ABSTRACT
Conditional random fields (CRFs) have recently found popularity in automatic speech recognition (ASR) applications. CRFs have previously been shown to be effective combiners of posterior estimates from multilayer perceptrons (MLPs) in phone and word recognition tasks. In this paper, we describe a novel hybrid Multilayer-CRF structure (ML-CRF), where a MLP-like hidden layer serves as input to the CRF; moreover, we propose a technique for directly training the ML-CRF to optimize a conditional log-likelihood based criterion, based on error backpropagation. The proposed technique thus allows for the implicit learning of suitable feature functions for the CRF. We present results for initial phone recognition experiments on the TIMIT database that indicate that our proposed method is a promising approach for training CRFs.

Index Terms—Random Fields, Multilayer Perceptrons, Backpropagation, Speech Recognition

1. INTRODUCTION
Conditional random fields (CRFs) [1] have recently found application in automatic speech recognition (ASR) tasks. One of the main advantages of CRFs is the fact that they do not need to make restrictive independence assumptions about the data compared to the Hidden Markov models (HMMs) that are traditionally used in ASR. CRFs directly estimate the probability of a label sequence conditioned on an input sequence - expressed as a normalized exponential sum of weighted feature functions. The performance of a CRF system on a given task is strongly dependent on the feature functions used by the system; examples of ASR CRF systems include phone classifiers using the sufficient statistics of MFCCs [2] and phone recognizers using phone-class and phonological feature posteriors [3] or gaussian responsibility scores [4]. In all of these cases, the input representation (PLP or MFCC coefficients) is transformed by a non-linear mapping to obtain suitable feature functions.

We propose a new technique for automatically learning feature functions for CRFs by integrating the machinery of multilayer perceptrons (MLPs) directly into the CRF system. Our method is based on optimizing a CRF-based loss functions directly using error backpropagation. This is similar to previous work [5] in which a neural network acoustic model is directly optimized with sequence-based optimization criteria. Our work, however, is directly concerned with learning feature functions for CRF-based phone recognition and does not use lattice-based techniques, unlike the work in [5]. Our initial experiments indicate that these are feasible entities to train; we show that the implicitly-learned functions of a hidden layer perform comparably or insignificantly improve the CRF system relative to external MLP inputs as feature functions.

In Section 2 we present a brief review of conditional random fields. Our proposed multilayer CRF (ML-CRF) model is presented in Section 3, with a training procedure presented in Section 4. We describe TIMIT phone recognition experiments and results in Sections 5 and 6 respectively. We end with our conclusions and a discussion of future work.

2. CONDITIONAL RANDOM FIELDS
Conditional random fields (CRFs) [1] are probabilistic models for labeling data sequences. CRFs are discriminative models that directly estimate the probability, \( p(y|x) \), of the label sequence \( y \) conditioned on the data \( x \). The sequence probability distribution can be decomposed into a number of factors based on the cliques of an undirected graph; for a linear-chain CRF (similar to the structure of an HMM), the cliques of the graph correspond to the vertices and edges, and therefore by the Hammersley-Clifford theorem we can write [1]

\[
p(y|x) = \frac{1}{Z(x)} \sum_{i,j} \lambda_{ij} f_j(y_i, x) + \sum_{i,k} \mu_{ik} t_k(y_{i-1}, y_i, x)
\]

where, \( Z(x) \) is a normalization term to ensure that Equation (1) forms a valid probability distribution. and is defined as

\[
Z(x) = \sum_y \exp \left\{ \sum_{i,j} \lambda_{ij} f_j(y_i, x) + \sum_{i,k} \mu_{ik} t_k(y_{i-1}, y_i, x) \right\}.
\]

The functions \( f_j(\cdot) \) and \( t_k(\cdot) \) in Equation (1) are known as the state feature functions and transition feature functions respectively and they allow the CRF to capture arbitrary dependencies in the data. Given a set of M training examples, \( \{x^n, y^n\}_{n=1,\ldots,M} \), the parameters \( \{\lambda_1, \lambda_2, \ldots ; \mu_1, \mu_2, \ldots \} \) of the system can be trained to maximize the conditional log-likelihood given by [1]

\[
\mathcal{L} = \sum_{n=1}^{M} \log p(y^n|x^n)
\]
Maximum aposteriori (MAP) estimation using a prior over the parameters [6] has also been suggested to avoid overfitting. The conditional log-likelihood is concave over the entire parameter space and can be optimized using standard convex optimization techniques.

