Structured Regularizer for Neural Higher-Order Sequence Models

Martin Ratajczak[1(✉)], Sebastian Tschiatschek[2], and Franz Pernkopf[1]

[1] Signal Processing and Speech Communication Laboratory, Graz University of Technology, Graz, Austria
{martin.ratajczak,pernkopf}@tugraz.at
[2] Learning and Adaptive Systems Group, ETH Zurich, Zurich, Switzerland
sebastian.tschiatschek@inf.ethz.ch

Abstract. We introduce both joint training of neural higher-order linear-chain conditional random fields (NHO-LC-CRFs) and a new structured regularizer for sequence modelling. We show that this regularizer can be derived as lower bound from a mixture of models sharing parts, e.g. neural sub-networks, and relate it to ensemble learning. Furthermore, it can be expressed explicitly as regularization term in the training objective.

We exemplify its effectiveness by exploring the introduced NHO-LC-CRFs for sequence labeling. Higher-order LC-CRFs with linear factors are well-established for that task, but they lack the ability to model non-linear dependencies. These non-linear dependencies, however, can be efficiently modeled by neural higher-order input-dependent factors. Experimental results for phoneme classification with NHO-LC-CRFs confirm this fact and we achieve state-of-the-art phoneme error rate of 16.7% on TIMIT using the new structured regularizer.

Keywords: Structured regularization · Ensemble learning · Additive mixture of experts · Neural higher-order conditional random field

1 Introduction

Overfitting is a common and challenging problem in machine learning. It occurs when a learning algorithm overspecializes to training samples, i.e. irrelevant or noisy information for prediction is learned or even memorized. Consequently, the learning algorithm does not generalize to unseen data samples. This results in large test error, while obtaining small training error. A common assumption is that complex models are prone to overfitting, while simple models have limited predictive expressiveness. Therefore a trade-off between model complexity and predictive expressiveness needs to be found. Usually, a penalty term for

This work was supported by the Austrian Science Fund (FWF) under the project number P25244-N15. Furthermore, we acknowledge NVIDIA for providing GPU computing resources.

A. Appice et al. (Eds.): ECML PKDD 2015, Part I, LNAI 9284, pp. 168–183, 2015.
DOI: 10.1007/978-3-319-23528-8_11

the model complexity is added to the training objective. This penalty term is called *regularization*. Many regularization techniques have been proposed, e.g. in parameterized model priors on individual weights or priors on groups of weights like the l_1-norm and l_2-norm are commonly used.

Recently, dropout [12] and dropconnect [33] have been proposed as regularization techniques for neural networks. During dropout training, input and hidden units are randomly canceled. The cancellation of input units can be interpreted as a special form of input noising and, therefore, as a special type of data augmentation [18,32]. During dropconnect training, the connections between the neurons are dropped [33]. Dropout and dropconnect can be interpreted as mixtures of neural networks with different structures. In this sense, dropout and dropconnect have been interpreted as ensemble learning techniques. In ensemble learning, many different classifiers are trained independently to make the same predictions, i.e. ensembles of different base classifiers. For testing, the predictions of the different classifiers are combined. In the dropout and dropconnect approaches, the mixture of models is trained and utilized for testing. Recently, *pseudo-ensemble learning* [1] has been suggested as a generalization of dropout and dropconnect. In pseudo-ensemble learning, a mixture of child models generated by perturbing a parent model is considered.

We propose a generalization of pseudo-ensemble learning. We introduce a mixture of models which share parts, e.g. neural sub-networks, called *shared-ensemble learning*. The difference is that in shared-ensemble learning, there is no parent model from which we generate child models. The models in the shared-ensemble can be different, but share parts. This is in contrast to traditional ensembles which typically do no share parts. Based on shared-ensembles, we derive a new regularizer as lower bound of the conditional likelihood of the mixture of models. Furthermore, this regularizer can be expressed explicitly as regularization term in the training objective. In this paper, we apply shared-ensemble learning to *linear-chain conditional random fields (LC-CRFs)* [13] in a sequence labeling task, derive a structured regularizer and demonstrate its advantage in experiments. LC-CRFs are established models for sequence labeling [7,35], i.e. we assign some given input sequence $\mathbf{x}$, e.g. a time series, to an output label sequence $\mathbf{y}$.

First-order LC-CRFs typically consist of transition factors, modeling the relationship between two consecutive output labels, and local factors, modeling the relationship between input observations (usually a sliding window over input frames) and one output label. But CRFs are not limited to such types of factors: *Higher-order LC-CRFs (HO-LC-CRFs)* allow for arbitrary input-independent (such factors depend on the output labels only) [35] and input-dependent (such factors depend on both the input and output variables) higher-order factors [16,23]. That means both types of factors can include more than two output labels. Clearly, the Markov order of the largest factor (on the output side) dictates the order of LC-CRFs.

It is common practice to represent the higher-order factors by linear functions which can reduce the model's expressiveness. To overcome this limitation,

a widely used approach is to represent non-linear dependencies by parametrized models and to learn these models from data. Mainly kernel methods [14] and *neural models* [15,19,20,22,24] have been suggested to parametrize *first-order* factors in LC-CRFs, i.e. mapping several input frames to one output label. In summary, most work in the past focused either on (a) higher-order factors represented by simple linear models, or (b) first-order input-dependent factors represented by neural networks. In this work, we explore joint-training of neural *and* higher-order input-dependent factors in LC-CRFs.

Unfortunately, higher-order CRFs significantly increase the model complexity and, therefore, are prone to overfitting. To avoid overfitting, the amount of training data has to be sufficiently large or, alternatively, regularizers can be utilized. In this work, we apply the structured regularizer derived from the shared-ensemble framework to higher-order CRFs and demonstrate its effectiveness.

