Neural conditional random fields

Trinh-Minh-Tri Do[†‡]
[†]Idiap Research Institute
Martigny, Switzerland
Tri.Do@idiap.ch

Thierry Artières[‡]
[‡]LIP6, Université Pierre et Marie Curie
Paris, France
Thierry.Artieres@lip6.fr

Abstract

We propose a non-linear graphical model for structured prediction. It combines the power of deep neural networks to extract high level features with the graphical framework of Markov networks, yielding a powerful and scalable probabilistic model that we apply to signal labeling tasks.

1 INTRODUCTION

This paper considers the structured prediction task where one wants to build a system that predicts a structured output from an (structured) input. It is a common framework for many application fields such as bioinformatics, part-of-speech tagging, information extraction, signal (e.g. speech) labeling and recognition and so on. We focus here on signal and sequence labeling tasks for signals such as speech and handwriting.

For decades, Hidden Markov Models (HMMs) have been the most popular approach for dealing with sequential data (e.g. for segmentation and classification). They rely on strong independence assumptions and are learned using Maximum Likelihood Estimation which is a non discriminant criterion. This latter point comes from the fact that HMMs are generative models and they define a joint probability distribution on the sequence of observations $\mathbf{X}$ and the associated label sequence $\mathbf{Y}$.

Discriminant systems are usually more powerful than generative models, and focus more directly on minimizing the error rate. Many studies have focused on developing discriminant training for HMM, for example Minimum Classification Error (MCE) (Juang & Katagiri, 1992), Perceptron learning (Collins, 2002), Maximum Mutual Information (MMI) (Woodland & Povey, 2002) or more recently large margin approaches (Sha & Saul, 2007; Do & Artières, 2009).

A more direct approach is to design a discriminative graphical model that models the conditional distribution $P(\mathbf{Y}|\mathbf{X})$ instead of modeling the joint probability as in generative model (Mccallum et al., 2000; Lafferty, 2001). Conditional random fields (CRF) are a typical example of this approach. Maximum Margin Markov network (M3N) (Taskar et al., 2004) go further by focusing on the discriminant function (which is defined as the log of potential functions in a Markov network) and extend the SVM learning algorithm for structured prediction. While using a completely different learning algorithm, M3N is based on the same graphical modeling as CRF and can be viewed as an instance of a CRF. Based on log-linear potentials, CRFs have been widely used for sequential data such as natural language processing or biological sequences (Altun et al., 2003; Sato & Sakakibara, 2005). However, CRFs with log-linear potentials only reach modest performance with respect to non-linear models exploiting kernels (Taskar et al., 2004). Although it is possible to use kernels in CRFs (Lafferty et al., 2004), the obtained dense optimal solution makes it generally inefficient in practice. Nevertheless, kernel machines are well known to be less scalable.

Besides, in recent years, deep neural architectures have been proposed as a relevant solution for extracting high level features from data (Hinton et al., 2006; Bengio et al., 2006). Such models have been successfully applied first to images (Hinton et al., 2006), then to motion caption data (Taylor et al., 2007) and text data. In these fields, deep architectures have shown great capacity to discover and extract relevant features as input to linear discriminant systems.

This work introduces neural conditional random fields which are a marriage between conditional random fields and (deep) neural networks (NNs). The idea is to rely on deep NNs for learning relevant high level

Appearing in Proceedings of the 13^{th} International Conference on Artificial Intelligence and Statistics (AISTATS) 2010, Chia Laguna Resort, Sardinia, Italy. Volume 9 of JMLR: W&CP 9.

features which may then be used as inputs to a linear CRF. Going further we propose such a global architecture that we call NeuroCRF and that can be globally trained with a discriminant criterion. Of course, using a deep NN as a feature extractor makes the learning become a non convex optimization problem. This prevents relying on efficient convex optimizer algorithms. However recently a number of researchers have pointed out that convexity at any price is not always a good idea. One has to look for an optimal trade-off between modeling flexibility and optimization ease (LeCun et al., 1998; Collobert et al., 2006; Bengio & LeCun, 2007).

Related works. Some previous works have successfully designed NN systems for structured prediction. For instance, graph transformer nets (Bottou et al., 1997) have been applied to a complex check reading system that uses convolution net at character level. (Graves et al., 2006) used a recurrent NN for handwriting and speech recognition where neural net outputs (sigmoid units) are used as conditional probabilities. Motivated by the success of deep belief nets on feature discovering, Collobert and his colleagues investigated the use of deep learning for information extraction on text data (Qi et al., 2009). A common point between these works is that the authors proposed mechanics to adapt NN for the structured prediction task rather than a global probabilistic framework, which is investigated in this paper. Recently, (Peng et al., 2009) also investigated the combination of CRFs and NNs in a parallel work. Our approach is different in the use of deep architecture and backpropagation, and it works for general loss function.

2 NEURAL CONDITIONAL RANDOM FIELDS

In this section, we propose a non-linear graphical model for structured prediction. We start with a general framework with any graphical structure. Then we focus on linear chain models for sequence labeling.

