Sequential Labeling with Online Deep Learning: Exploring Model Initialization

Gang Chen(✉), Ran Xu, and Sargur N. Srihari

Department of Computer Science and Engineering, SUNY at Buffalo, Buffalo, NY 14260, USA
{gangchen,rxu2,srihari}@buffalo.edu

Abstract. In this paper, we leverage both deep learning and conditional random fields (CRFs) for sequential labeling. More specifically, we explore parameter initialization and randomization in deep CRFs and train the whole model in a simple but effective way. In particular, we pretrain the deep structure with greedy layer-wise restricted Boltzmann machines (RBMs), followed with an independent label learning step. Finally, we re-randomize the top layer weight and update the whole model with an online learning algorithm – a mixture of perceptron training and stochastic gradient descent to estimate model parameters. We test our model on different challenge tasks, and show that this simple learning algorithm yields the state of the art results. The data and software related to this paper are available at https://github.com/ganggit/deepCRFs.

Keywords: Sequential labeling · Deep learning · Online learning · Parameter initialization

1 Introduction

Recent advances in deep learning [1,16,43] have sparked great interest in dimension reduction [15,44] and classification [16,26]. In a sense, the success of deep learning lies on learned features, which are useful for supervised/unsupervised tasks [1,4,11]. For example, the binary hidden units in the discriminative Restricted Boltzmann Machines (RBMs) [12,25] and deep belief networks (DBN) [16] can model latent features of raw data to improve classification. Unfortunately, one major difficulty in deep learning [16] is structured output prediction [31], where output space typically may have an exponential number of possible configurations. As for sequential labeling, the joint classification of all the items is also difficult because observations are of an indeterminated dimensionality and the number of possible classes is exponentially growing in the length of the sequences.

To address the sequential prediction, the architecture of recurrent neural networks (RNNs) have cycles incorporating the activations from previous time steps as input to the network to make a decision for the current input, which

P. Frasconi et al. (Eds.): ECML PKDD 2016, Part II, LNAI 9852, pp. 772–788, 2016.
DOI: 10.1007/978-3-319-46227-1_48

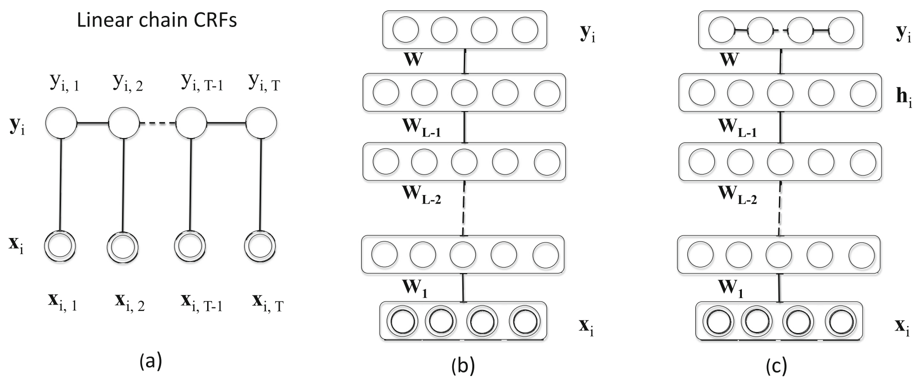

Fig. 1. (a) linear chain CRFs; (b) deep neural networks (for classification with 1 of K (encoding) vector representation); (c) our deep neural networks for sequential labeling. The two main differences between (b) and (c) are: (1) $\mathbf{y}_i$ in (c) is a label sequence, which has links between labels, while $\mathbf{y}_i$ in (b) is a single label without correlation; (2) the input of (c) is a sequence with multiple instances (or frames), while the input of (b) is an independent instance (or vector).

makes RNNs better suited for sequence labeling tasks. Long short term memory (LSTM), as an improved version of RNNs, shows good results on handwritten recognition [13]. And bi-directional LSTM trained on unsegmented sequence data has also outperformed the state of the art HMM-based system.

Another direction is to combine deep learning with CRFs for sequential labeling [9,28,33]. One of key advantages of linear CRFs can be attributed to its exploitation on context information and its structured output prediction. However, linear CRFs with the raw data as input strongly restricts its representation power for classification tasks. More recently, one trend is to generalize CRFs to learn discriminative and non-linear representations, such as kernel CRFs [23], hidden-unit CRFs [28,33] and CRFs with multilayer perceptrons [27,35]. As an alternative, some studies have trained CRFs on features learned by unsupervised deep learning [32]. Also the work in [9,33] has exploited to learn non-linear mappings by combing CRFs and neural networks. However, how to learn a better deep CRFs model is still a challenge, considering the overfitting issue with large parameter space and the non-convex objective function.

In this paper, we propose a deep model for sequential labeling, which inherits both advantages of linear chain CRFs and deep learning. Hence, our model can learn non-linear features and also handle structured output, refer to Fig. 1 for visual understanding about the model. Because the deep CRFs model is non-convex, it can be easily trapped into local minimum. Thus, how to learn a good model and generalize well in unseen dataset is still an challenge. Compared to the traditional deep CRFs [9,29,33], we take a different learning approach. We pre-train our model with stacked RBMs, followed with an independent learning step with backpropagation. Then, we re-randomize the top layer weight and optimize the whole deep model using an online learning algorithm, which is a mixture of perceptron training and stochastic gradient descent (SGD). In particular, we train the top layer with perceptron algorithm, while learning the weights of the lower layers in the deep structure with SGD. Thus, our model is more powerful than linear CRFs because the objective function learns latent non-linear features so that target labeling can be better predicted.

Our contributions: (1) an unified objective function to handle sequential data with or without patterns; (2) the independent learning step for parameter initialization and the re-randomization of the top layer weight, which we think is vital to handle overfitting and find a better local minimum in training the deep neural network; (3) the online learning approach to update model parameters, where the context parameters of CRFs in the top layer are trained with perceptron algorithm and the lower level weights of the deep neural network are updated with SGD. Lastly, we also introduce the regularization terms to handle overfitting in the deep neural network. We test our model over a range of tasks and show that it yields accuracy significantly better than the state of the art.