Our main contributions are:

1. We propose *shared-ensemble learning* as a generalization of pseudo-ensemble learning, i.e. a mixture of models which share parts, e.g. neural sub-networks.
2. From this framework we derive a new regularizer for higher-order sequence models. By lower-bounding the conditional likelihood of the mixture of models, we can explicitly express the regularization term in the training objective.
3. Furthermore, we introduce *joint-training* of *neural higher-order input-dependent factors* in LC-CRFs depending on both sub-sequences of the input and the output labels. These factors are represented by *individual multi-layer perceptron (MLP) networks*.
4. We present experimental results for phoneme classification. NHO-LC-CRFs with the proposed regularizer achieve state-of-the-art performance of 16.7% phone error rate on the TIMIT phoneme classification task.

The remainder of this paper is structured as follows: In Section 2 we briefly review related work. In Section 3 we introduce the NHO-LC-CRF model. Section 4 provides details on the structured regularizer. In Section 5 we evaluate our model on the TIMIT phoneme classification task. Section 6 concludes the paper.

2 Related Work

Dropout applied to the input has been formalized for some linear and log-linear models [18,32]. Assuming a distribution of the dropout noise, an analytical form of the dropout technique has been presented. The training objective has been formulated as the expectation of the loss function under this distribution. Furthermore, this objective has been reformulated as the original objective and an additive explicit regularization.

As mentioned before, HO-LC-CRFs have been applied to sequence labeling in tagging tasks, in handwriting recognition [23,35] and large-scale machine translation [16]. In these works, higher-order factors have not been modeled by neural networks which is the gap we fill. However, first-order factors have

been already modeled by several types of neural networks. Conditional neural fields (CNFs) [20] and multi-layer CRFs [22], propose a direct method to optimize MLP networks and LC-CRFs under the conditional likelihood criterion based on error back-propagation. Another approach is to pre-train an unsupervised representation with a deep belief network, transform it into an MLP network and finally fine-tune the network in conjunction with the LC-CRF [5]. *Hidden-unit conditional random fields* (HU-CRFs) [19] replace local factors by *discriminative RBMs* (DRBMs) [15], CNN triangular CRFs [34] by convolutional neural networks (CNNs) and *context-specific deep CRFs* [24] by sum-product networks [6,21] which can be interpreted as *discriminative deep Gaussian mixture models* generalizing discriminative Gaussian mixture models to multiple layers of hidden variables.

In more detail, we contrast our work from [5]: First, although formulated quite general that work focused on neural first-order factors in contrast to neural higher-order factors in our work. Second, they used one shared neural network for all factors in contrast to individual neural networks as in our case. Third, that work utilized unsupervised pre-training as initialization of their MLP network. We jointly train the NHO-LC-CRF and we improved the classification performance using the new structured regularizer. This work is an extension of [25] which focused on discriminative pre-training of neural higher-order factors to produce rich non-linear features. A linear higher-order LC-CRFs is subsequently trained on these features. In contrast, here we train jointly the NHO-LC-CRF. Furthermore, we introduce the new structured regularizer and show its relation to mixture models and ensemble learning.

Finally, in computer vision higher-order factors in Markov random fields [26] and conditional random fields [9,17,30] are much more common than in sequence labeling. Most of that work focus on higher-order factors represented by products of experts [11]. Typically, approximate inference such as belief propagation or a sampling method is utilized.

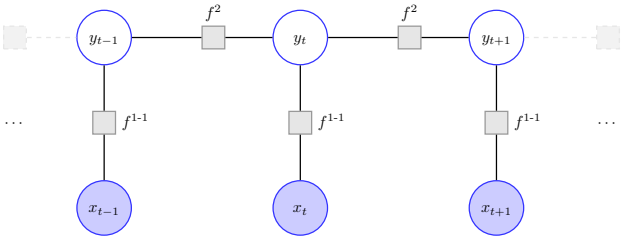
3 Higher-Order Conditional Random Fields

We consider HO-LC-CRFs for sequence labeling. The HO-LC-CRF defines a conditional distribution

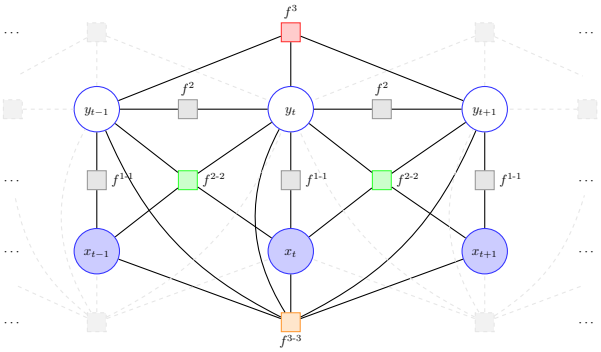
$$p^{CRF}(\mathbf{y}|\mathbf{x}) = \frac{1}{Z(\mathbf{x})} \prod_{t=1}^{T} \prod_{n=1}^{N} \Phi_t(\mathbf{y}_{t-n+1:t}; \mathbf{x}), \tag{1}$$

for an output sequence $\mathbf{y}$ of length T given an input sequence $\mathbf{x}$ of length T where $\Phi_t(\mathbf{y}_{t-n+1:t}; \mathbf{x})$ are non-negative factors that can depend on the label sub-sequence $\mathbf{y}_{t-n+1:t}$ and the whole input sequence $\mathbf{x}$, and where $Z(\mathbf{x})$ is an input-dependent normalization computed as

$$Z(\mathbf{x}) = \sum_{\mathbf{y}} \prod_{t=1}^{T} \prod_{n=1}^{N} \Phi_t(\mathbf{y}_{t-n+1:t}; \mathbf{x}). \tag{2}$$



(a) LC-CRF



(b) HO-LC-CRF

Fig. 1. Factor graph of LC-CRFs using (a) input-dependent uni-gram features $f^{1\text{-}1}$ and bi-gram transition features f^2 (typical) and (b) additionally 3-gram features f^3 as well as input-dependent features $f^{2\text{-}2}$ and $f^{3\text{-}3}$.