2.1 Conditional random fields

Structured output prediction aims at building a model that predicts accurately a structured output $\mathbf{y}$ for any input $\mathbf{x}$. The output $\mathbf{Y} = \{Y_i\}$ is a set of predicted random variables whose components belong to a set of labels $\mathcal{L}$ and are linked by conditional dependencies encoded by an undirected graph $G = (V, E)$ with cliques $c \in C$. Given $\mathbf{x}$, inference stands for finding the output that maximizes the conditional probability[1] $p(\mathbf{y}|\mathbf{x})$. Relying on the Hammersley-Clifford theorem a CRF defines a conditional probability according to:

$$\begin{array}{ll} p(\mathbf{y}|\mathbf{x}) & = 1/Z(\mathbf{x}) \prod_{c\in C} \psi_c(\mathbf{x}, \mathbf{y}_c) \\ with: & Z(\mathbf{x}) = \sum_{\mathbf{y}\in\mathcal{Y}} \prod_{c\in C} \psi_c(\mathbf{x}, \mathbf{y}_c) \end{array} \quad (1)$$

where $Z(\mathbf{x})$ is a global normalization factor. A common choice for potential functions is the exponential function of an energy, $\mathrm{E_c}$:

$$\psi_c(\mathbf{x}, \mathbf{y}_c) = e^{-\mathrm{E_c}(\mathbf{x}, \mathbf{y}_c, \mathbf{w})} \quad (2)$$

To ease learning, a standard setting is to use linear energy functions $\mathrm{E_c}(\mathbf{x}, \mathbf{y_c}, \mathbf{w}) = -\langle \mathbf{w}_c^{\mathbf{y}_c}, \Phi_c(\mathbf{x}) \rangle$ of the parameter vector $\mathbf{w}_c^{\mathbf{y}_c}$ and of a feature vector $\Phi_c(\mathbf{x})$. This leads to a log-linear model (Lafferty, 2001). A linear energy function is intrinsically limiting the CRF. We propose neural CRFs to replace this linear energy function by non linear energy functions that are computed by a NN.

2.2 Neural conditional random fields

Neural conditional random fields is a combination of NNs and CRFs. They extend CRFs by placing a NN structure between input and energy function. This NN, visualized in Figure 1, is described in detail next.

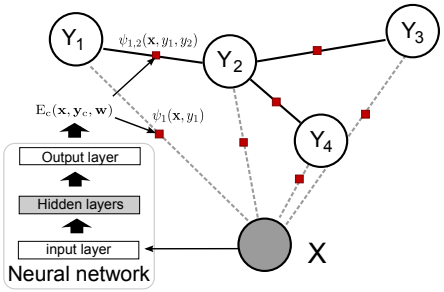

Figure 1: Example of a tree-structured NeuroCRF

The NN takes an observation as input and outputs a number of quantities which we call *energy outputs*[2] $\{\mathrm{E_c}(\mathbf{x}, \mathbf{y_c}, \mathbf{w})|\mathrm{c}, \mathbf{y_c}\}$ parameterized by $\mathbf{w}$. The NN is feed forward with multiple hidden layers, non-linear hidden units, and an output layer with linear output units (i.e. a linear activation function). With this setting, a NeuroCRF may be viewed as a standard log-linear CRF working on the high-level representation computed by a neural net. In the remainder of the paper we call the top part (output layer weights) of a NeuroCRF its CRF-part and we call the remaining part its deep-part (see Figure 2-right). Let $\mathbf{w}_{\text{NN}}$ and $\mathbf{w}_c^{\mathbf{y}_c}$ be the neural net weights of the deep-part and

[1] We use the notation $p(\mathbf{y}|\mathbf{x}) = p(\mathbf{Y} = \mathbf{y}|\mathbf{X} = \mathbf{x})$.

[2] We use the terminology *energy output* to stress the difference between NN outputs and model outputs $\mathbf{y}$.

the CRF-part respectively. NeuroCRF implements the conditional probability as:

$$p(\mathbf{y}|\mathbf{x}) \propto \prod_{c \in C} e^{-\mathrm{E_c}(\mathbf{x},\mathbf{y_c},\mathbf{w})} = \prod_{c \in C} e^{\langle \mathbf{w}_c^{\mathbf{y}_c}, \Phi_c(\mathbf{x},\mathbf{w}_{\mathrm{NN}}) \rangle} \quad (3)$$

where $\Phi_c(\mathbf{x}, \mathbf{w}_{\mathrm{NN}})$ stands for the high level representation of the input $\mathbf{x}$ at clique c computed by the deep part. This is illustrated in Figure 2-left, where the last hidden layer includes units that are grouped in a number of sets, e.g. one for every clique $\Phi_c(\mathbf{x}, \mathbf{w}_{\mathrm{NN}})$. Each output unit $-\mathrm{E_c}(\mathbf{x}, \mathrm{y_c}, \mathbf{w_c})$ is connected to $\Phi_c(\mathbf{x}, \mathbf{w}_{\mathrm{NN}})$ in the last hidden layer, with the weight vector $\mathbf{w}_c^{\mathbf{y}_c}$. Note that the number of energy outputs for each clique c equals $|\mathbf{Y}_c|$, hence there are $|\mathbf{Y}_c|$ weight vectors $\mathbf{w}_c^{\mathbf{y}_c}$ for each clique c.