2 Sequential Labeling with Deep Learning

Let $D = \{\langle \mathbf{x}_i, \mathbf{y}_i \rangle\}_{i=1}^N$ be a set of N training examples. Each example is a pair of a time series $\langle \mathbf{x}_i, \mathbf{y}_i \rangle$, with $\mathbf{x}_i = \{\mathbf{x}_{i,1}, \mathbf{x}_{i,2}, ..., \mathbf{x}_{i,T_i}\}$ and $\mathbf{y}_i = \{y_{i,1}, y_{i,2}, ..., y_{i,T_i}\}$, where $\mathbf{x}_{i,t} \in \mathbb{R}^d$ is the i-th observation at time t and $y_{i,t}$ is the corresponding label (we indicate its encoded vector as $\mathbf{y}_{i,t}$ that uses a so-called 1-of-K encoding). Linear first-order CRFs [24] is a conditional discriminative model over the label sequence given the data

$$p(\mathbf{y}_i|\mathbf{x}_i) = \frac{\exp\{-E(\mathbf{x}_i, \mathbf{y}_i)\}}{Z(\mathbf{x}_i)} \tag{1}$$

where $Z(\mathbf{x}_i)$ is the partition function and $E(\mathbf{x}_i, \mathbf{y}_i)$ is the energy function given by

$$\begin{aligned} -E(\mathbf{x}_i, \mathbf{y}_i) =& \mathbf{y}_{i,1}^T \boldsymbol{\pi} + \mathbf{y}_{i,T_i}^T \boldsymbol{\tau} \\ &+ \sum_{t=1}^{T_i} (\mathbf{x}_{i,t}^T \mathbf{W} \mathbf{y}_{i,t} + \mathbf{b}^T \mathbf{y}_{i,t}) + \sum_{t=2}^{T_i} \mathbf{y}_{i,t-1}^T \mathbf{A} \mathbf{y}_{i,t} \end{aligned} \tag{2}$$

where $\mathbf{y}_{i,1}^T \boldsymbol{\pi}$ and $\mathbf{y}_{i,T_i}^T \boldsymbol{\tau}$ are the initial-state and final-state factors respectively, $\mathbf{b}^T \mathbf{y}_{i,t}$ is the bias term for labels, $\mathbf{A} \in \mathbb{R}^{K \times K}$ represents the state transition parameters and $\mathbf{W} \in \mathbb{R}^{d \times K}$ represents the parameters of the data-dependent term. Compared to linear SVMs, the linear CRFs has an additional item $\mathbf{y}_{i,t-1}^T \mathbf{A} \mathbf{y}_{i,t}$ to model the label correlation. However, one of the main disadvantages of linear CRFs is the linear dependence on the raw input data. Thus, we introduce our sequential labeling model with deep feature learning, which leverages both context information, as well as the nonlinear representations from deep learning [15].

2.1 Objective Function

Although it is possible to leverage the deep neural networks for structured prediction, its output space is explosively growing because of non-determined length of sequential data. Thus, we consider a compromised model, which combine CRFs and deep learning in an unified framework, refer Fig. (1). On the one hand, we hope the independent label prediction is as accuracy as possible via the representation learning. On the other hand, we need to handle overfitting problem in the deep network. We propose an objective function with L layers neural network structure,

$$\begin{aligned}\mathcal{L}(D;\boldsymbol{\theta},\boldsymbol{\omega}) &= -\sum_{i=1}^{N} \text{log} p(\mathbf{y}_{i,1},\ldots,\mathbf{y}_{i,T_i}|\mathbf{h}_{i,1},\ldots,\mathbf{h}_{i,T_i}) \\ &+ \frac{\lambda_1}{2}\sum_{i=1}^{N}\sum_{t=1}^{T_i}||\underbrace{f_L \circ f_{L-1} \circ \cdots \circ f_1}_{L \text{ times}}(\mathbf{x}_{i,t}) - \mathbf{y}_{i,t}||^2 \\ &+ \lambda_2||\boldsymbol{\theta}||^2 + \lambda_3||\boldsymbol{\omega}|| \end{aligned} \tag{3}$$

where $\boldsymbol{\theta}$ and $\boldsymbol{\omega}$ are the top layer parameters and lower layer ($l = \{1, ..., L-1\}$) parameters respectively, which will be explained later. The first row on the right side of the equation is from the linear CRFs in Eq. (1), but with latent features. The conditional likelihood depends respectively on $\boldsymbol{\theta}$ and the latent non-linear features $\mathbf{h}_i = \{\mathbf{h}_{i,1}, .., \mathbf{h}_{i,T_i}\}$ in the coding space, with

$$\begin{aligned}&\text{log} p(\mathbf{y}_{i,1},\ldots,\mathbf{y}_{i,T_i}|\mathbf{h}_{i,1},\ldots,\mathbf{h}_{i,T_i}) \\ =&\sum_{t=2}^{T_i}\mathbf{y}_{i,t-1}^T\mathbf{A}\mathbf{y}_{i,t} + \sum_{t=1}^{T_i}\left(\mathbf{h}_{i,t}^T\mathbf{W}\mathbf{y}_{i,t} + \mathbf{b}^T\mathbf{y}_{i,t}\right) \\ +&\mathbf{y}_{i,1}^T\boldsymbol{\pi} + \mathbf{y}_{i,T_i}^T\boldsymbol{\tau} - \log(Z(\mathbf{h}_i))\end{aligned} \tag{4}$$

and non-linear mappings $\mathbf{h}_i$ is the output with $L-1$ layers neural network, s.t.

$$\mathbf{h}_i = \underbrace{f_{L-1} \circ f_{L-2} \circ \cdots \circ f_1}_{L-1 \text{ times}}(\mathbf{x}_i) \tag{5}$$

where $\circ$ indicates function composition, and f_i is logistic function with the weight parameter $\mathbf{W}_l$ respectively for $l = \{1, .., L-1\}$, refer more details in Sect. 2.2. With a bit abuse of notation, we denote $\mathbf{h}_{i,t} = f_{1\to(L-1)}(\mathbf{x}_{i,t})$.

