Probabilistic Graphical Models - MVA 2017/2018
Review on Conditional Random Fields

Victor Busa, Rémi Lespinet

1 Introduction

In a setting where we consider distributions over a set of random variables that can be split into a set of observed random variables X (the entities) and a set of latent random variables Y (the features), we have seen that traditional graphical models (directed and undirected) represent the joint probability distribution $p(x, y)$ of the variables. This involves choosing a model that represents well the dependencies between the observed data. This dependency is sometimes complex, hence difficult to model, and critical for the performance of the model.

In a case where our primary goal is to solve the classification problem (e.g we want to predict classes $y_1, \dots y_N$ given an observation $x_1, \dots x_M$), we can avoid modeling these complex dependences between observed random variables and directly model $p(y \mid x)$. This is the solution proposed by Conditional Random Fields (CRF).

Before the use of Neural network, CRFs were a state-of-the-art technique on lots of natural language processing tasks such as part-of-speech tagging (**lafferty2001**), shallow parsing (**sha2003**), named entity recognition (**mcdonald2005**). They have also been applied to object recognition (**quattoni2005**), image labeling (**he2004**) and gene prediction (**decaprio2007**).

In this review we will briefly provide the intuition behind CRFs by explaining differences between generative and discriminative models through the examples of logistic regression and naive bayes models, we then focus on linear-chain conditional random filed (LRCRF), where we discuss the techniques used for parameter estimation and inference tasks. We then briefly present the general setting for CRFs.

2 Generative vs Discriminative models

As we have seen, a difference between Naive Bayes and Logistic regression, is that the former is generative e.g. it models $p(y, x)$, whereas the later is discriminative e.g. it models $p(y|x)$.

As briefly introduced, by using a discriminative model, we can remain agnostic about the form of $p(x)$, which is often difficult to model and not necessary for classification. This explains why the logistic regression outperforms the Naive Bayes model on average in a range of tasks.

In fact we see easily that interpreting the Logistic regression generatively leads to Naive Bayes, and conversely, interpreting the conditional probability of the Naive Bayes model leads to Logistic regression. We say that these two models form a "generative-discriminative" pair.

In this regard, one question arise as to what the discriminative counter part for HMMs are. As we will see in the next sections, linear-chain CRFs provide an answer to this question.

3 Linear-chain conditional random fields

In the lectures, we have seen that a Hidden Markov model (HMM) can be thought as a generalization of a Naïve Bayes model that captures dependencies between sequential data. We have also discussed the advantages of discriminative models for classification tasks. In this section, we present the definition of *linear-chains* CRFs that combine both these characteristics, and we see that *linear-chains* CRFs can be seen as an extension of HMMs.

A linear-chain conditional random field is a distribution that takes the form:

$$p(x\,|\,y) = \frac{1}{Z(\eta)} \exp\left(\sum_{t=1}^{T}\sum_{k=1}^{K} \eta_k \phi_k(y_{t-1}, y_t, x_t)\right) \tag{1}$$

Let's verify that linear-chain CRF generalizes the HMM. The paper exposes the case where emission probabilities are discrete, but it actually works for the more general case where emission probabilities are in exponential family form. Let us consider the following emission probability:

$$p(x_t\,|\,y_t = i) = \frac{1}{Z(\mu)} \exp\left(\mu_i^T \phi_i(x_t)\right) \tag{2}$$

We can write

$$p(x, y) = \frac{1}{Z(\eta)} \exp\left(\sum_{t=1}^{T}\sum_{i\in S}\sum_{j\in S} \lambda_{i,j} 1_{\{y_t=i\}} 1_{\{y_{t-1}=j\}} + \sum_{t=1}^{T}\sum_{i\in\mathcal{S}} 1_{\{y_t=i\}} \mu_i^T \phi_i(x_t)\right) \tag{3}$$

We see that it takes the following form:

$$p(x, y) = \frac{1}{Z(\eta)} \exp\left(\sum_{t=1}^{T}\sum_{k=1}^{K} \eta_k \phi_k(y_{t-1}, y_t, x_t)\right)$$

Hence

$$p(x\,|\,y) = \frac{p(x,y)}{\sum_{x\in\mathcal{S}} p(x,y)} = \frac{1}{Z(x,\eta)} \exp\left(\sum_{t=1}^{T}\sum_{k=1}^{K} \eta_k \phi_k(y_{t-1}, y_t, x_t)\right) \tag{4}$$