Inference in NeuroCRFs consists of finding $\widehat{\mathbf{y}}$ that best matches input $\mathbf{x}$ (i.e. with lowest energy):

$$\begin{aligned} \widehat{\mathbf{y}} &= \mathrm{argmax}_{\mathbf{y}}\, p(\mathbf{y}|\mathbf{x}, \mathbf{w}) \\ &= \mathrm{argmin}_{\mathbf{y}} \textstyle\sum_{c \in C} \mathrm{E_c}(\mathbf{x}, \mathrm{y_c}, \mathbf{w}) \end{aligned} \quad (4)$$

This can be done in two steps. First one feeds the NN with input $\mathbf{x}$ and forward information to compute all energy outputs $\mathrm{E_c}(\mathbf{x}, \mathrm{y_c}, \mathbf{w})$. In a second step one uses dynamic programming to find the output $\widehat{\mathbf{y}}$ with the lowest energy.

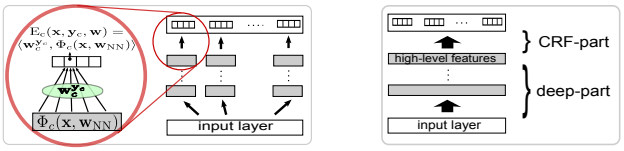

Figure 2: NN architecture with non shared weights (left) or shared weights (right).

Shared weights network architecture. Various architectures may be used for the NN. One can use a different NN for every energy function, which may result in overfitting and high computational cost. Instead, as we presented NeuroCRFs above, one may share weights to compute a high level representation per clique (and the corresponding energy outputs) (Figure 2-left). Or one may choose to compute a shared high level representation of the input, from which all energy outputs are computed (Figure 2-right). In this latter case a NeuroCRF implements the conditional probability as:

$$p(\mathbf{y}|\mathbf{x}) \propto \prod_{c \in C} e^{\langle \mathbf{w}_c^{\mathbf{y}_c}, \Phi(\mathbf{x},\mathbf{w}_{\mathrm{NN}}) \rangle} \quad (5)$$

2.3 LINEAR CHAIN NEUROCRFS FOR SEQUENCE LABELING

While the NeuroCRF framework we propose is quite general, in our experiments we focused on linear chain NeuroCRFs based on a first-order Markov chain structure (Figure 3). This allows investigating the potential power of NeuroCRFs on standard sequence labeling tasks. In a chain-structured NeuroCRF there are two kinds of cliques:

- *local cliques* $(\mathbf{x}, y_t)$ at each position t, whose potential functions are noted by $\psi_t(\mathbf{x}, y_t)$, and corresponding energy functions are noted by $\mathrm{E_{loc}}$
- *transition cliques* $(\mathbf{x}, y_{t-1}, y_t)$ between two successive positions at $t-1$ and t, whose potential functions are noted as $\psi_{t-1,t}(\mathbf{x}, y_{t-1}, y_t)$, and corresponding energy functions are noted by $\mathrm{E_{tra}}$

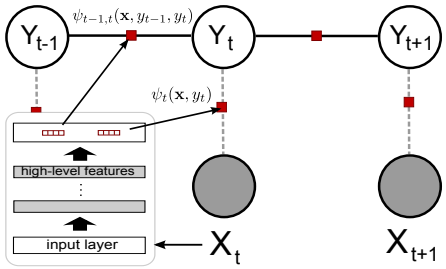

Figure 3: A chain-structured NeuroCRF

In such models it is usual to consider that energy functions are shared between similar cliques at different times (i.e. positions in the graph) (Lafferty, 2001)[3]. Then energy functions take an additional argument to specify the position in the graph, which is time t.

$$\begin{aligned} \psi_t(\mathbf{x}, y_t) &= e^{-\mathrm{E_{loc}}(\mathbf{x},\mathrm{t},\mathrm{y_t},\mathbf{w})} \\ \psi_{t-1,t}(\mathbf{x}, y_{t-1}, y_t) &= e^{-\mathrm{E_{tra}}(\mathbf{x},\mathrm{t},\mathrm{y_{t-1}},\mathrm{y_t},\mathbf{w})} \end{aligned} \quad (6)$$

The additional parameter t allows the consideration of a part of input $\mathbf{x}$, whose size may vary and cannot be handled by a fixed size input NN. Time is used to build the input to the NN in order to compute $\mathrm{E_{loc}}(\mathbf{x}, \mathrm{t}, \mathrm{y_t})$. It may consist of $\mathbf{x}_t$, the t^{th} element of $\mathbf{x}$ only (see Figure 3), or it may include a richer temporal context such as (x_{t-1}, x_t, x_{t+1}). At the end, the conditional probability of output $\mathbf{y}$ given input $\mathbf{x}$ is defined as:

$$p(\mathbf{y}|\mathbf{x}, \mathbf{w}) = \frac{e^{-\sum_{t \geq 1} \mathrm{E_{loc}}(\mathbf{x},\mathrm{t},\mathrm{y_t},\mathbf{w}) - \sum_{t>1} \mathrm{E_{tra}}(\mathbf{x},\mathrm{t},\mathrm{y_{t-1}},\mathrm{y_t},\mathbf{w})}}{Z(\mathbf{x})} \quad (7)$$

with $Z(\mathbf{x})$ being the normalization factor. With this modeling, one can derive a compact architecture of a NN with $|\mathcal{L}| + |\mathcal{L}|^2$ outputs to compute all energy outputs.