The least square (the second term) in the right hand side of Eq. (3) is for deep feature learning, with the top layer defined as

$$\begin{aligned}&\underbrace{f_L \circ f_{L-1} \circ \cdots \circ f_1}_{L \text{ times}}(\mathbf{x}_{i,t}) \\ =&f_{1\to L}(\mathbf{x}_{i,t}) = f_L(\mathbf{h}_{i,t}) = \mathbf{h}_{i,t}^T\mathbf{W} + \mathbf{c}^T\end{aligned} \tag{6}$$

where $\mathbf{W}$ has been defined in Eq. (4), and $\mathbf{c}$ is the bias term. Note that $\mathbf{W}$ is the same in both Eqs. 4 and 6. Hence, the second term in Eq. (3) can be thought as the independent label prediction without considering context information. Note that other objective functions such as softmax (which has the same gradient as least square, in other words, the model updating with SGD is the same as here) can be applied here too. The weighing variable λ_1 can control the balance between the first term and the second one on the RHS in Eq. (3). If $\lambda_1 \to +\infty$, then Eq. (3) can be thought as deep learning [15] for classification without context information, and it can handle the cases where outputs are independent (no significant patterns in the label sequences). If $\lambda_1 \to 0$, then Eq. (3) is the CRFs with non-linear deep feature learning, which generalizes the linear CRFs to learn non-linear deep mappings. The main purpose we incorporate the second term in our model is to introduce this parameter initialization step via the independent learning (see further). Note that we can vary λ_1 to achieve this purpose.

The last two terms in Eq. (3) are for regularization on all parameters with $\boldsymbol{\theta} = \{\mathbf{A}, \mathbf{W}, \boldsymbol{\pi}, \boldsymbol{\tau}, \boldsymbol{b}, \boldsymbol{c}\}$, and $\boldsymbol{\omega} = \{\mathbf{W}_l | l \in [1, .., L-1]\}$. We add the ℓ_2 regularization to $\boldsymbol{\theta}$ as most linear CRFs does, while we have the ℓ_1-regularized term on weight parameters $\boldsymbol{\omega}$ in the deep neural network to avoid overfitting in the learning process.

The aim of our objective function in Eq. (3) is for sequential labeling, which explores both the advantages of Markov properties in CRFs and latent representations in deep learning. Our model is different from the common deep learning structure in Fig. 1(b). Firstly, the input to our model in Fig. 1(c) is the sequential data, such as sequences with non-determined length, while the input to Fig. 1(b) is generally an instance with fixed length. Secondly, our model can predict structured outputs or label sequences, while the output in Fig. 1(b) is just one label for each instance, which is independent from each other. Note that we use the first-order CRFs for clarity in Eq. 4, which can be easily extended to the second or high-order cases. Moreover, our model is also different from other deep CRFs [9,34]. Our mixture objective function can handle sequential data with or without patterns. And we have the independent label learning (a pretraining step in our model) to learn better representations. Lastly, we use an online algorithm in our deep learning model for parameter updating, which has the potential to handle large scale dataset.

2.2 Parameter Learning

We use RBMs to initialize the weights layer by layer greedily, with Contrastive Divergence [16] (we used CD-1 in our experiments). Then we compute the sub-gradients w.r.t. $\boldsymbol{\theta}$ and $\boldsymbol{\omega}$ in the objective function, and initialize the whole deep CRFs with independent learning. Finally, we re-randomize the top layer weight and update the whole framework with online learning.

Initialization: The second term on the right hand side of Eq. (3) is from the deep belief network (DBN) for classification [16]. In our deep model, the weights from the layer 1 to $L-1$ are $\mathbf{W}_l$ respectively, for $l = \{1, .., L-1\}$, and the top layer L has weight $\mathbf{W}$. We first pre-train the L-layer deep structure with RBMs layer by layer greedily. Specifically, we think RBM is a 1-layer DBN, with weight $\mathbf{W}_1$. Thus, DBN can learn a parametric nonlinear mapping from input $\mathbf{x}$ to output $\mathbf{h}$, $f : \mathbf{x} \rightarrow \mathbf{h}$. For example, for 1-layer DBN, we have $\mathbf{h} = f_1(\mathbf{x}) = \text{logistic}(\mathbf{W_1}^T[\mathbf{x}, 1])$, where we extend $\mathbf{x} \in \mathbb{R}^d$ into $[\mathbf{x}, 1] \in \mathbb{R}^{(d+1)}$ in order to handle bias in the non-linear mapping. Note that we use the logistic function from layer 1 to $L-1$, and the top layer is a linear mapping with weight $\mathbf{W}$ in our deep neural network.

After initializing all weights in the deep neural network, we use the independent label learning by minimizing $\lambda_1 \sum_{i=1}^{N} \sum_{t=1}^{T_i} || \underbrace{f_L \circ f_{L-1} \circ \cdots \circ f_1}_{L \text{ times}}(\mathbf{x}_{i,t}) - \mathbf{y}_{i,t}||$ with L-BFGS (backpropagation is used to compute sub-gradient w.r.t. weight in each layer) to fine-tune all the weights in the deep neural network. More specifically, to learn the initial weights in the deep network, we think each instance $\mathbf{x}_{i,t} \in \mathbf{x}_i$ has its corresponding label $\mathbf{y}_{i,t} \in \mathbf{y}_i$ independently. Then, the parameters can be finetuned with backpropagation [15]. Note that it does not leverage the context information in this stage, and we will show the independent label learning step is helpful to boost the recognition accuracy in the experiments. Finally, we will update the parameters $\boldsymbol{\theta}$ and $\boldsymbol{\omega}$ in an online fashion simultaneously, which will be introduced in the following parts.

Learning: After we initialize the deep CRFs with independent learning and re-randomization, we need to minimize the final objective function $\mathcal{L}(D; \boldsymbol{\theta}, \boldsymbol{\omega})$ in Eq. (3). Because we introduce the deep neural network here for feature learning, the objective is not convex anymore. However, we can find a local minimum in Eq. (3). In our learning framework, we optimize the objective function with an online learning algorithm, by mixing perceptron training and stochastic gradient descent.