An $(\mathrm{N}-1)^{\mathrm{th}}$-order CRF models label sub-sequences of maximal span N in the corresponding factors. The factors $\Phi_t(\mathbf{y}_{t-n+1:t}; \mathbf{x})$ are assumed to be given in log-linear form, i.e.

$$\Phi_t(\mathbf{y}_{t-n+1:t}; \mathbf{x}) = \exp\left(\sum_k \mathbf{w}_k^{t,n} \mathbf{f}_k(\mathbf{y}_{t-n+1:t}; t, \mathbf{x})\right), \tag{3}$$

where $\mathbf{f}_k(\mathbf{y}_{t-n+1:t}; t, \mathbf{x})$ are arbitrary vector-valued and (possibly) position-dependent feature functions and $\mathbf{w}_k^{t,n}$ are the weights. These functions can be any functions ranging from simple indicator functions, linear functions, up to functions computed using neural networks as we have in this work. We distinguish the following types of feature functions:

– n**-Gram Input-Independent Features.** These features are observation-independent, i.e. $\mathbf{f}_k(\mathbf{y}_{t-n+1:t}; t, \mathbf{x}) = \mathbf{f}^n(\mathbf{y}_{t-n+1:t})$. Each entry of the vectors corresponds to the indicator function of a certain label sub-sequence $\mathbf{a}_i$, i.e. $\mathbf{f}^n(\mathbf{y}_{t-n+1:t}) = [\mathbf{1}(\mathbf{y}_{t-n+1:t} = \mathbf{a}_1), \mathbf{1}(\mathbf{y}_{t-n+1:t} = \mathbf{a}_2), \ldots]^T$. Typically $\mathbf{a}_1, \mathbf{a}_2, \ldots$ enumerate all possible label sub-sequences of length n. These transition functions are denoted as f^n in Figure 1.

– *m-n*-**Gram Input-Dependent MLP Features.** These features generalize local factors to longer label sub-sequences. In this way, these feature functions can depend on the label sub-sequence $\mathbf{y}_{t-n+1:t}$ and an input subsequence of $\mathbf{x}$, i.e. $\mathbf{f}^{m-n}(\mathbf{y}_{t-n+1:t}; t, \mathbf{x}) = [\mathbf{1}(\mathbf{y}_{t-n+1:t} = \mathbf{a}_1)\, \mathbf{g}^m(\mathbf{x}, t), \ldots]^T$ where $\mathbf{g}^m(\mathbf{x}, t)$ is an arbitrary function. This function maps an input subsequence into a new feature space. In this work, we choose to use MLP networks for this function being able to model complex interactions among the variables. More specific, the hidden activations of the last layer $\mathbf{h}^m(\mathbf{x}, t)$ of the MLP network are used, i.e. $\mathbf{g}^m(\mathbf{x}, t) = \mathbf{h}^m(\mathbf{x}, t)$. We call these features *m-n-gram MLP features*. They are denoted as $f^{m\text{-}n}$ in Figure 1, assuming that they only depend on input-output sub-sequences. Although possible, we do not consider the full input sequence $\mathbf{x}$ to counteract overfitting, but only use a symmetric and centered contextual window of length m around position t or time interval $t - n + 1 : t$. Exemplary, in case of two labels and four input sub-sequences we include the inputs from time interval $t-2 : t+1$. An important extension to prior work [5] is that the *m-n*-gram MLP features are modeled by individual networks to represent different non-linear interdependences between input and output sub-sequences.

Figure 1 show two LC-CRFs as factor graph. A typical LC-CRF as shown in the top of the Figure 1 consists of input-dependent uni-gram features f^{1-1} and input-independent bi-gram features f^2. In this work, we consider a rarely used extension using higher-order input-dependent *m-n*-gram features, for example f^{3-3}, shown in the bottom of Figure 1.

The benefit of input-dependent higher-order factors for phoneme classification is substantiated by the fact that the spectral properties of phonemes are strongly influenced by neighboring phonemes. This is illustrated in Figure 2. In conventional speech recognition systems, this well-known fact is tackled by introducing meta-labels in form of tri-phone models [10]. Input-dependent higher-order factors in HO-LC-CRF support this by mapping an input sub-sequence to an output sub-sequence, i.e. several output labels, without introducing meta-labels. Further in HO-LC-CRF, we are able to model arbitrary mappings from input sub-sequences of length m to output sub-sequences of length n, i.e we can also model mono-phones, bi-phones and tri-phones within the same model.

3.1 Parameter Learning

Parameters $\mathbf{w} = \{\mathbf{w}_k^{t,n} \mid \forall k, t, n\}$ are optimized to maximize the conditional log-likelihood of the training-data, i.e.

$$\mathcal{F}(\mathbf{w}) = \sum_{j=1}^{J} \log p(\mathbf{y}^{(j)} | \mathbf{x}^{(j)}), \tag{4}$$

where $((\mathbf{x}^{(1)}, \mathbf{y}^{(1)}), \ldots, (\mathbf{x}^{(J)}, \mathbf{y}^{(J)}))$ is a collection of J input-output sequence pairs drawn i.i.d. from some unknown data distribution. The partial derivatives of (4) with respect to the weights $\mathbf{w}_k^{t,n}$ can be computed as described in [23,35].

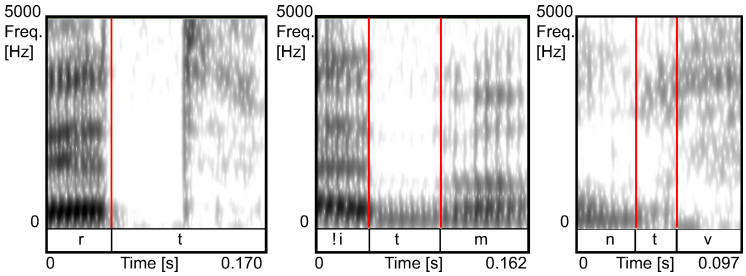


Fig. 2. Three realizations of word-final /t/ in spontaneous Dutch. Left panel: Realization of /rt/ in *gestudeerd* 'studied'. Middle panel: Realization of /ˈɛitm/ in *leeftijd mag* 'age is allowed'. Right panel: Realization of /ntv/ in *want volgens* 'because according' [28].