3. LEARNING FEATURE FUNCTIONS FOR CRFS USING MULTILAYER PERCEPTRONS

As can be observed from Equation (1), the performance of CRFs for a particular task rests firmly on being able to identify suitable feature functions $f(\cdot)$ and $t(\cdot)$. Identifying such feature functions can be a non-trivial task in many domains. This is especially true in ASR, where the input speech is usually represented by MFCC or PLP coefficients, and previous studies using CRFs for phone recognition [2] [3] [4] have demonstrated the importance of suitably transforming this input representation to derive feature functions for CRFs. Our proposed system, that we term a multilayer CRF (ML-CRF), is similar to these, in that we transform the input representation to construct feature functions from the data. However, unlike these systems, our feature functions do not have any direct interpretation and are learned implicitly.

Previous work in our lab has relied on multilayer perceptrons (MLPs) to provide functions related to posterior estimates of linguistic classes (e.g., phones or phonological features) [3]. In order to estimate posteriors, the MLPs are trained with a cross-entropy criterion and utilize a softmax layer output. The layer for a single frame’s output for a particular phone $y_k$ (where $q$ is the phone index) looks remarkably similar to Equation (1):

$$p_{\text{MLP}}(y_k|x) = \frac{1}{Z(x)} \exp \sum_j \lambda_{q,j} z_j(y_k, x)$$

where $z_j$ corresponds to the output of the $j^{th}$ hidden unit, and $\lambda_{q,j}$ corresponds to the MLP weight connecting $z_j$ and $y_k$. $x$ corresponds to the acoustics within the MLP’s input window. In other words, the MLP’s hidden layer can be thought of as providing the state feature functions of a CRF, albeit with only local (frame-wise) posterior normalization. The CRF provides transition information as well as a global (sequence) posterior normalization.

We propose to build the ML-CRF by utilizing a trainable hidden layer (as in an MLP) to provide the traditional feature function representations. Our system uses a multilayer perceptron (MLP), with one layer of hidden units, with a linear activation function for the output layer units and a sigmoid activation function for the hidden layer units. The MLP is constructed with $|\mathcal{Y}|$ units in the output layer, where $\mathcal{Y}$ is the set of all possible labels. As shall be explained in Section 4, the MLP is trained so that each output unit shall represent a CRF feature function corresponding to a particular label. If the activation of the output unit corresponding to the label $y_i$ is denoted by $s(y_i, t)$, when the MLP is provided with suitable input corresponding to the data $x$ at time frame $t$, then the probability of a label sequence $y$ conditioned $x$ can be written as

$$p(y|x) = \frac{1}{Z(x)} \exp \sum_i \left\{ s(y_i, t) + \sum_k \mu_k t_k(y_{i-1}, y_i, x) \right\}$$

where, $Z(x)$ is a normalization term as before. Contrasting Equation (1) with Equation (5) it is clear that the difference between the two lies in the fact that the standard CRF model treats the feature functions as fixed and known apriori and seeks to learn values for the weights associated with them. On the other hand, our ML-CRF system directly seeks to determine the feature functions during the MLP training. Since we use a linear activation function on the output layer nodes, one can view the hidden unit activations as being analogous to the state feature functions $f_j(y_k, x, t)$ in Equation (1) with the hidden-to-output layer weights being analogous to the associated weights $\lambda_j$ in the linear-chain CRF. We also note here that, if the inputs to the MLP consist of feature functions $f_j(y_k, x, t)$ used in a standard CRF system (for example sufficient statistics of MFCCs [2], phone and phonological posteriors [3] or gaussian responsibility scores [4]) then the feature functions derived from the MLP (in the ML-CRF system) shall represent non-linear combinations of these features. Alternatively, directly using a representation of the input data sequence $x$ as input to the ML-CRF allows the system to automatically learn suitable feature functions from the data. This increase in modeling power comes at a cost: the loss function is no longer convex over the parameter space (see Section 4). Thus, as with MLPs, our ML-CRF system is no longer guaranteed to attain a global optimum over the parameter space.

