# Learning in pair-wise CRFs - a tutorial. Application in entity linking

Octavian Ganea

December 1, 2014

## General Problem Statement

We are interested in learning the parameters $w = \{w_a\}_{a=1}^A$ of a conditional distribution $p(\boldsymbol{y}|\boldsymbol{x})$ given a learning sample of fully observed in-out pairs $D = \{(\boldsymbol{x}^{(t)}, \boldsymbol{y}^{(t)})\}_{t=1}^N$.

The general model of $p$ is determined by a graph $G$ with $A$ factors $\{\Psi_a\}_{a=1}^A$:

$$p(\boldsymbol{y}|\boldsymbol{x}) = \frac{1}{Z(\boldsymbol{x})} \prod_{a=1}^{A} \Psi_a(\boldsymbol{y}_a, \boldsymbol{x}_a) \tag{1}$$

where each factor depends on a subset of all nodes $\boldsymbol{y}_a \subseteq \boldsymbol{y}$ and it is modelled in an exponential form:

$$\Psi_a(\boldsymbol{y}_a, \boldsymbol{x}_a) = exp\{w_a^T f_a(\boldsymbol{y}_a, \boldsymbol{x}_a)\} \tag{2}$$

where $w_a$ is a real valued parameter vector and $f_a$ is a vector of features over the variables in set $a$. Maximum likelihood estimation (MLE) of $w$ is:

$$w^* = \underset{w}{\text{argmax}} \mathcal{L}(w) \qquad \mathcal{L}(w) := \sum_{t=1}^{N} \left( \sum_{a=1}^{A} w_a^T f_a(\boldsymbol{y}_a^t, \boldsymbol{x}_a^t) - logZ(\boldsymbol{x}^t) \right) \tag{3}$$

which is intractable for general graphs because of the difficult evaluation of the partition function $Z(\boldsymbol{x}) = \sum_{\boldsymbol{y'}} \prod_{a=1}^{A} exp\{w_a^T f_a(\boldsymbol{y'}_a, \boldsymbol{x}_a)\}$.

## Entity linking case

In the case of an entity linking pipeline, a document is represented by a vector of observed mentions $\boldsymbol{x}$ and by a vector of hidden entities $\boldsymbol{y}$ that we want to determine. Each single RV $y_j$ takes values from a set of labels/entities $L = \{1, \dots, K\}$,

The training data consists of triplets $D = \{(\boldsymbol{x}^{(t)}, \boldsymbol{y}^{(t)}, l^{(t)})\}_{t=1}^N$, where $l^{(t)}$ is the (variable) number of mentions in a document. The graphical model in this case is a pair-wise CRF:

$$p(\boldsymbol{y}|\boldsymbol{x}) = \frac{1}{Z(\boldsymbol{x})} exp\left\{ \frac{1}{l} \sum_{j=1}^{l} \rho_{x_j, y_j} + \frac{1}{\binom{l}{2}} \sum_{i<j} \lambda_{y_i, y_j} \right\} \tag{4}$$

Eq 4 can be rewritten as eq 1 using only unary and binary factors, unary vector of features being

$$f_j(y_j, x_j) = \{f_{k,m}(y_j, x_j)\}_{k=1\dots K; m \in dict} \qquad f_{k,m}(y_j, x_j) = \frac{1}{l} \mathbb{1}[y_j = k, x_j = m]$$

and the binary vector of features being

$$f_{i,j}(y_i, y_j, x_i, x_j) = \{f_{k,h}(y_i, y_j)\}_{k=1\dots K, h=1\dots K} \qquad f_{k,h}(y_i, y_j) = \frac{1}{\binom{l}{2}} \mathbb{1}[y_i = k, y_j = h]$$

Note that, in this case, the set of parameters $w$ that we wish to learn is composed of $(\rho_{m,k})_{k=1...K;m\in dict}$ and $(\lambda_{k,h})_{k=1...K,h=1...K}$ .

The above formulation is similar to the formalization described in [Alahari 2010].

## First training method: Pseudolikelihood

MLE is not possible without some kind of computation of the partition function $Z(\boldsymbol{x})$. One approximation method is pseudolikelihood (described in detail in [McCallum 2007], [Alahari 2010], [Hyvarinen 2006], [McCallum 2007] and [Murphy 2006]).