The weights are shared over time $\mathbf{w}_k^{t,n} = \mathbf{w}_k^n$ as we use time-homogeneous features. To perform parameter learning using gradient ascent all marginal posteriors of the form $p(\mathbf{y}_{t-n+1:t}|t, \mathbf{x}^{(j)})$ are required. These marginals can be efficiently computed using the *forward-backward algorithm* [23,35]. The algorithm can be easily extended to CRFs of order $(N-1) > 2$. However, for simplicity and as we are targeting GPU platforms, we choose another approach. As we describe in more detail in Section 3.2, we compute the conditional log-likelihood by computing just the forward recursion. Then we utilize back-propagation [27] as common in typical neural networks to compute their gradients. The conditional likelihood, the forward recursion and the corresponding gradients are computed using Theano [2], a mathematical expression compiler for GPUs and automatic differentiation toolbox.

3.2 Forward Algorithm for 2nd-Order CRFs

The main ingredient needed for applying the back-propagation algorithm is the forward recursion and the computation of the normalization constant. For a given input-output sequence pair $(\mathbf{x}, \mathbf{y})$, the forward recursion is given in terms of quantities $\alpha_t(\mathbf{y}_{t-1:t})$ that are updated according to

$$\alpha_t(\mathbf{y}_{t-1:t}) = \Phi_t(y_t; \mathbf{x})\Phi_t(\mathbf{y}_{t-1:t}; \mathbf{x}) \sum_{y_{t-2}} \Phi_t(\mathbf{y}_{t-2:t}; \mathbf{x})\alpha_{t-1}(\mathbf{y}_{t-2:t-1}). \tag{5}$$

The recursion is initialized as

$$\alpha_2(\mathbf{y}_{1:2}) = \Phi_2(y_2; \mathbf{x})\Phi_1(\mathbf{y}_{1:2}; \mathbf{x})\Phi_1(y_1; \mathbf{x}). \tag{6}$$

Finally, the normalization constant can be computed as

$$Z(\mathbf{x}) = \sum_{\mathbf{y}_{T-1:T}} \alpha_T(\mathbf{y}_{T-1:T}). \tag{7}$$

The most probable label sequence can be found by the Viterbi algorithm generalized to HO-LC-CRFs: The summation in the forward recursion is exchanged by the maximum operation, i.e. quantities $\hat{\alpha}_t(\mathbf{y}_{t-1:t})$ are computed as

$$\hat{\alpha}_t(\mathbf{y}_{t-1:t}) = \Phi_t(y_t; \mathbf{x}) \Phi_t(\mathbf{y}_{t-1:t}; \mathbf{x}) \max_{y_{t-2}} \Phi_t(\mathbf{y}_{t-2:t}; \mathbf{x}) \hat{\alpha}_{t-1}(\mathbf{y}_{t-2:t-1}). \tag{8}$$

At the end of the recursion, we identify the most probable state at the last position and apply back-tracking. For details and for time complexities we refer to [23,35].

4 Structured Regularizer

As mentioned in the introduction, NHO-LC-CRFs have high predictive expressiveness but are prone to overfitting. To fully exploit the potential of these models, special regularization techniques must be applied. Therefore, on top of the NHO-LC-CRF, we add a new structured regularizer. In the following, we derive this regularizer in a quite general form based on additive mixture of experts [3]. Our derivation is based on a single training sample $(\mathbf{x}, \mathbf{y})$, the generalization to multiple samples is straightforward.

A mixture of models, i.e. additive mixture of experts, is defined as

$$\log\ p(\mathbf{y}|\mathbf{x}) = \log \sum_{M \in \mathcal{M}} p(\mathbf{y}, M|\mathbf{x}), \tag{9}$$

where $\mathcal{M}$ is the set of models. We assume that the models in $\mathcal{M}$ have K *shared parts* $S_1, \ldots, S_K$, e.g. neural sub-networks. We consider the special case $\mathcal{M} = \{M_{S_1,\ldots,S_K}, M_{S_1}, \ldots, M_{S_K}\}$, where $M_{S_1,\ldots,S_K}$ is the *combination model* which contains all shared parts and M_{S_i} are *sub-models* containing the corresponding parts S_i. The intuition behind this model choice is the following: Shared parts in the combination model should not rely on the predictions of the other parts. Therefore, the sub-models should produce good predictions by itself. This approach improves robustness by counteracting co-adaptation comparable to dropout training in neural networks.

Expanding Equation 9 yields

$$\log\ p(\mathbf{y}|\mathbf{x}) = \log \Big(p(\mathbf{y}, M_{S_1,\ldots,S_K}|\mathbf{x}) + \sum_{M_S \in \mathcal{M}_S} p(\mathbf{y}, M_S|\mathbf{x}) \Big), \tag{10}$$

where the first term in the logarithm is the conditional joint probability of $\mathbf{y}$ and the combination model $M_{S_1,\ldots,S_K}$ and the sum is over the conditional joint probabilities of $\mathbf{y}$ and the sub-models in $\mathcal{M}_S = \{M_{S_1}, \ldots, M_{S_K}\}$. By the chain-rule $p(\mathbf{y}, M|\mathbf{x}) = p(\mathbf{y}|\mathbf{x}, M)\ p(M|\mathbf{x})$ and Jensen's inequality, we obtain a lower bound to the log-likelihood as