3.1 Parameters estimations

In this section we study how to estimate the parameters $\theta = \{\lambda_k\}$ of a *linear-chain* CRF. Given a i.i.d. training data $\mathcal{D} = \{x^{(i)}, y^{(i)}\}_{i=1}^N$, where $x^{(i)} = \{x_1^{(i)}, \ldots, x_T^{(i)}\}$ is a sequence of inputs and each $y^{(i)} = \{y_1^{(i)}, \ldots, y_T^{(i)}\}$ is a sequence of the desired predictions. Since CRFs model the conditional log-likelihood, we are interested in the vector θ that maximizes the conditional log-likelihood. e.g that maximizes

$$\ell(\theta) = \sum_{i=1}^{N} \log\ p(y^{(i)}\,|\,x^{(i)}) \tag{5}$$

replacing the log probability by its expression (1), we seek to maximize:

$$\ell(\lambda) = \underbrace{\sum_{i=1}^{N}\sum_{t=1}^{T}\sum_{k=1}^{K} \lambda_k f_k(y_t^{(i)}, y_{t-1}^{(i)}, x_t^{(i)})}_{\mathcal{A}} - \underbrace{\sum_{i=1}^{N} \log\left(\underbrace{\sum_{y\in\mathcal{Y}} \exp\left(\sum_{t=1}^{T}\sum_{k=1}^{K} \lambda_k f_k(y_t^{(i)}, y_{t-1}^{(i)}, x_t^{(i)})\right)}_{\mathcal{Y}_\lambda(x)}\right)}_{\mathcal{B}} - \underbrace{\sum_{k=1}^{K} \frac{\lambda_k^2}{2\sigma^2}}_{\mathcal{C}}$$

To avoid overfitting we penalized the log-likelihood with the regularization term denoted by $\mathcal{C}$ (See (**chen2000**)). The log-likelihood function is concave (the log-sum exp function is convex). The regularized term being strictly concave, the regularized version of the log likelihood is strictly concave, and we can minimize using standard gradient ascent. Computing the partial derivatives of $\ell(\theta)$ w.r.t to the parameters λ_p leads to

$$\frac{\partial \ell}{\partial \lambda_p}(\lambda) = \sum_{i=1}^{N}\sum_{t=1}^{T} f_p(y_t^{(i)}, y_{t-1}^{(i)}, x_t^{(i)}) + \sum_{i=1}^{N}\sum_{y\in\mathcal{Y}} p_\lambda(y|x)\sum_{t=1}^{T} f_p(y_t^{(i)}, y_{t-1}^{(i)}, x_t^{(i)}) - \frac{\lambda_p}{\sigma^2} \tag{6}$$

We see that we need the marginal probabilities $p_\lambda(y|x)$ in order to compute the gradient, which is the purpose of the next section.

3.2 Inference

As for HMMs, the inference problem on Linear-Chain CRFs can be solved exactly using dynamic programming. In this section, we use a generalization of the forward-backward algorithm [1] to compute the marginal probabilities. We also discuss the generalization of the Viterbi algorithm that can be used to compute the most probable sequence $y_1 \dots y_N$. Deriving the expression of (1), linear-conditional random fields can be written as:

$$p_\lambda(y|x) = \frac{1}{Z_\lambda(x)}\prod_{t=1}^{T}\exp\left(\sum_{k=1}^{K}\lambda_k f_k(y_t, y_{t-1}, x_t)\right) \triangleq \frac{1}{Z_\lambda(x)}\prod_{t=1}^{T}\Psi_t(y_t, y_{t-1}, x_t) \tag{7}$$

As for the HMMs we can take advantage of the structure of the linear-chain CRF and save an exponential number of computations by writing

$$p(x) = Z_\lambda(x) = \sum_{y_T}\sum_{y_{T-1}}\Psi_T(y_T, y_{T-1}, x_T)\ \dots \overbrace{\sum_{y_1}\Psi_2(y_2, y_1, x_2)\underbrace{\sum_{y_0}\Psi_1(y_1, y_0, x_1)}_{\alpha_1(y_1)}}^{\alpha_2(y_2)} = \sum_{y_T}\alpha_T(y_T) \tag{8}$$

The backwards recursion is done exactly the same way by stacking the sums in the reverse order in (8). This gives the following formulas.

$$\begin{cases} \alpha_t(j) \triangleq p(x_1, \dots, x_t) = \sum_{i\in\mathcal{S}}\Psi_t(j, i, x_t)\alpha_{t-1}(i) \\ \beta_t(j) \triangleq p(x_{t+1}, \dots, x_T) = \sum_{j\in\mathcal{S}}\Psi_{t+1}(j, i, x_{t+1})\beta_{t+1}(j) \end{cases} \tag{9}$$

These formulas are identical to the formula for HMMs seen in the course, the difference between the two is captured in the definition of Ψ.