4. BACKPROPAGATION TRAINING FOR ML-CRFS

In this section, we describe an algorithm for training the MLP to obtain feature functions for the CRFs as described in Section 3 that is based on the error backpropagation algorithm [7] used traditionally for training multilayer perceptrons. For our system, we directly optimize the negative conditional log-likelihood of the CRF (negative log-loss) as the loss function. Thus, from Equation (3) the loss function for the MLP is given by

$$\hat{E} = -\mathcal{L} = \sum_{n=1}^{M} -\log p(y^n|x^n)$$

Training the MLP to minimize the loss function in Equation (6) requires us to compute its gradients with respect to the network weights. To compute the required gradients, we use the following notation. Each unit in the hidden and the output layers of the network first computes a linear weighted sum of the activations of all the units in previous layers from which it receives connections. If we denote the activations of a unit $i$ in the network by $z_i$, and the weights associated with a connection from a unit $j$ to a unit $k$ by $w_{kj}$, then the activation of a unit $i$ in the hidden layer is computed as, $z_i = g(\sum_j w_{ij} z_j)$, where $g(\cdot)$ is a suitable activation function. As mentioned before, in our experiments, we use a sigmoid activation function for hidden layer nodes, and a linear activation function for output layer nodes. Note that we use an additional bias term $z_0 = 1$ and corresponding weight $w_{i0}$ associated with each of the nodes in the hidden and the output layers. The ‘activations’ $z_i$ of the input layer nodes correspond to the inputs $x_i$ of the MLPs themselves.

Since the gradients of $\hat{E}$ in Equation (6) with respect to the network weights can be obtained by summing up the respective gradients for each of the training examples, we demonstrate the gradient computation for a single training instance. Thus, we can re-write Equation (6) for a single example as

$$E = -\sum_i \left\{ s(y_i, t) + \sum_k \mu_k t_k(y_{i-1}, y_i, x) \right\} + \log Z(x)$$

To compute the partial derivatives of Equation (7) with respect to a weight $w_{kj}$ connecting a unit $j$ in the hidden layer to a unit $k$ in the

---

1The linear output layer from the feature functions allows the CRF to perform a softmax-like log-linear operation in the final computation according to Equation (1).
output layer, we first observe that the use of a linear activation function for the output layer nodes implies that \( s(k, i) = \sum_j w_{kj} z_j(i) \), where the summation is over all hidden layer nodes and \( z_j(i) \) is the activation of the \( j \)-th hidden layer unit corresponding to the inputs at frame \( i \). Thus, we can write,

\[
\frac{\partial s(y, i)}{\partial w_{kj}} = \begin{cases} 
  z_j(i) & \text{if } y = k \\
  0 & \text{otherwise}
\end{cases} \tag{8}
\]

Differentiating Equation (7), with respect to a hidden to output layer weight, \( w_{kj} \), using Equation (8) and after simplification we get

\[
\frac{\partial E}{\partial w_{kj}} = -\sum_i \delta(y_i = k) z_j(i) + \sum_i p(y_i = k|x) z_j(i) \tag{9}
\]

where, \( \delta(a = b) \) is an indicator function that is 1 if and only if \( a = b \) and 0 otherwise. Note that these computations are analogous to those used for computing the gradients with respect to the CRF weights in a standard linear-chain CRF system.