Firstly, we can calculate the (sub)gradients w.r.t. all parameters. Considering different regularization methods for $\boldsymbol{\theta}$ and $\boldsymbol{\omega}$ respectively, we can calculate gradients w.r.t. them separately. As for the parameters in the negative log likelihood in Eq. 3, we can compute the gradients w.r.t. $\boldsymbol{\theta}$ as follows

$$\frac{\partial \mathcal{L}}{\partial \mathbf{A}} = \sum_{i=1}^{N} \sum_{t=2}^{T_i} \mathbf{y}_{i,t-1}(\mathbf{y}_{i,t})^T - \boldsymbol{\gamma}_{i,t-1}(\boldsymbol{\gamma}_{i,t})^T; \tag{7a}$$

$$\frac{\partial \mathcal{L}}{\partial \boldsymbol{\pi}} = \sum_{i=1}^{N} (\mathbf{y}_{i,1} - \boldsymbol{\gamma}_{i,1}); \tag{7b}$$

$$\frac{\partial \mathcal{L}}{\partial \boldsymbol{\tau}} = \sum_{i=1}^{N} (\mathbf{y}_{i,T_i} - \boldsymbol{\gamma}_{i,T_i}); \tag{7c}$$

$$\frac{\partial \mathcal{L}}{\partial \mathbf{b}} = \sum_{i=1}^{N} \big(\sum_{t=1}^{T_i} (\mathbf{y}_{i,t} - \boldsymbol{\gamma}_{i,t}) \big); \tag{7d}$$

$$\frac{\partial \mathcal{L}}{\partial \mathbf{W}} = \sum_{i=1}^{N} \sum_{t=1}^{T_i} \big(\mathbf{h}_{i,t}(\mathbf{y}_{i,t} - \boldsymbol{\gamma}_{i,t})^T + \lambda_1 f_{1 \rightarrow (L-1)}(\mathbf{x}_{i,t})(\mathbf{y}_{i,t} - \hat{\mathbf{y}}_{i,t})^T \big) \tag{7e}$$

where $\boldsymbol{\gamma}_{i,t} \in \mathbb{R}^K$ is the vector of K dimensions, which can be thought as the posterior probability for labels in the sequence and will be introduced in Sect. 2.3, and $\hat{\mathbf{y}}_{i,t} = f_{1\to L}(\mathbf{x}_{i,t})$ is the output from Eq. (6). Note that it is easy to derive the gradients of the ℓ_2 regularization term w.r.t. $\boldsymbol{\theta}$ in the objective in Eq. (3), which can be added to the gradients in Eq. (7).

As for the gradients of weights $\boldsymbol{\omega} = \{\mathbf{W}_l | l \in [1, .., L-1]\}$, we first use backpropagation to get the partial gradient in the neural network, refer to [15] for more details. Then the gradient of the ℓ_1 term in Eq. (3) can be attached to get the final gradients w.r.t. $\mathbf{W}_l$ for $l = \{1, .., L-1\}$.

Finally, we use a mixture of perceptron learning and stochastic gradient descent to optimize the objective function. There are various optimization methods, such as L-BFGS [3,37], stochastic gradient descent (SGD) and perceptron-based learning [28]. L-BFGS as a gradient descent method, has been widely used to optimize weights in the deep structure [16]. However, it can be slow, and there are no guarantees if there are multiple local minima in the error surface. SGD and perceptron training both are the online learning algorithms by updating the parameters using the gradient induced by a single time series, so they have significant computational advantages over L-BFGS. Furthermore, perceptron-based online learning can be viewed as a special case of SGD, but it is more flexible than SGD on parameter updating (i.e. parallelization). In our experiments, we tried L-BFGS, but it can be easily trapped into the bad local minimum, and performs worse than other optimization methods in almost all experiments. Thus, in this work, we use perceptron-based learning for the CRF related parameters and stochastic gradient descent for the weights in the deep neural structure.

If the perceptron incorrectly classifies a training example, each of the input weights is nudged a little bit in the right direction for that training example. In other words, we only need to update the CRF model only for frames that are misclassified in each training example. To update the CRF's parameters, we need to find the most violated constraints for each example. Basically, given a training example $\langle \mathbf{x}_i, \mathbf{y}_i \rangle$, we infer its most violated label $\mathbf{y}_i^*$. If the frame is misclassified, then it directly performs a type of stochastic gradient descent on the energy gap between the observed label sequence and the predicted label sequence. Otherwise, we do not need to update the model parameters. Thus, for the parameters $\boldsymbol{\theta}$ from the negative log likelihood in Eq. (3), we first project $\mathbf{x}_i$ into the code $\mathbf{h}_i$ according to Eq. (5). Then, the updating rule takes the form below

$$\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} + \eta_{\boldsymbol{\theta}} \frac{\partial}{\partial \boldsymbol{\theta}} \big(E(\mathbf{h}_i, \mathbf{y}_i) - E(\mathbf{h}_i, \mathbf{y}_i^*) \big) \tag{8}$$

where $\mathbf{y}_i^*$ is the most violated constraint in the misclassificated case, and $\eta_{\boldsymbol{\theta}}$ is the step size. Note that the posterior probability $\boldsymbol{\gamma}_{i,t} \in \mathbb{R}^K$ in Eq. (7) should be changed into the hard label assignment $\mathbf{y}_{i,t}^*$ in the inference stage. Note that this is the key difference between perceptron training (using hard label) and SGD (using label likelihood) while updating parameters.

While for the weights $\boldsymbol{\omega} = \{\mathbf{W}_l | l \in [1, .., L-1]\}$ in the deep neural network, we first use backpropagation to compute the gradients, and then update it as follows

$$\boldsymbol{\omega} \leftarrow \boldsymbol{\omega} - \eta_{\boldsymbol{\omega}} \frac{\partial \mathcal{L}}{\partial \boldsymbol{\omega}} \tag{9}$$

where $\eta_{\boldsymbol{\omega}}$ is the learning rate to update the parameters. Note that one vital step before our online learning over the parameters is that we re-randomize the top layer weight in the deep neural network. We note that this randomization step is very important for the model to generalize well in the testing data. Note that in the independent learning step, we already have trained a very good model. However, it may overfit to the data and trap into a bad local minimum. For example, the parameters in all layers have fitted the data well. Thus, if we only introduce additional CRF parameters (i.e. the transition matrix) into our whole framework, it may have no chance to update the low lever weights in the neural network (because of overfitting). On the contrary, if we re-randomize the top layer weight, we can update all parameters effectively.