[3]These authors consider two set of parameters, one for local cliques and one for transition cliques.

3 PARAMETER ESTIMATION

Let $(\mathbf{x}^1, \mathbf{y}^1), ..., (\mathbf{x}^n, \mathbf{y}^n) \in \mathcal{X} \times \mathcal{Y}$ be a training set of n input-output pairs. We seek parameters $\mathbf{w}$ such that:

$$\mathbf{y}^i = \underset{\mathbf{y} \in \mathcal{Y}}{\operatorname{argmax}}\, p(\mathbf{y}|\mathbf{x}^i, \mathbf{w}) \tag{8}$$

This translates into a general optimization problem:

$$\min_{\mathbf{w}} \lambda \Omega(\mathbf{w}) + R(\mathbf{w}) \tag{9}$$

where $R(\mathbf{w}) = \frac{1}{n} \sum_i R_i(\mathbf{w})$ is a data-fitting measurement (e.g. empirical risk), and $\Omega(\mathbf{w})$ is a regularization term, with λ being a regularization factor that is used to find a tradeoff between a good fit on training data and a good generalization. A common choice of $\Omega(\mathbf{w})$ is to use L_2 regularization.

Now, we discuss different criterion for training NeuroCRFs. Then, we explain how we optimize these criterion to learn NeuroCRFs. Finally, we discuss about regularization and evoke semi-supervised learning.

3.1 CRITERIA

There are many discriminative criteria for training CRFs (more generally log-linear models), which can all be used for learning NeuroCRFs as well.

Probabilistic criterion. In (Lafferty, 2001), estimation of CRF parameters $\mathbf{w}$ was done by maximizing the conditional likelihood (CML) which results in:

$$\begin{aligned} R_i^{CML}(\mathbf{w}) &= -\log p(\mathbf{y}^i|\mathbf{x}^i, \mathbf{w}) \\ &= \textstyle\sum_c \mathrm{E_c}(\mathbf{x}^\mathrm{i}, \mathrm{y}_\mathrm{c}^\mathrm{i}, \mathbf{w}) \\ &+ \textstyle\sum_{y \in \mathcal{Y}} \exp[-\sum_c \mathrm{E_c}(\mathbf{x}^\mathrm{i}, \mathrm{y_c}, \mathbf{w})] \end{aligned} \tag{10}$$

Large margin criterion. The large margin method focuses more directly on giving highest discriminant score to the correct output. In NeuroCRFs, the discriminant function is a sum of energy functions over cliques (see Eq. (4)):

$$F(\mathbf{x}, \mathbf{y}, \mathbf{w}) = -\sum_{c \in C} \mathrm{E_c}(\mathbf{x}, \mathrm{y_c}, \mathbf{w}) \tag{11}$$

Large margin training for structured output (Taskar et al., 2004) aims at finding $\mathbf{w}$ so that:

$$F(\mathbf{x}^i, \mathbf{y}^i, \mathbf{w}) \geq F(\mathbf{x}^i, \mathbf{y}, \mathbf{w}) + \Delta(\mathbf{y}^i, \mathbf{y}) \; \forall \mathbf{y} \in \mathcal{Y} \tag{12}$$

where $\Delta(\mathbf{y}^i, \mathbf{y})$ allows taking into account differences between labelings (e.g. Hamming distance between $\mathbf{y}$ and $\mathbf{y}^i$). We assume a decomposable loss (alike Hamming distance) such that $\Delta(\mathbf{y}^i, \mathbf{y}) = \sum_c \delta(y_c^i, y_c)$ so that it can be factorized along the graph structure and integrated in the dynamic programming pass needed to compute $\operatorname{argmax}_{\mathbf{y} \in \mathcal{Y}} p(\mathbf{y}|\mathbf{x})$.

The elementary loss function of NeuroCRFs is then:

$$\begin{aligned} & R_i^{LM}(\mathbf{w}) \\ = \; & \max_{\mathbf{y} \in \mathcal{Y}} F(\mathbf{x}^i, \mathbf{y}, \mathbf{w}) - F(\mathbf{x}^i, \mathbf{y}^i, \mathbf{w}) + \Delta(\mathbf{y}^i, \mathbf{y}) \\ = \; & \max_{\mathbf{y} \in \mathcal{Y}} \textstyle\sum_c \Delta \mathrm{E_c}(\mathbf{x}^\mathrm{i}, \mathrm{y}_\mathrm{c}^\mathrm{i}, \mathrm{y_c}, \mathbf{w}) + \delta(\mathrm{y}_\mathrm{c}^\mathrm{i}, \mathrm{y_c}) \end{aligned} \tag{13}$$

with $\Delta \mathrm{E_c}(\mathbf{x}^\mathrm{i}, \mathrm{y}_\mathrm{c}^\mathrm{i}, \mathrm{y_c}, \mathbf{w}) = \mathrm{E_c}(\mathbf{x}^\mathrm{i}, \mathrm{y}_\mathrm{c}^\mathrm{i}, \mathbf{w}) - \mathrm{E_c}(\mathbf{x}^\mathrm{i}, \mathrm{y_c}, \mathbf{w})$.