$$\log \sum_{M \in \mathcal{M}} p(\mathbf{y}|\mathbf{x}, M)\ p(M|\mathbf{x}) \geq \sum_{M \in \mathcal{M}} p(M|\mathbf{x})\ \log\ p(\mathbf{y}|\mathbf{x}, M), \tag{11}$$

where $\sum_{M \in \mathcal{M}} p(M|\mathbf{x}) = 1$. By lower-bounding the log-likelihood we reformulated the additive mixture of experts as a product of experts, i.e. summation in log-space. By assuming that the model prior satisfies $p(M|\mathbf{x}) = p(M)$, i.e. the prior is independent of the sample $\mathbf{x}$, we obtain $\sum_{M \in \mathcal{M}} p(M) \log p(\mathbf{y}|\mathbf{x}, M)$. To this end, we can rewrite our lower bound as

$$p(M_{S_1,\ldots,S_K}) \log p(\mathbf{y}|\mathbf{x}, M_{S_1,\ldots,S_K}) + \sum_{M_S \in \mathcal{M}_S} p(M_S) \log p(\mathbf{y}|\mathbf{x}, M_S). \tag{12}$$

We apply this result to our sequence labeling model introduced in Section 3. We utilize the MLP networks for the different sub-sequences as sub-models M_S and the NHO-LC-CRF as the combination model $M_{S_1,\ldots,S_K}$. Assuming a prior probability of λ for the combination model, i.e. $p(M_{S_1,\ldots,S_K}) = \lambda$, and a uniform model prior $p(M_S) = (1-\lambda)/|\mathcal{M}_S|$ for the label sub-sequence models, the final training objective over sequences including the structured regularizer is

$$\mathcal{F}(\mathbf{w}) = \sum_{j=1}^{J} \Big(\lambda \log p^{CRF}(\mathbf{y}^{(j)}|\mathbf{x}^{(j)}) + (1-\lambda) \frac{1}{|\mathcal{M}_S|} \sum_n \log R^n(\mathbf{y}^{(j)}|\mathbf{x}^{(j)}) \Big), \tag{13}$$

where $(\mathbf{x}^{(j)}, \mathbf{y}^{(j)})$ is the j^{th} training sample. The regularizers $R^n(\mathbf{y}|\mathbf{x})$ for the corresponding label sub-sequences are defined as

$$\log R^n(\mathbf{y}|\mathbf{x}) = \sum_{t=n:T} \log p^{\text{MLP}}(\mathbf{y}_{t-n+1:t}|t, m, \mathbf{x}), \tag{14}$$

where the conditional probabilities of the corresponding label sub-sequences are

$$p^{\text{MLP}}(\mathbf{y}_{t-n+1:t}|t, m, \mathbf{x}) = \frac{\exp\left(\mathbf{w}_n^T \mathbf{f}^{m-n}(\mathbf{y}_{t-n+1:t}; t, \mathbf{x})\right)}{Z_n^{\text{MLP}}(\mathbf{x})} \tag{15}$$

and the normalization constants of the MLP networks are

$$Z_n^{\text{MLP}}(\mathbf{x}) = \sum_{\mathbf{y}_{t-n+1:t}} \exp\left(\mathbf{w}_n^T \mathbf{f}^{m-n}(\mathbf{y}_{t-n+1:t}; t, \mathbf{x})\right). \tag{16}$$

This means the sub-models are MLP networks sharing the MLP features f^{m-n} with the NHO-LC-CRF, the combination model. The trade-off parameter λ balances the importance of the sequence model against the other sub-models.

For testing we drop the regularizer and find the most probable sequence by utilizing the Viterbi algorithm for NHO-LC-CRFs as described in Section 3.2.

5 Experiments

We evaluated the performance of the proposed models on the TIMIT phoneme classification task. We compared isolated phone classification (without label context information) with MLP networks to phone labeling with neural HO-LC-CRFs. This comparison substantiates the effectiveness of joint-training of neural higher-order factors. Furthermore, we show performance improvements using our introduced structured regularizer during joint-training.

5.1 TIMIT Data Set

The TIMIT data set [36] contains recordings of 5.4 hours of English speech from 8 major dialect regions of the United States. The recordings were manually segmented at phone level. We use this segmentation for phone classification. Note that phone classification should not be confused with phone recognition [10] where no segmentation is provided. We collapsed the original 61 phones into 39 phones. All frames of Mel-frequency cepstral coefficients (MFCC), delta and double-delta coefficients of a phonetic segment are mapped into one feature vector. Details are presented in [8]. The task is, given an utterance and a corresponding segmentation, to infer the phoneme within every segment. The data set consists of a training set, a development set (dev) and a test set (test), containing 140.173, 50.735 and 7.211 phonetic segments, respectively. The development set is used for parameter tuning.

5.2 Experimental Setup

In all experiments, input features were normalized to zero mean and unit variance. Optimization of our models was in all cases performed using stochastic gradient ascent using a batch-size of one sample. An ℓ_2-norm regularizer on the model weights was used. We utilized early stopping determined on the development data set. We trained for up to 500 epochs. However, in most cases less iterations have been required. The proposed model was entirely trained on NVIDIA GPUs using Theano [2], a mathematical expression compiler for GPUs and automatic differentiation toolbox. The classification performance is measured by phone error rate (PER), i.e. Hamming distance between the reference and predicted label sequence for all test samples.

5.3 Labeling Results Using Only MLP Networks

In the first experiment, we trained MLP networks with a single hidden layer to predict the phone label of each segment. We tuned the number of hidden units $H \in \{100, 150, 200, 300, 400, 500\}$ and their activation functions, i.e. rectifier and tanh. Furthermore, we analyzed the effect of the number of input segments, i.e. we used the current segment and three or five surrounding segments centered at the current position index as input. Results in Table 1 (only a sub-set is reported) show that more hidden units result in better performance. For tanh activations, the best performance of 21.0% is achieved with $m = 3$ input segments and using $H = 500$ hidden units. More input segments reduced the performance. In preliminary experiments, we found that more than one hidden layer decreased the performance. Therefore, we used in the following experiments tanh activations and one hidden layer.