2.3 Inference

Given the observation $\mathbf{x}_i = \{\mathbf{x}_{i,1}, ..., \mathbf{x}_{i,T_i}\}$, we first use Eq. (5) to compute the non-linear code $\mathbf{h}_i = \{\mathbf{h}_{i,1}, ..., \mathbf{h}_{i,T_i}\}$, and then we use Viterbi algorithm [36] to infer labels. To simplify the problem, we assume the first-order CRFs here. To estimate the parameters $\boldsymbol{\theta}$, there are two main inferential problems that need to be solved during learning: (1) the posterior probability (or the marginal distribution of a label given the codes) $\gamma_{i,t}(k) = p(y_{i,t} = k|\mathbf{h}_{i,1}, ..., \mathbf{h}_{i,T_i})$; (2) the distribution over a label edge $\xi_{i,t}(j,k) = p(y_{i,t} = j, y_{i,t+1} = k|\mathbf{h}_{i,1}, ..., \mathbf{h}_{i,T_i})$. Apparently, the inference problem can be solved efficiently with Viterbi algorithm [2,36].

For the given hidden sequence $\mathbf{h}_i = \{\mathbf{h}_{i,1}, ..., \mathbf{h}_{i,T_i}\}$, we assume the corresponding states $\{q_{i,1}, ..., q_{i,T_i}\}$. Furthermore, we define the forward messages $\alpha_{i,t}(k) \propto p(y_{i,1}, .., y_{i,t}, q_{i,t} = k|\mathbf{h}_{i,1}, ..., \mathbf{h}_{i,T_i})$, and the backward messages $\beta_{i,t}(k) \propto p(y_{i,t+1}, .., y_{i,T_i}|q_{i,t} = k, \mathbf{h}_{i,1}, ..., \mathbf{h}_{i,T_i})$

$$\alpha_{i,t+1}(j) = \left[\sum_{k=1}^{K} \alpha_{i,t}(k) A_{kj}\right] B(j, y_{i,t+1}); \tag{10}$$

$$\beta_{i,t}(j) = \sum_{k=1}^{K} A_{jk} B(k, y_{i,t+1}) \beta_{i,t+1}(k); \tag{11}$$

where $B(k, y_{i,t})$ is the probability to emit $y_{i,t}$ at the state k. We can compute it as follows

$$B(:, y_{i,t}) = exp\{\mathbf{h}_{i,t}^T \mathbf{W} + \mathbf{b}^T + \lambda_1 f_L(\mathbf{h}_{i,t})\} \tag{12}$$

After calculating $\alpha_{i,t+1}(j)$ and $\beta_{i,t}(j)$, we can compute the marginal probability for $\gamma_{i,t}$ and $\xi_{i,t}$ respectively

$$\gamma_{i,t}(k) \propto \alpha_{i,t}(k)\beta_{i,t}(k), \tag{13}$$

$$\xi_{i,t}(k,j) \propto \alpha_{i,t}(k) A_{kj} B(j, y_{i,t+1}) \beta_{i,t+1}(j); \tag{14}$$

Then, we can compute $\boldsymbol{\gamma}_{i,t}$ in Eq. (7), which is the concatenation: $[\gamma_{i,t}(1), ...,$ $\gamma_{i,t}(K)]$.

In the testing stage, the main inferential problem is to compute the most likely label sequence $\mathbf{y}^*_{1,...,T}$ given the data $\mathbf{x}_{1,...,T}$ by $\text{argmax}_{\mathbf{y'}_{1,...,T}}\, p(\mathbf{y'}_{1,...,T}|\mathbf{x}_{1,...,T})$, which can be addressed similarly using the Viterbi algorithm mentioned above.

The learning algorithm is shown in Algorithm 1. The steps 2 and 3 are the pretraining processes (independent training and re-randomization) in our model, which is different from traditional deep CRFs [9,28,33]. There two steps initialize the model parameters, which offers the advantages over traditional deep CRFs approaches and makes our model to yield significantly better results, see further in the experimental parts. Also our online mixture learning strategy is different from traditional approaches.

Algorithm 1. Deep CRFs with online learning

```
Input: sequential training data $D = \{\langle \mathbf{x}_i, \mathbf{y}_i \rangle\}_{i=1}^N$, $C$, $\lambda_1$ and $\lambda_2$, $\eta_{\boldsymbol{\omega}}$, $\eta_{\boldsymbol{\theta}}$, iterations $T$
Output: $\boldsymbol{\omega}$ and $\boldsymbol{\theta}$
1: Initialize the deep neural networks layer by layer with RBMs $\boldsymbol{\omega} = \{\mathbf{W}_l | l \in [1, .., L-1]\}$ and the top layer weight $\mathbf{W}$;
2: Fine-tune the deep neural network parameters ($\boldsymbol{\omega}$ and $\mathbf{W}$) without context information, just like to learn a deep SVM classifier;
3: Re-randomize the top layer weight $\mathbf{W}$;
4: Learn all parameters of our deep CRFs online, including $\boldsymbol{\omega}$, $\mathbf{W}$ and correlation matrix $\mathbf{A}$;
5: Return $\boldsymbol{\omega}$, $\mathbf{W}$ and $\mathbf{A}$;
```

3 Experiments

To test our method, we compared our method to the state of the art approaches and performed experiments on four sequential labeling tasks: (1) optical character recognition, (2) labeling questions and answers, (3) protein secondary structure prediction, and (4) part-of-speech tagging. Below, we described the datasets we used and also the parameter setting in the experiments.