Perceptron approach. Perceptron learning is a simple approach for discriminative training which has been originally proposed for training linear classifiers (Rosenblatt, 1988) but can also be applied to graphical models (Collins, 2002). The idea can be extended to NeuroCRFs by considering the following loss term:

$$R_i^{Perc}(\mathbf{w}) = \max_{\mathbf{y} \in \mathcal{Y}} \sum_c \mathrm{E_c}(\mathbf{x}^\mathrm{i}, \mathrm{y}_\mathrm{c}^\mathrm{i}, \mathbf{w}) - \sum_\mathrm{c} \mathrm{E_c}(\mathbf{x}^\mathrm{i}, \mathrm{y_c}, \mathbf{w}) \tag{14}$$

which is very similar to the large margin criterion.

3.2 LEARNING

Due to non-convexity, initialization is a crucial step for NN learning, especially in the case of deep architectures (see (Erhan et al., 2009) for an analysis). Fortunately, an unsupervised greedy-wise pretraining algorithm for deep architectures has recently been proposed to tackle this problem with notable success (Hinton et al., 2006). (Bengio et al., 2006) provides a comprehensible analysis about greedy pretraining. We describe in detail NN initialization first, then we discuss fine tuning the NeuroCRF.

3.2.1 INITIALIZATION

Initialization of hidden layers in the NeuroCRF is done incrementally as it has been popularized for learning deep architectures. In our implementation, the deep-part of the NeuroCRF is initialized layer by layer in an unsupervised manner using restricted Boltzmann machines[4] (RBMs) as proposed by Hinton and colleagues (Hinton et al., 2006). Depending on the task, inputs may be real valued or binary valued, this may be handled by slightly different RBMs. We considered both cases in our experiments, while coding (hidden) layers always consisting of binary units.

Once a cascade of successive RBMs have been trained one at a time, one obtains a deep belief net which is then transformed into a feed forward NN which implements the deep-part of the NeuroCRF (without output layer). Once the deep-part is initialized, the NN is

[4]The NN weights are initialized with zero-mean Gaussian noise before RBM learning.

used to compute a high-level representation (i.e. the vector of activations on the last hidden-layer) of input samples. The CRF-part may then be initialized by training (in a supervised way) a linear CRF with this high-level coding of input samples. As we said, such a linear CRF is actually an output layer which is stacked over the deep part. The union of the weights of the deep-part and of the CRF-part constitutes an initialization solution $\mathbf{w}_0$ which is fine tuned, as described below, using supervised learning.

3.2.2 FINE TUNING

Fine tuning aims at learning the NeuroCRF parameters globally based on an initial and reasonable solution. None of the criterion we discussed earlier (Subsection 3.1) are convex since we naturally consider NNs with non linear (sigmoid) activation functions in hidden layers. However, provided one can compute an initial and reasonable solution and provided one can compute the gradient of the criterion with respect to NN weights, one can use any gradient-based optimization method such as stochastic gradient or bundle method to learn the model and reach a (eventually local) minimum. We show now how to compute the gradient with respect to the NN weights.

As long as $R_i(\mathbf{w})$ is continuous and there is an efficient method for computing $\frac{\partial R_i(\mathbf{w})}{\partial \mathrm{E_c}(\mathbf{x},\mathrm{y_c},\mathbf{w})}$ (this is true for all criteria discussed in the previous section) the (sub)gradient of $R(\mathbf{w})$ with respect to $\mathbf{w}$ can be computed with a standard backpropagation procedure. Let $\mathbf{E}_i$ be the set of energy outputs corresponding to input $\mathbf{x}^i$. Using the chain rule for every $\frac{\partial R_i(\mathbf{w})}{\partial \mathbf{w}}$:

$$\frac{\partial R(\mathbf{w})}{\partial \mathbf{w}} = \frac{1}{n}\sum_i \frac{\partial R_i(\mathbf{w})}{\partial \mathbf{w}} = \frac{1}{n}\sum_i \frac{\partial R_i(\mathbf{w})}{\partial \mathbf{E}_i}\frac{\partial \mathbf{E}_i}{\partial \mathbf{w}} \quad (15)$$

where $\frac{\partial \mathbf{E}_i}{\partial \mathbf{w}}$ is the Jacobian matrix of the NN outputs (for input $\mathbf{x}^i$) with respect to weights $\mathbf{w}$.

Then by setting $\frac{\partial R_i(\mathbf{w})}{\partial \mathbf{E}_i}$ as backpropagation errors of the NN output units, we can backpropagate and get $\frac{\partial R_i(\mathbf{w})}{\partial \mathbf{w}}$ using the chain rule over hidden layers.

3.2.3 REGULARIZATION AND SEMI-SUPERVISED LEARNING

In our implementation, we used the initial solution for building a quadratic regularization term of the form:

$$\Omega(\mathbf{w}) = \frac{1}{2}\|\mathbf{w} - \mathbf{w}_0\|^2 \quad (16)$$

The idea is that since the deep-part of a NeuroCRF is initialized in an unsupervised manner, we may expect using this solution for regularizing will avoid overfitting during fine tuning (we found that this gives better experimental results than using standard regularization by 0). Since the CRF-part is not initialized with a generative model, we regularize this part with 0 - both for initialization and fine-tuning.