To compute the partial derivatives of Equation (7) with respect to a weight \( w_{jl} \) connecting a unit \( l \) in the input layer to a unit \( j \) in the hidden layer, we observe that, the MLP output corresponding to the label \( y \) is only affected by changes to the weight \( w_{jl} \) through the activation \( z_j \) of the \( j \)-th hidden layer unit, and hence,

\[
\frac{\partial s(y, i)}{\partial w_{jl}} = \frac{\partial s(y, i)}{\partial z_j(i)} \frac{\partial z_j(i)}{\partial w_{jl}} = w_{yl} g'(a_j(i)) x_l(i) \tag{10}
\]

where \( x_l(i) \) is the value of the \( l \)-th input to the MLP at time frame \( i \), \( g'(\cdot) \) is the derivative of the sigmoid function and \( a_j(i) = \sum_l w_{jl} x_l(i) \) is the input sum into the \( j \)-th hidden layer unit. Using Equation (10) we can write the derivative of \( E \) with respect to the weight \( w_{jl} \) as

\[
\frac{\partial E}{\partial w_{jl}} = -\sum_i w_{yl} g'(a_j(i)) x_l(i) + \sum_i p(y_i = y|x) w_{yl} g'(a_j(i)) x_l(i) \tag{11}
\]

The marginal probabilities \( p(y_i = y|x) \) that appear in Equation (9) and (11) can be computed by a standard dynamic programming technique used in linear-chain CRFs. Details of these computation can be found in [1]. Thus, once the gradients have been computed using Equations (9) and (11), the weight update for a particular weight \( w_{kj} \) can be computed as

\[
w_{kj} \leftarrow w_{kj} - \eta \frac{\partial E}{\partial w_{kj}} \tag{12}
\]

where \( \eta \) is the step-size. The updates for the weights \( \mu_{kl} \) associated with the transition features in Equation (5) can be computed exactly as in a standard linear-chain CRF since these do not depend on the MLP. We refer the reader to [1] [3] or [6] for details.

5. EXPERIMENTS

To determine the effectiveness of our proposed method, we conducted phone recognition experiments on the TIMIT corpus [8]. The ML-CRF system described in this paper was implemented by modifying the QuickNet MLP software package [9]. Both the ML-CRF system, as well as the linear-chain CRF system that we use for comparison are trained using a stochastic gradient descent based optimization algorithm, with the parameters of the system being updated after processing each utterance in the training set. Also note that both systems are trained to optimize the conditional log-likelihood criterion as described in Equations (3) and (6) respectively.

In our first set of experiments, we analyze the ability of the ML-CRF system to learn feature functions from direct representations of the data. We use 12th order PLP coefficients (with cepstral mean normalization and double-deltas) to obtain a 39-dimensional vector representing each frame as the inputs to the ML-CRF system. As a baseline for comparison, we use a linear-chain CRF system (STD-CRF) using the PLP coefficients augmented with their squares (PLP+SQ) to form a 78-dimensional vector representing each frame of the input utterance.

In a second set of experiments, we used phone posteriors obtained from a MLP-based phone classifier as the feature functions. The phone classifier was constructed with 2000 hidden layer units, with a sigmoid activation function on hidden layer units and a softmax activation function for output layer units. We experimented with a phone classifier using both 48-class labels (48Lab) as defined in [10] as well as 61-class labels (61Lab) based on the full TIMIT set.

In the linear-chain CRF system, we associate a state feature with every possible label and feature pair. In other words, for each feature \( r \) and label \( y \), we define a state feature function,

\[
f_{(r,y)}(y_i, x) = \begin{cases} 
  \text{val}(r, x, y, i) & \text{if } y_i = y \\
  0 & \text{otherwise}
\end{cases} \tag{13}
\]

where \( \text{val}(r, x, y, i) \) is the value of the feature \( r \) corresponding to the input at frame \( i \) and label \( y \). In both the linear-chain CRF systems as well as the ML-CRF systems, we use only a bias transition feature function, associated with every label pair, defined as follows,

\[
t_{(y', y)}(y_i, x) = \begin{cases} 
  1 & \text{if } y_i - 1 = y' \text{ and } y_i = y \\
  0 & \text{otherwise}
\end{cases} \tag{14}
\]

The optimal number of ML-CRF hidden units was determined using the 400-sentence development set defined by Halberstadt and Glass [11]. We collapse all system results to 39 phone labels for scoring following the standard TIMIT recognition paradigm [10].