5.4 Labeling Results Using LC-CRFs with Linear or Neural Higher-Order Factors

Experiments with linear HO-LC-CRFs as shown in Table 2 reveal that classification performance degrades with linear 3-3 gram factor. The best performance

Table 1. Isolated Phone Classification using MLP networks ($n = 1$) with different number of hidden units H and lengths of the contextual input window m. The classification performance is measured by phone error rate (PER).

		rectifier		tanh	
H	m	dev	test	dev	test
100	1	23.2	23.7	23.3	24.1
100	3	22.0	23.4	22.3	22.9
100	5	22.6	22.9	23.8	24.4
150	1	22.6	23.0	22.6	23.0
150	3	21.4	22.2	21.8	22.3
150	5	22.4	22.9	23.2	23.9

		rectifier		tanh	
H	m	dev	test	dev	test
200	1	22.5	23.2	22.4	22.6
200	3	21.3	21.8	21.4	22.6
200	5	22.3	22.7	22.7	22.9
500	1	22.1	22.6	22.1	22.9
500	3	**20.9**	22.1	**20.6**	**21.0**
500	5	22.3	22.7	21.9	22.7

Table 2. Linear higher-order CRFs. All results with $m = 1$ and $n = 1$ already include input-independent 2-gram factors.

m=n	1	+ 2	+ 3
dev	25.8	**20.4**	20.7
test	25.9	**20.5**	21.6

of 20.5% is achieved with factors up to order $n = m = 2$. The plus sign indicates additional higher-order factors on top to the ones from previous columns in the table, i.e. the model of column +2 includes the linear factors $\{f^1, f^{1-1}, f^2, f^{2-2}\}$.

In the next set of experiments, we consider LC-CRFs with neural input-dependent higher-order factors and we will show their effectiveness in contrast to their linear counterparts in Table 2. In Table 3, we explore the combination of higher-order factors up to the order $n = m = 3$ as described in Section 3. By including more higher-order factors in the first column of Table 3, the classification performance improves for the baseline using only l_2 regularization only. For the baseline, we tuned the learning rate 0.01, 0.001, 0.0001 and $l2$ regularizer trade-off-parameter 0.1, 0.01, 0.001, 0.0001 and report the best observed performance. The best result of 17.7% is achieved with 150 hidden units and factors up to order $n = m = 3$ which is significantly better than the best performance of 20.5% with linear factors.

Furthermore, we tested our new structured regularizer with factors up to order $n = m = 3$ for various trade-off parameters $\lambda \in \{0.01, 0.3, 0.6\}$ as shown in Table 3. We used fixed a learning rate of 0.001 and l_2 regularizer of 0.001. We achieved the best performance of 16.8% with factors up to order $m = n = 3$ and a trade-off parameter of 0.3. We further tuned the trade-off parameter $\lambda \in \{0.01, 0.05, 0.1, 0.2, 0.3, 0.4, 0.6, 0.95, 1.0\}$ for different number of hidden units $H \in \{100, 150, 200\}$. This is shown in the Figure 3. For different network sizes, we observe a clear optimum with respect to the trade-off parameter

Table 3. Neural higher-order LC-CRF without and with our structured regularizer. The order $m = n$ of the higher order factors is examined. Furthermore, the effectiveness of the new structured regularizer is demonstrated for factors up to order $m = n = 3$. All results with $m = 1$ and $n = 1$ already include input-independent 2-gram factors. In all experiments, the number of hidden units is $H = 150$ and one hidden layer.

		no reg.		$\lambda = 0.6$		$\lambda = 0.3$		$\lambda = 0.01$	
	$m = n$	dev	test	dev	test	dev	test	dev	test
	1	21.2	21.5	21.7	22.4	21.7	21.9	25.7	26.4
+	2	17.6	18.3	17.9	18.3	17.5	18.0	19.8	20.4
+	3	**17.9**	**17.7**	16.9	17.5	**16.6**	**16.8**	18.5	19.2

λ and a margin to the baseline results without the regularizer, i.e. $\lambda = 1.0$, which we indicated by a dotted line in Figure 3. By this additional tuning, we improved the result further and achieved the best overall performance of 16.7% with factors up to order $m = n = 3$ and a trade-off parameter of 0.1.

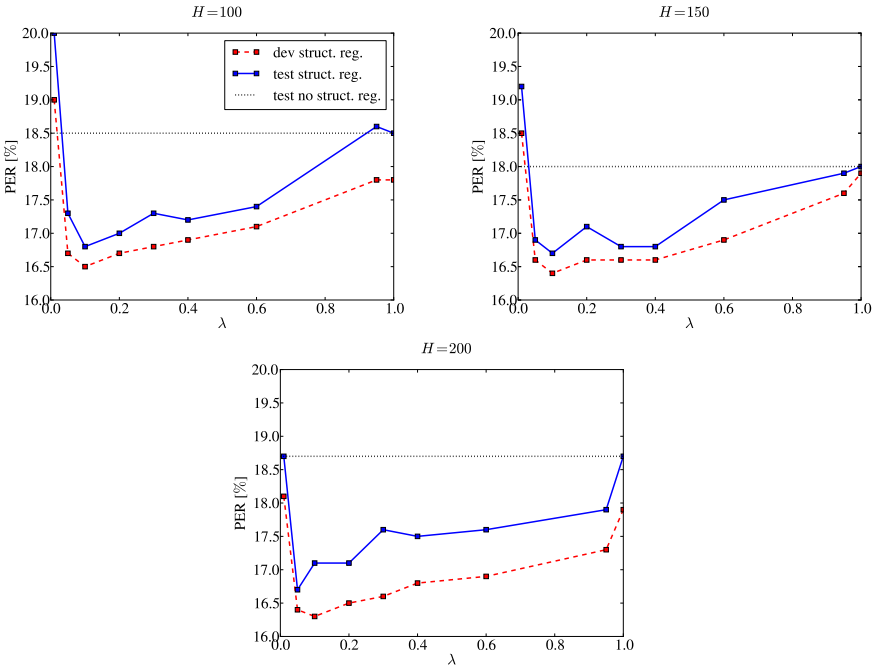


Fig. 3. Performance using the structured regularizer for various trade-off parameters λ. The number of hidden units $H \in \{100, 150, 200\}$ in the neural higher-order factors varies in the different plots. Baseline results without the regularizer $\lambda = 1.0$ are indicated by a dotted line.