Note that NeuroCRF permits semi-supervised learning in a very natural way. Indeed one may easily use unlabeled data for initializing the deep-part while labeled data are used in fine tuning only. One can expect that a good initialization of the deep-part will improve the global performance of a NeuroCRF, since the deep-part plays an important role of finding a relevant high-level representation of the input. It is a perspective of our work that we did not explore yet.

4 EXPERIMENTS

We performed experiments on two sequence labeling tasks with two well-known datasets. We first investigate the behaviour of NeuroCRFs in a first series of experiments on Optical Character Recognition with the OCR dataset (Kassel, 1995). Then we report experimental comparative results of NeuroCRFs and state of the art methods for the more complex task of automatic speech recognition using the TIMIT dataset (Lamel et al., 1986). In both cases we replicated experimental settings of previous works in order to get fair comparison, building on the compilation of Ben Taskar[5] for the OCR dataset, and using standard partitioning of the data and standard preprocessing for the TIMIT corpus. We use linear chain NeuroCRF for both tasks.

In fine-tuning we use a variant of our batch optimizer named non-convex regularized bundle method (Do & Artières, 2009) rather than a stochastic gradient procedure. As usual in bundle methods, our stopping condition is based on the Gap (G) between the Best observed objective function value (B) and the minimum of the approximation. We stop when the ratio of G/B is below 1%. This is a good trade-off for reaching low error-rates with a limited number of iterations (100 for OCR, 150 for ASR experiments).

4.1 OPTICAL CHARACTER RECOGNITION

OCR dataset consists of 6877 words which correspond to roughly 52K characters (Kassel, 1995; Taskar et al., 2004). OCR data are sequences of isolated characters (each represented as a binary vector of dimension 128) belonging to 26 classes. The dataset is divided in 10 folds for cross validation. We investigated two settings, using a large training set by training on 9 folds and testing on 1 fold (this is the *large* setting) and using a

[5] http://ai.stanford.edu/∼btaskar/ocr/

small one by training on 1 fold and testing on 9 folds (this is the *small* setting). Note that OCR results are cross-validation results, we do not use any extra validation set to set lambda.

We learned NeuroCRFs with one or two hidden layers. Transition energy outputs has only one connection to a bias unit, meaning that we do not use any input information for building transition energy. We learned standard RBMs for initializing the deep part of NeuroCRFs, which are learned with 50 iterations through the training set. Learning is performed using 1 step Contrastive Divergence.

Influence of network architecture. Figure 4 reports error rates gained on the *small* setting with NeuroCRF with one or two hidden layers of varying size. As can be seen, increasing the size of hidden layers improves performance for one hidden layer and two hidden layer NeuroCRFs. Also two hidden layers architectures systematically outperform single hidden layer architectures. Note that whatever the number of hidden layers, performance reaches a plateau when increasing the hidden layers' size. However the plateau is lower and reached faster for the two hidden layer architecture. These results suggest that increasing both the size of hidden layers and the number of hidden layers may significantly improve performance.

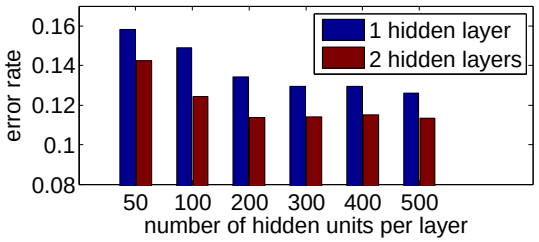

Figure 4: Influence of NN architecture on OCR dataset (*small* training set).

Accuracy. We compared the performance of two variants of NeuroCRFs, one trained with conditional maximum likelihood (CML), the other one trained with a large margin criterion (LM) with state of the art methods : linear and cubic M3Ns, linear CRFs, conditional neural fields (CNFs) with one hidden layer. NeuroCRFs have 2 hidden layers of 200 units each. Table 1 reports cross validation error rates of these models for the *small* setting and the *large* setting. We also report the performance of initial solutions (i.e. before fine tuning) for NeuroCRFs (in brackets).

NeuroCRFs significantly outperform all other methods, including M3N with non-linear kernel (whose results are not reported for the large setting due to scalability). Also looking at the performance of NeuroCRF before fine tuning shows that initialization by RBMs and CRFs indeed produce a good starting point, but fine tuning is essential for obtaining optimal performance. Finally, one sees here that both NeuroCRF training criteria are similar with a slight advantage of conditional likelihood criterion on large margin criterion. Surprisingly, we observed that the large margin criterion required more iterations than the conditional likelihood criterion[6]. In the following, we only considered NeuroCRFs trained with the CML criterion.