6. RESULTS AND DISCUSSION

The results of our experiments are reported on the 192-sentence core test set as well as the larger 944-sentence enhanced test set consisting of all non-SA sentences from the full test set not in the development set. Note that a difference in performance of about 1.2% on the core test set and about 0.6% on the enhanced test is statistically significant at the \( p \leq 0.05 \) level.

As can be seen in Table 1, the ML-CRF system trained directly on the PLP coefficients significantly outperforms the linear-chain CRF systems trained directly on the PLP coefficients augmented with their squares (modeling the sufficient statistics). The relatively poor performance of the linear-chain CRF system is explained by the fact that these systems use a single-label for each phone (1-state model), effectively modeling each phone by a single gaussian; improvements can theoretically be made by integrating a 3-state mixture model. Nevertheless, as a pilot experiment, this is a satisfying result, since it conforms with our intuition that a non-linear transformation of the PLPs is important for good recognition accuracy.

The results of our experiments using phone posteriors as the input representation also appear in Table 1. The results in the table indicate that the ML-CRF system trained on 61-class posteriors as
its input representation significantly outperforms a standard linear-chain CRF system trained on the 48 or 61-class posteriors with a single frame window of context (although the MLP posteriors observed 9 frames of acoustic input). However, when the linear-chain CRF systems are provided with a 9 frame context window, they perform comparably with the ML-CRF system. As was noted previously, it is possible to interpret the outputs of the hidden layer in the ML-CRF system as being analogous to the input feature functions in the linear-chain CRF system. It is thus interesting to note that the ML-CRF systems achieve comparable performance using significantly fewer feature functions than the standard linear-chain CRFs (requiring effectively 61*9=549 input feature functions). The ML-CRF is achieving some level of information compression relative to the standard system, although there are certainly more parameters in the ML-CRF system due to the full connectivity of the input to hidden layer in the ML-CRF.

The phone recognition results from these pilot experiments are in line with previous work with single phone-posterior streams [3], but do not reach the state of the art (ranging from 74-79% phone accuracy). However, we are encouraged that the system was able to automatically extract usable feature functions directly from the acoustic data; we suggest some fruitful extensions in the conclusions that may be able to better utilize the ML-CRF technology.

7. CONCLUSIONS AND FUTURE WORK

In this paper we have presented a multilayer CRF model and a training procedure based on error backpropagation. Our proposed system inherit the advantages of MLPs, allowing non-linear feature combination and feature expansion using a hidden layer representation. Our initial TIMIT phone recognition results suggest that the ML-CRF system may be a promising approach for training CRFs.

Although our pilot experiments with posterior inputs yielded insignificant improvements over the baseline, we believe that our proposed system would be effective in situations where we would expect a non-log-linear interaction between the features, as evidenced by our results using PLPs as the input features. We intend to explore the use of the ML-CRF systems as feature combiners, since it has been previously shown [12] that dimensionality reduction and a non-linear transformation are important when using multiple feature streams for phone recognition. Note that these requirements are naturally and elegantly captured in the ML-CRF model.

In future work, we intend to examine the effectiveness of using different network structures in the underlying MLP in the ML-CRF system. For example, Bottleneck Features [13] utilize a narrow hidden layer in a 5-layer MLP to compress phonetic information into a set of features usable in the Tandem MLP-HMM setting. Given the theoretical advances of this paper, extending the backpropagation of CRF errors to multiple layers is relatively straightforward.

ML-CRF systems could be employed to implicitly learn transition feature functions in addition to the state feature functions, albeit at the cost of a substantial increase in the number of parameters and system training time. Our preliminary results with this approach suggest that the non-convexity of the loss function coupled with the large increase in system parameters makes the system more susceptible to converge to a local minimum. We intend to explore the use of better initializations for the ML-CRF system to avoid this issue.

8. REFERENCES