=1}^{R_W} \ln[\Pr(w_t|w_{t-2}w_{t-1}) + B'] = \sum_{t=1}^{R_W} \ln \Pr(w_t|w_{t-2}w_{t-1}) + B' \quad (6)
\]

where \( R_W \) is the length of utterance \( W \), and \( B' \) is a constant due to acoustic scores. Equation (6) is a linear function of language model parameters, and can be expressed slightly differently to clarify this fact:

\[
\ln[\Pr(W'|\Lambda) \cdot \mathcal{A}(W)] = \sum_{i \in P_1} a'_i(W) \cdot \lambda_i + \sum_{j \in P_2} b'_j(W) \cdot \eta_j + \sum_{k \in P_1} c'_k(W) \cdot \mu_k + \sum_{l \in Q_2} d'_l(W) \cdot \phi_l + \sum_{m \in Q_1} e'_m(W) \cdot \psi_m + B' \quad (7)
\]

where \( a'_i(W) \) denotes the count of tri-gram context \( i \) in sentence \( W \), \( b'_j(W) \) denotes the count of bi-gram context \( j \) in sentence \( W \), and the remaining constants \( c'_k(W), d'_l(W), \) and \( e'_m(W) \) represent the counts of their corresponding contexts in sentence \( W \). These counts are initially set to zero, and consequently incremented by considering each tri-gram context \( w_tw_{t-1}w_{t-2} \) in sentence \( W \) and incrementing the variables in the following order:

- if \( \lambda_a = \phi_b = P_1 \), increment \( a_{abc}'(W) \)
- otherwise, if \( \eta_c = P_2 \)
  - increment \( b_{bc}'(W) \)
  - if \( \phi_{ab} = Q_2 \), increment \( d_{ab}'(W) \)
- otherwise, increment \( c_{c}'(W) \) and \( e_{c}'(W) \)

3.2. Competing Hypotheses

The term related to the competing hypotheses is approximated using the EM-style auxiliary function, \( Q(\Lambda|\Lambda^{(n)}) \), which is computed as follows:

\[
Q(\Lambda|\Lambda^{(n)}) = \sum_{W' \in \mathcal{G}} \ln[\Pr(W'|\Lambda) \cdot \mathcal{A}(W')] \cdot \Pr(W'|\Lambda^{(n)}, \mathcal{G})
\]

\[
= \sum_{W' \in \mathcal{G}} \sum_{t=1}^{R_{W'}} \ln \Pr(w_t'|w_{t-2}', w_{t-1}') \cdot \gamma_{W'} + B'' \quad (8)
\]

5399
where \( \gamma_{W'} = \Pr(W'|\Lambda^{(n)}, G) \) is the so-called posterior probability of path \( W' \) in word-graph \( G \), and \( B'' \) is an acoustic score related constant. Using the same intuition as in Equation (7), Equation (8) can be expressed as a linear function of the language model parameters as follows:

\[
Q(\Lambda|\Lambda^{(n)}) = \sum_{i \in P_3} a_i''(G) \cdot \lambda_i + \sum_{j \in P_2} b_j''(G) \cdot \eta_j + \sum_{k \in P_1} c_k''(G) \cdot \mu_k + \sum_{l \in Q_2} d_l''(G) \cdot \phi_l + \sum_{m \in Q_1} e_m''(G) \cdot \psi_m + B''
\]  

(9)

where \( a_i''(G) = \sum_{i \in G} \gamma_i \), denotes the sum of posterior probabilities for each occurrence of context \( i \) in word graph \( G \), and \( b_j''(G) \) through \( e_m''(G) \) serve respective roles for their corresponding language model parameters in \( G \) and they are all computed in the same way as \( a_i''(G) \) based on the sum of posterior probabilities for all other contexts:

\[
\eta_j(j \in P_2), \mu_k(k \in P_2), \phi_l(l \in Q_2) \text{ and } \psi_m(m \in Q_1).
\]

Initially, the constants are set to zero, and consequently, for each tri-gram \( w_{n-1}w_nw_{n+1} \), the constants are computed as follows:

- if \( \lambda_{abc} \in P_3 \), compute \( a''_{abc}(G) \)
- otherwise, if \( \eta_{bc} \in P_2 \)
  - compute \( b'_{bc}(G) \)
  - if \( \phi_{ab} \in Q_2 \), compute \( d'_{ab}(G) \)
- otherwise, compute \( c'_i(G) \) and \( e'_m(G) \)

The posterior probabilities of all contexts, i.e., \( \lambda, \eta, \mu, \phi \) and \( \psi \), can be computed efficiently, using the forward-backward algorithm for word-graphs, as described by Wessel et al. [14].