3.1 Data Sets

1. The OCR dataset [42] contains data with 6, 877 handwritten words, in which there are total 55 unique words, and each word $\mathbf{x}_i$ is represented as a series of handwritten characters $\{\mathbf{x}_{i1}, ..., \mathbf{x}_{i,T_i}\}$. The data consists of a total of 52, 152 characters (i.e., frames), with 26 unique classes. Each character is a binary image of size 16×8 pixels, leading to a 128-dimensional binary feature vector.
2. The FAQ data set [30] contains data of 48 files on questions and answers, with a total of 55,480 sentences (i.e., frames). Each sentence is a 24- dimensional

binary feature that describes lexical characteristics of the sentence. Each sentence in the FAQ data set belongs to one of four labels: (1) question, (2) answer, (3) header, or (4) footer.
3. The CB513 contains amino acid structures of 513 proteins [6], and has been widely used for protein secondary structure prediction. For each of the proteins, it has 20-dimensional position-specific score matrix features. In the experiment, we concatenate the features from the surrounding 13 frames into the 260 dimensional vector [28]. As common in protein secondary structure prediction, we convert the eight-class labeling into a three-class labeling. The resulting data set has 513 sequences with total 74, 874 frames (260 dimensions), belonging to 3 classes.
4. The Penn Treebank corpus[1] has 74, 029 sentences with a total of 1, 637, 267 words. The whole data set contains 49, 115 unique words, and each word in each sentence is labeled according to its part of speech with total 43 different tags. To represent each word, all features are measured in a window with width 3 around the current word, which leads to a total of 212, 610 features. If we use 1000 hidden nodes, then we need to store 2×10^8 parameters in the one-layer neural network. Considering the high storage demanding for the personal computer, we calculated the frequency for each dimension in the total 212, 610 features, and selected the most frequent 5000 features as our codebook. Then we can represent each word with 5000 dimensions in our experiment. All the four data sets in [28] are available on the author's website[2].

3.2 Experimental Setup

In our experiments, we randomly initialized the weight $\mathbf{W}$ by sampling from the normal Gaussian distribution, and all other parameters in $\boldsymbol{\theta}$ to be zero (i.e. biases $\boldsymbol{b}$ and $\boldsymbol{c}$, and the transition matrix $\mathbf{A}$ all to be zero). As for $\boldsymbol{\omega} = \{\mathbf{W}_l | l \in [1, .., L-1]\}$, we initialized them with DBN, which had been mentioned before. As for the number of layers and the number of hidden units in each layer, we set differently according to the dimensionality for different datasets. In all the experiments, we use the 3-layer deep neural networks on the four datasets. Considering the OCR dataset has 128 dimensional binary feature, while FAQ is the dataset with 24 dimensional vector, we set the number of hidden nodes [100 100 64] in each layer respectively on both the OCR dataset and FAQ dataset. For the CB513 dataset, we set the number of hidden units to be [400 200 100]. For the treebank dataset, the hidden units [1000 400 200] are used in the 3-layer network. We did not try other deep structure in the experiments.

After weight initialization with independent learning, we set $\lambda_1 = 0$ and then used perceptron training [12] to estimate the CRF related parameters and SGD to learn the weights in the deep neural network. We did not use regularization by setting $\lambda_2 = 0$ in perceptron learning, and set $\lambda_3 = 2 \times 10^{-4}$ for SGD weights in the deep network. From the base step size, we computed parameter-specific step sizes $\eta_{\boldsymbol{\theta}}$ and $\eta_{\boldsymbol{\omega}}$ as suggested by [12]. For each dataset, we divided

[1] www.cis.upenn.edu/~treebank.
[2] https://github.com/ganggit/deepCRFs.

it into 10 folds (9 folds as the training set, and the rest as the testing/validation set), and performed 100 full sweeps through the training data, to update the model parameters. We tuned the base step size based on the error on a small held-out validation set. Unless otherwise indicated, we use the average generalization error to measure all methods in 10-fold cross-validation experiments.

3.3 Results

We test our method on the four data sets mentioned above with the second-order CRFs with deep learning, and compare our method with linear CRFs, deep neural networks, traditional deep CRFs and LSTM. We also test whether the pretraining step: independent learning and re-randomization of the top layer weight is helpful or not in the sequential labeling tasks.

Table 1. The experimental comparisons on the OCR dataset. Our method (without pretraining) is from the result without steps 2 and 3 in Algorithm 1. The results reveal the merits of our method over other methods.

Hand-written recognition (Error rate %)	
Linear-chain CRF [9]	14.2
Max-margin Markov net [9]	13.4
Searn [8]	9.09
SVM + CRF [17]	5.76
Deep learning (DBN+Labeling) [16]	4.0
NeuroCRFs (Deep learning + CRFs)[9]	4.44
Cond. graphical models [34]	2.7
LSTM [19]	0.40
Bidirectional LSTM [19]	0.36
Hidden-unit CRF [28]	1.9
Our method (without pretraining)	1.56
Our method	**0.2**

In Table 1, we compared the performance of our method with the performance of competing models on the handwriting recognition task. It shows that the pretraining stage (independent learning and re-randomization) in our model is helpful to improve the recognition accuracy (boosting error rate from 1.56 % to 0.2 %). It also shows that the label correlation is helpful in this case. For example, the deep learning without label correlation yields accuracy 4.0 %, while is significantly lower than our model. Compared to previous deep CRFs and RNNs, our method with all steps in Algorithm 1 yields a generalization error of 0.2 %, while the best performance of other methods is 0.36 %. Note that we change the LSTM code a little bit in [19] to handle the handwritten images.

Table 2. The comparison (generalization errors) on the FAQ dataset using different methods. It shows that our method is significantly better than the Hidden-unit CRF.

Model	Error rate (%)
Linear SVM	9.87
Linear CRF [28]	6.54
NeuroCRFs (Deep learning + CRFs)[9]	6.05
Hidden-unit CRF [28]	4.43
Deep learning (DBN + Labeling) [16]	7.75
Our method (without pretraining)	7.44
Our method	**3.34**

It demonstrates that our learning approach is significantly better than other methods, and the deep structure is definitely helpful than the shallow models.

On the FAQ data set, the lowest generalization error of hidden-unit CRFs is 4.43 %, compared to 3.34 % for our method in Table 2. And again, our method outperforms other competitive baselines. It also shows that the CRF with deep feature learning (3 layers) in this case, is better than the one hidden layer CRF. Note that we just used the original 24 dimension features in the experiment, instead of extending the feature set into a 24 + 242 = 600-dimensional feature representation in [28].

We also test our method on the protein secondary structure prediction task. The results of these experiments are presented in Table 3. In particular, our method achieves a generalization error of only 3.16 %, compared to 19.5 % error with the conditional neural field on the CB513 data set. The results presented

Table 3. The comparison on the CB513 dataset for protein secondary structure prediction task. It demonstrates that our method significantly outperforms other approaches. And the pretraining step with the independent label learning is very helpful to boost the performance.