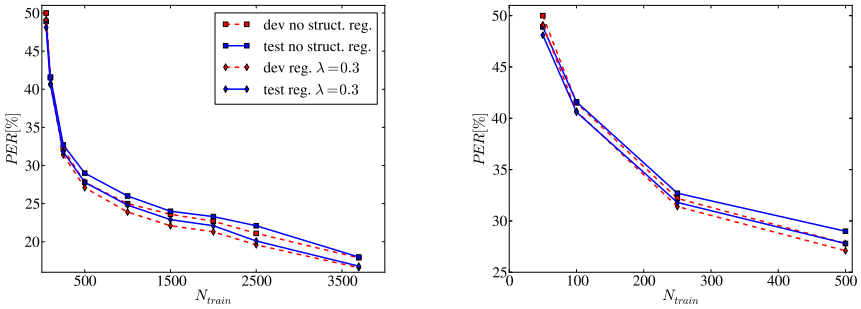


Fig. 4. Test performance for (left) various training set sample sizes with and without our structured regularizer and (right) its zoomed presentation to illustrate the effectiveness for small training set sample sizes.

In additional experiments, we explored the performance for varying numbers of training samples N_{train}. We fixed the number of hidden units $H = 150$, trade-off parameter $\lambda = 0.3$, learning rate of 0.001 and l_2 regularizer of 0.001. Figure 4 shows the effectiveness of our structured regularizer for small and full training sample set, i.e. we are able to outperform the same model without the structured regularizer by a large margin over the range of used training samples. For small sample sizes, the margin between the baseline performance results and the one with the structured regularizer decreased slightly.

Finally, we compare our best result to other state-of-the-art methods based on MFCC features as shown in Table 4. Using the code of [20] we tested CNFs

Table 4. Summary of labeling results. Results marked by (†) are from [31], by (††) are from [29], by (†††) are from [8], by (††††) are from [24], and by (*) are from [4]. Performance measure: Phone error rate (PER) in percent.

Model	PER [%]
GMMs ML††	25.9
HCRFs†	21.5
Large-Margin GMM††	21.1
Heterogeneous Measurements†††	21.0
CNF	20.67
Linear HO-LC-CRF	20.5
GMM+LC-CRF (1st order)††††	22.10
CS-DCRF+MEMM (8th order)††††	22.15
CS-DCRF+LC-CRF (1st order)††††	19.95
Hierarchical Large-Margin GMMs*	**16.7**
NHO-LC-CRF	**17.7**
+ structured regularizer	**16.7**

with 50, 100 and 200 hidden units as well as one and three input segments. We achieved the best result with 100 hidden units and one segment as input. Furthermore, hierarchical large-margin GMMs achieve a performance of 16.7% and outperform most other referenced methods but exploit human-knowledge and committee techniques. However, our best model, the HO-LC-CRF augmented by m-n-gram MLP factors using our new structured regularizer achieves a performance of 16.7% without human-knowledge and is outperforming most of the state-of-the-art methods.

6 Conclusion

We considered NHO-LC-CRFs for sequence labeling. While these models have high predictive expressiveness, they are prone to overfitting due to their high model complexity. To avoid overfitting, we applied a novel structured regularizer derived from the proposed shared-ensemble framework. We show that this regularizer can be derived as lower bound from a mixture of models sharing parts of each other, e.g. neural sub-networks. We demonstrated the effectiveness of this structured regularizer in phoneme classification experiments. Furthermore, we experimentally confirmed the importance of non-linear representations in form of neural higher-order factors in LC-CRFs in contrast to linear ones. In TIMIT phoneme classification, we achieved state-of-the-art phoneme error rate of 16.7% using the NHO-LC-CRFs equipped with our proposed structured regularizer. Future work includes testing of different types of neural networks.

References

1. Bachman, P., Alsharif, O., Precup, D.: Learning with pseudo-ensembles. In: Advances in Neural Information Processing Systems, pp. 3365–3373 (2014)
2. Bergstra, J., Breuleux, O., Bastien, F., Lamblin, P., Pascanu, R., Desjardins, G., Turian, J., Warde-Farley, D., Bengio, Y.: Theano: a CPU and GPU math expression compiler. In: Proceedings of the Python for Scientific Computing Conference (SciPy) (2010). Oral Presentation
3. Bishop, C.M.: Pattern Recognition and Machine Learning. Springer (2006)
4. Chang, H.A., Glass, J.R.: Hierarchical large-margin Gaussian mixture models for phonetic classification. In: Workshop on Automatic Speech Recognition and Understanding (ASRU), pp. 272–277 (2007)
5. Do, T.M.T., Artières, T.: Neural conditional random fields. In: International Conference on Artificial Intelligence and Statistics (AISTATS), pp. 177–184 (2010)
6. Gens, R., Domingos, P.: Learning the structure of sum-product networks. In: International Conference on Machine Learning (ICML), vol. 28, pp. 873–880 (2013)
7. Gunawardana, A., Mahajan, M., Acero, A., Platt, J.C.: Hidden conditional random fields for phone classification. In: Interspeech, pp. 1117–1120 (2005)
8. Halberstadt, A.K., Glass, J.R.: Heterogeneous acoustic measurements for phonetic classification. In: EUROSPEECH, pp. 401–404 (1997)
9. He, X., Zemel, R.S., Carreira-Perpiñán, M.A.: Multiscale conditional random fields for image labeling. In: Conference on Computer Vision and Pattern Recognition (CVPR), pp. 695–703 (2004)