### 3.3. Linear Programming Solution

The LME formulation in Equation (5) can be approximated with a linear function using the results from Equations (7) and (9), to form the approximate margin:

\[
d(W_n|\Lambda) \approx \sum_{i \in P_3} a_i(W, G) \cdot \lambda_i + \sum_{j \in P_2} b_j(W, G) \cdot \eta_j + \sum_{k \in P_1} c_k(W, G) \cdot \mu_k + \sum_{l \in Q_2} d_l(W, G) \cdot \phi_l + \sum_{m \in Q_1} e_m(W, G) \cdot \psi_m - B_n
\]

(10)

where \( B_n = B''_n - B'' \). The parameter coefficient \( a_i \) through \( e_i \) are computed as follows:

\[
a_i(W, G) = a''_{abc}(W) - a_i''(G), \\
b_j(W, G) = b''_{bc}(W) - b_i''(G), \\
c_k(W, G) = c''_i(W) - c'_i(G), \\
d_l(W, G) = d''_{ab}(W) - d_i''(G), \\
e_m(W, G) = e''_m(W) - e'_m(G).
\]

As shown in [13] and [4], the max-min problem in Equation (5) can be expressed as a simpler maximization operation by introducing an extra variable \( \rho \) to represent the margin. This leads to the following linear program formulation of LME of \( n \)-gram LM:

\[
\begin{align*}
\text{argmax}_{\Lambda, \rho} & \quad \rho - \frac{\epsilon}{|\mathcal{E}|} \cdot c^T \Lambda \\
\text{Subject to:} & \\
& \forall W_n \in \mathcal{S} : d(W_n|\Lambda) \geq \rho \\
& \forall i \in \Lambda : s^{(n)} - \tau \leq i \leq s^{(n)} + \tau \\
& \rho \geq 0
\end{align*}
\]

(12)

The training algorithm algorithm iteratively improves the language model by performing two steps in each iteration. The first step involves using the forward-backward algorithm to compute all the necessary posterior probabilities from word graphs, in order to build up the LME LP problem, and the second step involves solving the linear program in order to get a new and improved language model. The above optimization problem is a standard linear program where the objective function and all constraints are linear functions of free variables. Therefore, it can be efficiently solved by using many standard convex optimization tools and the new language model parameters can be derived from the found solution to the linear program.

### 4. EXPERIMENTS

The effectiveness of the proposed discriminative training algorithm has been evaluated on the Speech in Noisy Environments 1 (SPINE1) task [16], which features speech from simulated military exercises. SPINE1 contains six partitions, each of which features different levels of background noise. The "quiet" subset, which features the least amount of background noise, was chosen for experiments in this paper. This subset contains 5210 training utterances, and 2030 test utterances. A trigram language model was built directly from the training data of the entire SPINE1 task and it contains 1210 unigrams, 12880 bigrams, and 27924 trigrams. The language model was built using the CMU-Cambridge Statistical Language Modeling toolkit. HMM model training, word graph generation, as well as recognition were performed using HTK. GNU Linear Programming Kit was used to solve the linear programming problems. The running time, which is highly dependent on the size of the word graphs, was about one hour per iteration for our parallelized implementation running on a dual quad-core Xeon E5420 machine. In our experiments, fewer than 15 iterations were needed to obtain desired results.

We first obtained the baseline for our experiments by running the recognizer on the SPINE1 database using the original LM. We also obtained word graphs for each utterance, as a byproduct of the recognition process, by using a weak language model (unigram) in
order to prevent the over-fitting problems inherent to statistical language modeling.

The performance of the training procedure is affected by several parameters. The size of the box bounds on LM parameters controls the quality of approximation, at the expense of running time. Smoothing factor, which is a small (less than unity) multiplicative factor that’s applied to acoustic and language model probabilities in word graphs, was found to be crucial in improving generalization performance of the training algorithm, because it helps to spread out the posterior probabilities in a word graph, preventing just one path from being extremely dominant. Pruning LM parameters with small associated coefficients was found beneficial for performance, because some of the smaller coefficients may be caused by noise. The motivation behind pruning is that the training procedure should only focus on the more important parameters. The final parameter comes from Equation (12): $\epsilon$, which controls the contribution of the error set to the objective function. Detailed results of the effects of parameter tuning on performance are omitted due to space limitations.

The best performance of the training algorithms is shown in Table 1. Word error rates (WER) and sentence error rates (SER) are reported for MMIE and LME. The values inside parentheses indicate the relative change. The LME algorithm achieves a reduction in word error rate of close to 2.6% on unseen evaluation data, while the relative error reduction on training data is roughly 55%. This improvement is slightly better than the 2.3% relative reduction in error on unseen data we obtained with our MMIE-based algorithm.

The LME-based discriminative training algorithm, as presented in this paper, has been shown to be effective at reducing error rates in a speech recognition task. Furthermore, the LME-based formulation has been shown to be an improvement over the MMIE-based algorithm. This work can also be extended to discriminatively train language models for tasks like optical character recognition (OCR), machine translation, information retrieval, and others. The methods presented here are general and can be easily scaled to handle any size of language model. Furthermore, the objective functions described here are global and encompass all language model parameters: probabilities and back-off weights, unlike some of the previous efforts in this area of research. Finally, this research can also be extended to the general case of discriminative training from graphical models, which word graphs are an example of. However, the great disparity between the training set performance and the test set performance shows that there remains a great deal of work in improving the generalization performance of discriminative training algorithms. The LME-based algorithm is one forward in that direction.

6. REFERENCES