Model	Error rate (%)
PSIRED [18]	24.0
SVM [14]	23.4
SPINE [10]	23.2
YASSP [20]	22.2
Cond. neural field (Deep learning + CRFs)[33]	19.5
NeuroCRFs (Deep learning + CRFs)[9]	28.4
Hidden-unit CRF [28]	20.2
Deep learning (DBN+Labeling) [16]	8.57
Our method (without pretraining)	27.1
Our method	**3.16**

Table 4. The experimental comparison on the treebank data set by varying the number of training data. It demonstrates that given few training data, our method is generalized well and more robust in the recognition task.

Model	generalization error rate (%)						
	1000	2000	4000	5000	8000	10000	20000
Linear SVM	12.22	12.0	10.6	9.33	8.96	8.71	8.27
Deep learning [15]	58.73	11.4	9.44	9.28	8.36	8.17	7.60
Hidden-unit CRF [28]	10.2	8.74	7.59	7.45	7.02	6.79	6.29
Our method	**9.79**	**7.97**	**6.66**	**6.5**	**6.26**	**6.24**	**6.01**

in the figure indicate that the CRFs with deep feature learning can significantly improve the performance, compared to hidden-unit CRFs.

Lastly, we also tested our method on part-of-speech tagging task. Note that we already take context information into consideration by using a window width 3 for feature representations. And the final representation is based on only 5,000 codebooks because of storage problem for model parameters. To test whether our method can tackle overfitting problem effectively, we randomly sampled a subset from the Penn Treebank corpus, and did the 10 fold cross validation. We show the experimental results in Table 4. It demonstrates that when there's a few data set available for training, deep learning with L-BFGS has overfitting problems. As the number of training data increasing, the performance of the deep learning also is increasing. While our method outperforms other baselines remarkably, and show stable and better performance with increasing training data. It also shows that our method can generalize well effectively, and it is more robust with few training data in the recognition task.

4 Related Work

Deep learning has significantly improvement on classification, such as object recognition, handwriting and natural language processing [5,19,22]. However, to predict structured output with deep learning is still a challenge in machine learning [31]. The difficulty of this problem is that the input and output data have non-determinated length, which may lead to an exponential number of possible configurations. Recently, a conditional RMBs is proposed for structured output prediction [31]. Unfortunately, the model is shallow with only one hidden layer, and also cannot deal with large output configurations well. Typically, it either considers a small output space or uses semantic hashing in order to efficiently compute a small set of possible outputs to predict. LSTM has attracted great attention for sequential labeling. For example, a 2-branch LSTM model has been proposed for machine translation [40]. Also LSTM is combined with CNN for image to text task [19].

On the other hand, graphic models, such as hidden Markov model (HMM) and CRFs have been an popular method for segmentation and labeling time series data [24]. And CRFs, as a discriminative probabilistic model for structured prediction [24], has been widely used in natural language processing [38], handwriting recognition [28,42] and scene parsing [39]. Over the last decade, many different approaches have been proposed to improve its performance on the sequential labeling problems. One trend is to extend the linear CRFs into the high-order graphical model, by exploiting more context information [7,21]. However, the main weakness of those approaches is the time-consuming inference in the high-order graphical model. Another trend in the CRFs is to discover discriminative features to improve classification performance. One related work is a multilayer CRF (ML-CRF) [35]. The 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. Similarly, hidden-unit CRFs [28,33] also assumes one-hidden layer for feature representation. The main idea of these two methods is similar to our approach here, in that we also transform the input to construct hidden features from the data so that these hidden units are discriminative in classification. But, unlike those systems, our model inherits the advantages of deep learning, and feature functions do not have any direct interpretation and are learned implicitly. Moreover, the deep features learned with large hidden units are powerful enough to represent the data, and generalize well in the classification tasks. As demonstrated by previous work, the performance of linear CRFs on a given task is strongly dependent on the feature representations [41]; while deep learning [16] can learn representations that are helpful for classification. Thus, it is possible to unify these two methods into one framework.

Our sequential labeling model with deep learning also bears some resemblance to approaches that train a deep network, and then train a linear CRF or Viterbi decoder on the output of the resulting network [9,29,32]. However, these methods differ from our approach in that (1) The initialization step in our approach with independent learning and weight re-randomization can significantly boost performance; (2) they do not learn all state-transition, data-dependent parameters and weights in the deep networks jointly. As a result, the top hidden units in these models may not discover latent distributed representations that are discriminative for classification. (3) Previous approaches [27,35] does not take an online learning strategy to estimate model parameters. But we consider to update the weights with an online algorithm in our deep learning model, which can learn more useful representations [1] to handle large scale dataset. Our work here inherits both advantages of CRFs and deep learning. Thus, our model can effectively handle structured prediction, and also learn discriminative features automatically for better sequential labeling under an unified framework.

5 Conclusions

In this paper, we introduce a model for sequential labeling with online deep CRFs. More specifically, we propose a mixture objective function, which learns the non-linear features with deep learning and predict labels with CRFs in the sequential data. Hence, our approach leverage both feature learning and context information for the classification and segmentation of time series. One vital issue arising while training the deep model is how to handle overfitting and local bad minimum. We take an effective initialization step (with dependent label learning and re-randomization) to address this problem. Finally, we use a simple but efficiently online learning method to update the whole model (end-to-end), which has the potential to handle large-scale learning problem. In the experiments, we show that our model outperforms the current state of the art remarkably on a wide range of tasks.