10. Hinton, G., Deng, L., Yu, D., Dahl, G., Mohamed, A., Jaitly, N., Senior, A., Vanhoucke, V., Nguyen, P., Sainath, T., Kingsbury, B.: Deep neural networks for acoustic modeling in speech recognition. Signal Processing Magazine, pp. 82–97 (2012)
11. Hinton, G.E.: Products of experts. In: International Conference on Artificial Neural Networks (ICANN), pp. 1–6 (1999)
12. Hinton, G.E., Srivastava, N., Krizhevsky, A., Sutskever, I., Salakhutdinov, R.: Improving neural networks by preventing co-adaptation of feature detectors. CoRR abs/1207.0580 (2012)
13. Lafferty, J., McCallum, A., Pereira, F.: Conditional random fields: probabilistic models for segmenting and labeling sequence data. In: International Conference on Machine Learning (ICML), pp. 282–289 (2001)
14. Lafferty, J., Zhu, X., Liu, Y.: Kernel conditional random fields: representation and clique selection. In: International Conference on Machine Learning (ICML), p. 64 (2004)
15. Larochelle, H., Bengio, Y.: Classification using discriminative restricted Boltzmann machines. In: International Conference on Machine Learning (ICML), pp. 536–543 (2008)
16. Lavergne, T., Allauzen, A., Crego, J.M., Yvon, F.: From n-gram-based to crf-based translation models. In: Workshop on Statistical Machine Translation, pp. 542–553 (2011)
17. Li, Y., Tarlow, D., Zemel, R.S.: Exploring compositional high order pattern potentials for structured output learning. In: Conference on Computer Vision and Pattern Recognition (CVPR), pp. 49–56 (2013)
18. Maaten, L., Chen, M., Tyree, S., Weinberger, K.Q. In: International Conference on Machine Learning (ICML), pp. 410–418 (2013)
19. van der Maaten, L., Welling, M., Saul, L.K.: Hidden-unit conditional random fields. In: International Conference on Artificial Intelligence and Statistics (AISTATS), pp. 479–488 (2011)
20. Peng, J., Bo, L., Xu, J.: Conditional neural fields. In: Neural Information Processing Systems (NIPS), pp. 1419–1427 (2009)
21. Poon, H., Domingos, P.: Sum-product networks: a new deep architecture. In: Uncertainty in Artificial Intelligence (UAI), pp. 337–346 (2011)
22. Prabhavalkar, R., Fosler-Lussier, E.: Backpropagation training for multilayer conditional random field based phone recognition. In: International Conference on Acoustics Speech and Signal Processing (ICASSP), pp. 5534–5537 (2010)
23. Qian, X., Jiang, X., Zhang, Q., Huang, X., Wu, L.: Sparse higher order conditional random fields for improved sequence labeling. In: International Conference on Machine Learning (ICML), pp. 849–856 (2009)
24. Ratajczak, M., Tschiatschek, S., Pernkopf, F.: Sum-product networks for structured prediction: context-specific deep conditional random fields. In: International Conference on Machine Learning (ICML): Workshop on Learning Tractable Probabilistic Models Workshop (2014)
25. Ratajczak, M., Tschiatschek, S., Pernkopf, F.: Neural higher-order factors in conditional random fields for phoneme classification. In: Interspeech (2015)
26. Roth, S., Black, M.J.: Fields of experts: a framework for learning image priors. In: Conference on Computer Vision and Pattern Recognition (CVPR), pp. 860–867 (2005)
27. Rumelhart, D.E., Hinton, G.E., Williams, R.J.: Parallel distributed processing: Explorations in the microstructure of cognition, vol. 1. chap. Learning Internal Representations by Error Propagation, pp. 318–362 (1986)

28. Schuppler, B.: Automatic Analysis of Acoustic Reduction in Spontaneous Speech. Ph.D. thesis, PhD Thesis, Radboud University Nijmegen, Netherlands (2011)
29. Sha, F., Saul, L.: Large margin Gaussian mixture modeling for phonetic classification and recognition. In: International Conference on Acoustics, Speech and Signal Processing (ICASSP), pp. 265–268 (2006)
30. Stewart, L., He, X., Zemel, R.S.: Learning flexible features for conditional random fields. IEEE Trans. Pattern Anal. Mach. Intell. **30**(8), 1415–1426 (2008)
31. Sung, Y.H., Boulis, C., Manning, C., Jurafsky, D.: Regularization, adaptation, and non-independent features improve hidden conditional random fields for phone classification. In: Workshop on Automatic Speech Recognition and Understanding (ASRU), pp. 347–352 (2007)
32. Wager, S., Wang, S., Liang, P.: Dropout training as adaptive regularization. In: Neural Information Processing Systems (NIPS), pp. 351–359 (2013)
33. Wan, L., Zeiler, M., Zhang, S., Cun, Y.L., Fergus, R.: Regularization of neural networks using dropconnect. In: International Conference on Machine Learning (ICML), pp. 1058–1066 (2013)
34. Xu, P., Sarikaya, R.: Convolutional neural network based triangular crf for joint intent detection and slot filling. In: Workshop on Automatic Speech Recognition and Understanding (ASRU), pp. 78–83 (2013)
35. Ye, N., Lee, W.S., Chieu, H.L., Wu, D.: Conditional random fields with high-order features for sequence labeling. In: Neural Information Processing Systems (NIPS), pp. 2196–2204 (2009)
36. Zue, V., Seneff, S., Glass, J.R.: Speech database development at MIT: Timit and beyond. Speech Communication **9**(4), 351–356 (1990)