Table 1: Comparative error rates of NeuroCRF and state of the art methods on OCR dataset with either a small and a large training sets. Performance of NeuroCRF before fine tuning are indicated in brackets. Results of SVM cubic, M3N cubic and CNF come from (Taskar et al., 2004; Peng et al., 2009).

	small	large
CRF linear	0.2162	0.1420
M3N linear	0.2113	0.1346
SVM cubic	0.19	*not available*
M3N cubic	0.13	*not available*
CNF	0.131	*not available*
NeuroCRF^{CML}	**0.1080**(*0.1224*)	**0.0444**(*0.0697*)
NeuroCRF^{LM}	**0.1102**(*0.1221*)	**0.0456**(*0.0736*)

Note that (Perez-Cruz & Pontil, 2007) address the structured prediction with a different way and they are able to reach an error rate of 0.125 in small setting and 0.031 in large setting (using RBF kernel). Their approach considers an approximated problem that can be transformed into many multiclass SVM problems. Nevertheless, the non-linear SVM problems still scale quadratically with the problem size. Such a scaling prevents working with large data sets with millions of tokens (such as ASR experiments in next section).

Training time. We investigated the learning time of NeuroCRF and its scalability. We performed experiments on the OCR dataset both for *small* and *large* settings. Training time decomposes into RBM unsupervised learning, linear CRF initialization, and fine tuning the whole model. Roughly speaking RBM learning and fine tuning take about 45% of the time each, while CRF initialization takes about 10%. More importantly, overall training time in the *large* setting took about 10 times the training time in the *small* setting while the training dataset is 9 times bigger, which

[6]Actually we did not succeed at optimizing the large margin criterion. We conjecture that the parameter search space might be more complex in this case.

suggests a quasi-linear scaling with the problem size.

4.2 AUTOMATIC SPEECH RECOGNITION

We performed ASR experiments on the TIMIT dataset (Lamel et al., 1986) with standard train-test partitioning. The wave signal was preprocessed using the procedure described in (Sha & Saul, 2007), except that we do not use whitening by PCA. The 39-dimensional MFCC are simply normalized to have zero mean and unit variance. There are roughly 1.1 million frames in the training set, 120K frames and 57K frames respectively in the development and test sets. We used 2-layered NeuroCRFs trained with the CML criterion.

Handling continuous feature. Note here that inputs (real valued vectors of MFCC coefficients) are continuous while RBMs originally use binary logistic units for both visible and hidden variables. We used an extension of RBM for dealing with continuous variables that have Gaussian noise (In our implementation we consider a Gaussian noise with standard derivation 0.2) (Taylor et al., 2007). This *Gaussian-Binary* RBM was trained for 100 passes through the training data of 1.1M frames, using one step Contrastive Divergence. Once this first RBM is trained, we forward the input to the hidden layer and obtain a binary-logistic representation of the speech data, which is the inputs (i.e. visible data) for learning a second binary RBM. Since binary RBM converge much faster, we performed only 10 learning iterations through the training data for the second layer. The remaining initialization is performed as in section 3.2.1.

Table 2: Comparative phone recognition error rate on TIMIT dataset for discriminant and non discriminant HMM systems and for two hidden layer NeuroCRFs (of 500 or 1000 hidden units each) trained with CML.

	CDHMM				
	ML	CML	MCE	PT	LM
1 Gaussian	40.1	36.4	35.2	35.6	31.2
2 Gaussians	36.5	34.6	33.2	34.5	30.8
4 Gaussians	34.7	32.8	31.2	32.4	29.8
8 Gaussians	32.7	31.5	31.9	30.9	28.2
	NeuroCRF (CML)				
500x500	29.6				
1000x1000	29.1				

Results. Table 2 reports the phone error rates for CDHMMs and NeuroCRFs with increasing complexity (number of Gaussians in CDHMMs or number of hidden units in NeuroCRFs). We compared NeuroCRFs with non discriminant CDHMMs (i.e. Maximum Likelihood) and with state of the art approaches for learning CDHMMs with a discriminant criterion, Maximum Conditional Likelihood (CML) (Woodland & Povey, 2002), Minimum Classification Error (MCE) (Juang & Katagiri, 1992), Large margin (LM) (Sha & Saul, 2007), Perceptron learning (PT) (Cheng et al., 2009) (note that all results come from a compilation in (Sha & Saul, 2007) and from (Cheng et al., 2009)).

These results call for a few comments. First increasing the hidden layers' size improves the NeuroCRF error rate. Unfortunately we do not know if it still improves when using larger hidden layers (due to lack of time) but one can reasonably expect that even better results may be reached by using larger hidden layers and/or adding hidden layers. Second, NeuroCRFs outperform all other discriminant and non discriminant methods except Large Margin training of (Sha & Saul, 2007) when using up to 8 Gaussian distributions per state. While this may not look like an impressive result at first glance, we claim this result to be very promising. Indeed, all other systems in Table 2 rely on the learning of a preliminary CDHMM system, which is then used as initialization and/or for regularization. Hence all these systems integrate prior information from decades of research on how to learn and tune a non discriminant CDHMM for speech. In contrast NeuroCRF are trained from scratch with a non supervised initialization and a supervised fine tuning, they require no prior information.

Note that we did not compare NeuroCRF to multiple states per phone CDHMM systems as those traditionally used in ASR although such systems may reach better performance (especially for ML CDHMM). Reason is that comparison would have not been so fair. Indeed one may imagine extending this work in order to have multiple nodes per phone in NeuroCRF and one may expect this to provide even better results, that would be more directly comparable with multiples states per phone HMM systems.