Pseudolikelihood simultaneously classifies each node conditioned on all its neighbours from the graph. We follow the notations from [McCallum 2007]. For any variable s, denote by $N(s)$ the set of its neighbours and by $a \ni s$ the set of factors that depend on variable s. Then the pseudolikelihood approximates $\mathcal{L}(w)$ from eq 3 and does this by approximating:

$$log\ p(\boldsymbol{y}|\boldsymbol{x}) \approx \sum_{s\in G} log\ p(y_s|y_{N(s)}, \boldsymbol{x}) \tag{5}$$

where

$$log\ p(y_s|y_{N(s)}, \boldsymbol{x}) = \frac{\prod_{a\ni s} \Psi_a(y_s, y_{N(s)}, \boldsymbol{x}_a)}{\sum_{y'_s} \prod_{a\ni s} \Psi_a(y'_s, y_{N(s)}, \boldsymbol{x}_a)} \tag{6}$$

Thus, eq 3 reduces to maximising the following function:

$$\mathcal{L}_{PL}(w) = \sum_{t=1}^{N} \sum_{s\in G^{(t)}} log\ p(y_s^{(t)}|y_{N(s)}^{(t)}, \boldsymbol{x}^{(t)}; w) \tag{7}$$

**Computational expenses**
This formula is computationally cheap and is used especially for cases where graph factors have many variables (which is not the case of our entity linking model).

**Performance**
[Murphy 2006] states that: "Note that pseudo-likelihood estimates the parameters condition on i's neighbors being observed. As a consequence, PL tends to place too much emphasis on the edge potentials, and not enough on the local evidence. For image denoising problems, this is often evident as "oversmoothing". [...] Regularizing the edge parameters does help."

More important, the performance of pseudolikelihood seems to be worse than that of piecewise training (that we will se below) in practice. As [McCallum 2007] notes, "On the NLP data we consider in this paper, piecewise performs better than pseudolikelihood, sometimes by a very large amount. So piecewise training can have good accuracy, however, unlike pseudolikelihood it does not scale well in the variable cardinality."

**Theoretical guarantees**
[McCallum 2007] states that : "It is a well known result that if the model family includes the true distribution, then pseudolikelihood converges to the true parameter setting in the limit of infinite data ([Gidas 1988], [Hyvarinen 2006])".

[Hyvarinen 2006] prove that, in the context of fully visible Boltmann machines, the gradient of $\mathcal{L}_{PL}$ is zero at the true distribution parameters. They also prove that, under one additional assumption, the point of zero gradient is the global maximum of $\mathcal{L}_{PL}$.

## Second training method: Piecewise training

Again following notations from [McCallum 2007], piecewise training approximates $\mathcal{L}(w)$ from eq 3 and does this by:

$$log\ p(\boldsymbol{y}|\boldsymbol{x}) \approx \sum_{a=1}^{A} log \frac{\Psi_a(\boldsymbol{y}_a, \boldsymbol{x}_a)}{\sum_{\boldsymbol{y'}_a} \Psi_a(\boldsymbol{y'}_a, \boldsymbol{x}_a)} \tag{8}$$

Each term in (8) corresponds to a single piece or factor of the graph. That term would be the exact likelihood if the rest of the graph would be discarded.

**Computational expenses**

As said before, piecewise training is more expensive than pseudolikelihood, especially if the graph contains factors with many variables. However, in the case of our entity linking model, all the factors have at most 2 variables, which is fine.

**Theoretical guarantees**

[McCallum 2008] provides a detailed theoretical analysis.

In addition, [McCallum 2007] states that : "Another way of viewing piecewise training is that it is equivalent to approximating log Z by the Bethe energy with uniform messages, as would be the case after running 0 iterations of BP".

**Performance**

As stated in [McCallum 2007] and [McCallum 2008], piecewise training performs most of the times better than pseudolikelihood.

## Third training method: spanning-trees

A new method proposed by the writer which works just for graphs containing only unary or binary potentials, like it is the case of our entity linking model, that is, $p(\boldsymbol{y}|\boldsymbol{x}) = \frac{1}{Z(\boldsymbol{x})} \exp\left\{\sum_{i<j} f(y_i, y_j) + \sum_i g(x_i, y_i)\right\}$.

We know that, if the graph G that gives the formula of $p(\boldsymbol{y}|\boldsymbol{x})$ would be a tree, the computation of $log\ Z(\boldsymbol{x})$ would be exact and could be done in polynomial time.