> **Note :** We can easily derive the Viterbi algorithm for the linear-chain CRFs by replacing sum with max in the recurence formulas. The algorithm then computes the most probable assignment $y^* = \arg\max_y p(y|x)$

4 General CRFs

General CRFs are easily defined, we say that $p(y \mid x)$ is a conditional random field if it factorizes according to a factor graph G. If the factors of G is in the exponential family form, we can write

$$p(y|x) = \frac{1}{Z(x)}\prod_{\Psi_A\in G}\exp\left(\lambda_A^T f_A(x_A, y_A)\right) \tag{10}$$

[1] used and implemented in homework 3

We see that we can recover the general formulation of a CRF using the undirected graphical model :

$$p(y|x) = \frac{p(x,y)}{p(x)} = \frac{p(x,y)}{\sum_{y'} p(y',x)} = \frac{\frac{1}{Z}\prod_A \Psi_A(x_A, y_A)}{\frac{1}{Z}\sum_{y'}\prod_A \Psi_A(x_A, y'_A)} = \frac{1}{Z(x)}\prod_A \Psi_A(x_A, y_A) \tag{11}$$

In practice parameters are often tied[2]. So we can partition the factors of a graph G into $\mathcal{C} = \{C_1, \ldots, C_P\}$, where each C_p is a (clique template) set of factors which has a corresponding set of sufficient statistics $f_{pk}(x_p, y_p)$ and θ_p. Using this notations, CRF can be formulated as:

$$p(y|x) = \frac{1}{Z(x)}\prod_{C_p \in \mathcal{C}}\prod_{\Psi_c \in C_p} \Psi_c(x_c, y_c; \theta_p) \triangleq \frac{1}{Z(x)}\prod_{C_p \in \mathcal{C}}\prod_{\Psi_c \in C_p} \exp\left(\sum_{k=1}^{K(p)} \lambda_{pk} \underbrace{f_{pk}(x_c, y_c)}_{\text{sufficient statistics}}\right) \tag{12}$$

In a CRF we are not tied to sequential transition as we can add a factor (that models the direct dependencies between nodes) between any 2 nodes in the graph. Intuitively, the bigger the factors f_{pk} is, the more likely seeing y_c associated with the observation x_c is going to happen in the model.

4.1 Inexact inference: general CRFs

Exact inference algorithms exist in the case of general CRFs and are based on the junction tree algorithm that we've briefly discussed in class. Indeed, the idea is to eliminate cycles by constructing a tree. This is done by clustering potential functions over each cliques. We can then use the Junction Tree algorithm to do exact inference on any kind of model. For complex models however, the join potential over a clique grows exponentially with the number of features, which makes the problem $\mathcal{NP}$-hard in the worst case.

When the exact inference is intractable, one can rely on inexact approaches such as Markov Chain Monte Carlo[3], or even the sum-product algorithm : when the graph is not a tree, we can iterate, updating all messages at each iteration. This method, called loopy belief propagation is not guaranteed to give a good approximation of the marginal, and not even to converge, but often give good results in practice.

5 Conclusion

As we have seen, CRFs allow to model conditional probabilities using a graphical structure. For classification task, this provides a better representation, because it avoids modeling complex dependencies between the features. For this reason, CRFs have been successfully applied in a large variety of fields including text processing and computer vision. In practice, inference in CRF is $\mathcal{NP}$ hard in general, which is why a lot of the studies focus on a subset of CRFs, in which inference is easier. In particular, we have presented linear-chain CRFs, which are a natural choice for modeling sequential data. We have seen their link with HMMs and how we can derive algorithms to solve efficiently the inference problem and the parameter estimation problem.
For non sequential CRFs models, the algorithms derived cannot be used and we must rely on approximate methods. These algorithms might need a lot of iterations and are not always tractable in practice, For this reason, latest researches in the field focus on finding more efficient techniques for parameter estimation.

[2]For example in homework 3 we used a fixed transition matrix
[3]for example Gibbs sampling