5 CONCLUSION

We presented a model combining CRFs and deep NNs aiming at taking advantage of both the ability of deep networks to extract high level features and the discriminant power of CRFs for sequence labeling tasks. Results on OCR data show significant improvement over state of the art methods and demonstrate the relevance of the combination. On the larger scale speech recognition task our systems outperform most state of the art discriminant systems without relying on any prior in contrast to all other systems relying on an initial solution gained with a non discriminant criterion.

Acknowledgements

Authors acknowledge the support by PASCAL 2 EU Network of Excellence.

References

Altun, Y., Johnson, M., & Hofmann, T. (2003). Investigating loss functions and optimization methods for discriminative learning of label sequences. *EMNLP*.

Bengio, Y., Lamblin, P., Popovici, D., & Larochelle, H. (2006). *Greedy layer-wise training of deep networks* (Technical Report 1282). Université de Montréal.

Bengio, Y., & LeCun, Y. (2007). Scaling learning algorithms towards ai. In *Large scale kernel machines*. Cambridge, MA: MIT Press.

Bottou, L., Bengio, Y., & LeCun, Y. (1997). Global training of document processing systems using graph transformer networks. *In Proc. of Computer Vision and Pattern Recognition* (pp. 490–494). Puerto-Rico. IEEE.

Cheng, C.-C., Sha, F., & Saul, L. K. (2009). Matrix updates for perceptron training of continuous density hidden markov models. *ICML* (pp. 153–160).

Collins, M. (2002). Discriminative training methods for hidden markov models: theory and experiments with perceptron algorithms. *EMNLP* (pp. 1–8).

Collobert, R., Sinz, F., Weston, J., & Bottou, L. (2006). Trading convexity for scalability. *ICML*.

Do, T.-M.-T., & Artières, T. (2009). Large margin training for hidden Markov models with partially observed states. *ICML* (pp. 265–272). Omnipress.

Erhan, D., Manzagol, P.-A., Bengio, Y., Bengio, S., & Vincent, P. (2009). The difficulty of training deep architectures and the effect of unsupervised pre-training. *AISTATS*.

Graves, A., Fernández, S., Gomez, F., & Schmidhuber, J. (2006). Connectionist temporal classification: labelling unsegmented sequence data with recurrent neural networks. *ICML* (pp. 369–376).

Hinton, G. E., Osindero, S., & Teh, Y.-W. (2006). A fast learning algorithm for deep belief nets. *Neural Computation*, *18*, 1527–1554.

Juang, B., & Katagiri, S. (1992). Discriminative learning for minimum error classification. *IEEE Trans. Signal Processing, Vol.40, No.12*.

Kassel, R. H. (1995). *A comparison of approaches to on-line handwritten character recognition*. Doctoral dissertation, Cambridge, MA, USA.

Lafferty, J. (2001). Conditional random fields: Probabilistic models for segmenting and labeling sequence data. *ICML* (pp. 282–289). Morgan Kaufmann.

Lafferty, J., Zhu, X., & Liu, Y. (2004). Kernel conditional random fields: representation and clique selection. *ICML*.

Lamel, L., Kassel, R., & Seneff, S. (1986). Speech database development: Design and analysis of the acoustic-phonetic corpus. *DARPA* (pp. 100–110).

LeCun, Y., Bottou, L., Bengio, Y., & Haffner, P. (1998). Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, *86*.

Mccallum, A., Freitag, D., & Pereira, F. (2000). Maximum entropy markov models for information extraction and segmentation. *ICML* (pp. 591–598).

Peng, J., Bo, L., & Xu, J. (2009). Conditional neural fields. *NIPS*.

Perez-Cruz, F, G. Z., & Pontil, M. (2007). Conditional graphical models. In G. H. Bakir, T. Hofmann, B. Schlkopf, A. J. Smola, B. Taskar and S. V. N. Vishwanathan (Eds.), *Predicting structured data*. MIT Press.

Qi, Y., Kuksa, P. P., Collobert, R., Sadamasa, K., Kavukcuoglu, K., & Weston, J. (2009). Semi-supervised sequence labeling with self-learned features. *ICDM'09*. IEEE.

Rosenblatt, F. (1988). The perceptron: a probabilistic model for information storage and organization in the brain. *Neurocomputing: foundations of research*, 89–114.

Sato, K., & Sakakibara, Y. (2005). Rna secondary structural alignment with conditional random fields. *ECCB/JBI* (p. 242).

Sha, F., & Saul, L. K. (2007). Large margin hidden markov models for automatic speech recognition. *NIPS 19* (pp. 1249–1256). MIT Press.

Taskar, B., Guestrin, C., & Koller, D. (2004). Max-margin markov networks. *NIPS 16*. MIT Press.

Taylor, G. W., Hinton, G. E., & Roweis, S. T. (2007). Modeling human motion using binary latent variables. In *Nips*, 1345–1352. MIT Press.

Woodland, P., & Povey, D. (2002). Large scale discriminative training of hidden markov models for speech recognition. *Computer Speech and Language*, *16*, 25–47(23).