In the case of general graphs G, we can see the graph G as the superposition of all its possible spanning trees. Formally, if we define SP(G) to be the set of all spanning trees of G, we can use the following approximation:

$$log\ p(\boldsymbol{y}|\boldsymbol{x}) \approx \frac{1}{|SP(G)|} \sum_{T \in SP(G)} log\ p_T(\boldsymbol{y}|\boldsymbol{x}) \tag{9}$$

where $p_T(\boldsymbol{y}|\boldsymbol{x})$ is the same as $p(\boldsymbol{y}|\boldsymbol{x})$ but computed just on the spanning tree T:

$$p_T(\boldsymbol{y}|\boldsymbol{x}) = \frac{1}{Z_T(\boldsymbol{x})} \exp\left\{\sum_{i \sim_T j} f(y_i, y_j) + \sum_i g(x_i, y_i)\right\} \tag{10}$$

An SGD update on this model will work as follows: first select a random training pair $(\boldsymbol{x}^{(t)}, \boldsymbol{y}^{(t)})$, then select a random spanning tree T from $SP(G)$ and finally update the parameters $w$ using the gradient of $p_T(\boldsymbol{y}^{(t)}|\boldsymbol{x}^{(t)})$ that can be computed exactly in polynomial time.

**Computational expenses**

For conditionals decomposed just as unary and binary potential functions, the above computation has polynomial complexity.

**Theoretical guarantees**

Not analysed.

### Fourth training method: struct SVM

This method does not do MLE, but structured prediction, that is, it tries to find parameters $w$ that would best predict $y$ given $x$ in the following:

$$y^* = \underset{y}{\text{argmax}}\ w^T f(x, y)$$

Given a training set $D = \{(x^{(t)}, y^{(t)})\}_{t=1}^{N}$, this method tries to find $w$ that maximize the margin between $w^T f(x^{(t)}, y^{(t)})$ and $\max_{y \neq y^{(t)}} w^T f(x^{(t)}, y)$ for each $i = 1 \dots N$.

Following a similar notation as in Section 2 of [Online SSVM 2007], we want to minimize the following regularized loss function:

$$L(w) = \frac{1}{N} \sum_{t=1}^{N} \left( \max_{y} \left( w^T f(x^{(t)}, y) + \Delta(y, y^{(t)}) \right) - w^T f(x^{(t)}, y^{(t)}) \right) + \frac{\lambda}{2} \|w\|^2$$

which can be done fast by SGD and using Loopy Belief Propagation to find the maximum of the margin in the above loss function. The distance function $\Delta(y, y')$ should decompose based on the same graph as $f(x, y)$ and can be a simple cardinality difference function.

**Computational expenses**

The running time is slowed down compared to the previous approaches by the inference step that is needed in each SGD update to find the max margin. But still, the entire computation has polynomial complexity.

**Theoretical guarantees**

[Pletscher 2010] looked at the correlation between max-margin-loss and log-loss (ML estimator). [Online SSVM 2007] and [Vechev 2015] provide other useful insights.

**Performance**

[Vechev 2015] has a very similar setting with our entity linking model. They succesfuly used Struct SVM to train the parameters of their model.

## References


[Alahari 2010] Alahari, Russell, Torr. "Efficient piecewise learning for conditional random fields."

[McCallum 2007] Sutton, McCallum "Piecewise Pseudolikelihood for Efficient Training of Conditional Random Fields."

[Hyvarinen 2006] A. Hyvarinen "Consistency of pseudolikelihood estimation of fully visible Boltmann machines."

[McCallum 2008] Sutton, McCallum "Piecewise Training for Structured Prediction."

[Murphy 2006] Vishwanathan, Schraudolph, Schmidt, Murphy "Accelerated Training for Conditional Random Fields with Stochastic Gradient Methods."

[Gidas 1988] Gidas "Consistency of maximum likelihood and pseudolikelihood estimators for gibbs distributions."

[Vechev 2015] V. Raychev, M.Vechev, A. Krause "Predicting program properties from "Big Code"." `http://www.srl.inf.ethz.ch/papers/jsnice15.pdf`

[Online SSVM 2007] N. Ratliff, J. Bagnell, M.Zinkevich "(Online) Subgradient Methods for Structured Prediction"

[Taskar 2004] Taskar, Guestrin, Koller "Max-Margin Markov Networks"

[Pletscher 2010] Pletscher, Ong, Buhmann "Entropy and margin maximization for structured output learning"