References

1. Bengio, Y., Courville, A., Vincent, P.: Representation learning: a review and new perspectives. PAMI **35**, 1798–1828 (2013)
2. Bishop, C.M.: Pattern Recognition and Machine Learning. Springer, Secaucus (2006)
3. Byrd, R.H., Lu, P., Nocedal, J., Zhu, C.: A limited memory algorithm for bound constrained optimization. SIAM J. Sci. Comput. **16**(5), 1190–1208 (1995)
4. Chen, G.: Deep transductive semi-supervised maximum margin clustering. CoRR abs/1501.06237 (2015)
5. Chen, G., Srihari, S.N.: Removing structural noise in handwriting images using deep learning. In: ICVGIP 2014, pp. 28:1–28:8. ACM, New York (2014)
6. Cuff, J.A., Barton, G.J.: Evaluation and improvement of multiple sequence methods for protein secondary structure prediction. PSFG **34**, 508–519 (1999)
7. Cuong, N.V., Ye, N., Lee, W.S., Chieu, H.L.: Conditional random field with high-order dependencies for sequence labeling and segmentation. JMLR **15**(1), 981–1009 (2014)
8. Daumé III, H., Langford, J., Marcu, D.: Search-based structured prediction. Mach. Learn. **75**(3), 297–325 (2009)
9. Do, T.M.T., Artires, T.: Neural conditional random fields. In: AISTATS, vol. 9, pp. 177–184 (2010)
10. Dor, O., Zhou, Y.: Achieving 80 % ten-fold cross-validated accuracy for secondary structure prediction by large-scale training (2007)
11. Erhan, D., Bengio, Y., Courville, A., Manzagol, P.A., Vincent, P., Bengio, S.: Why does unsupervised pre-training help deep learning? J. Mach. Learn. Res. **11**, 625–660 (2010)
12. Gelfand, A., Chen, Y., van der Maaten, L., Welling, M.: On herding and the perceptron cycling theorem. In: NIPS, pp. 694–702 (2010)
13. Graves, A., Schmidhuber, J.: Offline handwriting recognition with multidimensional recurrent neural networks. In: NIPS, pp. 545–552. Curran Associates, Inc. (2008)
14. Kim, H., Park H.: Protein secondary structure prediction based on an improved support vector machines approach (2003)
15. Hinton, G.E., Salakhutdinov, R.R.: Reducing the dimensionality of data with neural networks. Science **313**(5786), 504–507 (2006)
16. Hinton, G.E., Osindero, S., Teh, Y.W.: A fast learning algorithm for deep belief nets. Neural Comput. **18**(7), 1527–1554 (2006)
17. Hoefel, G., Elkan, C.: Learning a two-stage svm/crf sequence classifier. In: CIKM, pp. 271–278. ACM (2008)

18. Jones, D.T.: Protein secondary structure prediction based on position-specific scoring matrices. J. Mol. Biol. **292**, 195–202 (1999)
19. Karpathy, A., Li, F.F.: Deep visual-semantic alignments for generating image descriptions. In: CVPR, pp. 3128–3137. IEEE (2015)
20. Karypis, G.: Yasspp: Better kernels and coding schemes lead to improvements in protein secondary structure prediction (2006)
21. Krähenbühl, P., Koltun, V.: Efficient inference in fully connected CRFs with gaussian edge potentials. In: NIPS (2011)
22. Krizhevsky, A., Sutskever, I., Hinton, G.E.: Imagenet classification with deep convolutional neural networks. In: NIPS (2012)
23. Lafferty, J., Zhu, X., Liu, Y.: Kernel conditional random fields: representation and clique selection. In: ICML, p. 64. ACM (2004)
24. Lafferty, J.D., McCallum, A., Pereira, F.C.N.: Conditional random fields: probabilistic models for segmenting and labeling sequence data. In: ICML, pp. 282–289 (2001)
25. Larochelle, H., Bengio, Y.: Classification using discriminative restricted boltzmann machines. In: ICML, pp. 536–543. ACM, New York (2008)
26. Larochelle, H., Mandel, M., Pascanu, R., Bengio, Y.: Learning algorithms for the classification restricted Boltzmann machine. J. Mach. Learn. Res. **13**(1), 643–669 (2012)
27. Lecun, Y., Bottou, L., Bengio, Y., Haffner, P.: Gradient-based learning applied to document recognition. Proc. IEEE **86**(11), 2278–2324 (1998)
28. van der Maaten, L., Welling, M., Saul, L.K.: Hidden-unit conditional random fields. In: AISTATS, pp. 479–488 (2011)
29. Chen, G., Li, Y., Srihari, S.N.: Word recognition with deep conditional random fields. In: ICIP (2016)
30. McCallum, A., Freitag, D., Pereira, F.C.N.: Maximum entropy markov models for information extraction and segmentation. In: ICML, pp. 591–598 (2000)
31. Mnih, V., Larochelle, H., Hinton, G.E.: Conditional restricted boltzmann machines for structured output prediction. In: UAI, pp. 514–522 (2011)
32. Mohamed, A.-r., Dahl, G.E., Hinton, G.E.: Deep belief networks for phone recognition. In: NIPS Workshop on Deep Learning for Speech Recognition and Related Applications (2009)
33. Peng, J., Bo, L., Xu, J.: Conditional neural fields. In: NIPS, pp. 1419–1427 (2009)
34. Pérez-Cruz, F., Ghahramani, Z., Pontil, M.: Conditional Graphical Models (2007)
35. Prabhavalkar, R., Fosler-Lussier, E.: Backpropagation training for multilayer conditional random field based phone recognition. In: ICASSP 2010, pp. 5534–5537 (2010)
36. Rabiner, L.R.: A tutorial on hidden markov models and selected applications in speech recognition. Proc. IEEE **77**(2), 257–287 (1989)
37. Rasmussen, C.E., Williams, C.K.I.: Gaussian Processes for Machine Learning. The MIT Press, Cambridge (2005)
38. Sha, F., Pereira, F.: Shallow parsing with conditional random fields. In: NAACL, pp. 134–141 (2003)
39. Shotton, J., Winn, J., Rother, C., Criminisi, A.: *TextonBoost*: joint appearance, shape and context modeling for multi-class object recognition and segmentation. In: Leonardis, A., Bischof, H., Pinz, A. (eds.) ECCV 2006. LNCS, vol. 3951, pp. 1–15. Springer, Heidelberg (2006). doi:10.1007/11744023_1
40. Sutskever, I., Vinyals, O., Le, Q.V.: Sequence to sequence learning with neural networks. In: NIPS (2014)

41. Sutton, C., Mccallum, A.: Introduction to Conditional Random Fields for Relational Learning. MIT Press, Cambridge (2006)
42. Taskar, B., Guestrin, C., Koller, D.: Max-margin markov networks. In: NIPS. MIT Press (2003)
43. Vincent, P., Larochelle, H., Lajoie, I., Bengio, Y., Manzagol, P.A.: Stacked denoising autoencoders: Learning useful representations in a deep network with a local denoising criterion. J. Mach. Learn. Res. **11**, 3371–3408 (2010)
44. Weston, J., Ratle, F.: Deep learning via semi-supervised embedding. In: International Conference on Machine Learning (2008)