Conditional Random Field Notes

Laura Ruis

November 8, 2018

Abstract

Notes on implementing a linear-chain CRF for POS tagging and a general CRF for dependency parsing. These implementations involve the forward-backward algorithm, Viterbi decoding and Tutte's Matrix Tree Theorem. The first two sections briefly go over the theory behind structured prediction and conditional random fields, and the final two sections go deeper into the implementation of these models for two tasks.

Contents

1 Structured Prediction

Structured prediction is a framework for solving problems of classification or regression in which the output variables are mutually dependent or constrained. It can be seen as a subset of prediction where we make use of the structure in the output space. The loss function used in structured prediction considers the output as a whole. If we denote the input by x, the output space (set of all possible outputs) by $\mathcal{Y}$ and $f(x, y)$ some function that expresses how well y fits x, then prediction can be denoted by the following formula:

$$y^* = arg \max_{y \in \mathcal{Y}} f(x, y)$$

The distinction between regular classification and structured prediction lies in the fact that for the latter case some kind of search needs to be done over the output space, namely finding the correct $y \in \mathcal{Y}$. An exhaustive search would be infeasible and structure prediction methods are needed.
In dependency parsing, x would correspond to a sentence, y to a dependency structure (tree) and $\mathcal{Y}$ to all possible dependency structures. A conditional random field (CRF) is a probabilistic model very useful for structured prediction.

2 Conditional Random Field

A conditional random field is a probabilistic graphical model that combines advantages of discriminate classification and graphical models[4].

2.1 Discriminative vs. Generative

In the *discriminative* approach to classification we model $p(y \mid x)$ directly as opposed to the *generative* case where we model $p(x \mid y)$ and use it together with Bayes' rule for prediction. If we model the conditional directly, dependencies among x itself play no rule, resulting in a much simpler inference problem than modeling the joint $p(y, x)$. Generative models describe how some label y can generate some feature vector x, whereas discriminative models directly describe how to assign a feature vector x a label y.

2.2 Graphical Models

Graphical models represent complex distributions over many variables as a product of local factors of smaller subsets of variables. Based on what kind of graphical structure one assumes, one can use linear chain CRFs or more general CRFs. Generative models are often most naturally represented by directed graphical models ($p(x, y) = p(y)p(x \mid y)$), whereas discriminative models are most naturally represented by undirected graphical models. CRFs can be represented as *factor graphs*.
A factor graph is a graph $G = (V, F, E)$, in which V denotes the random variables and F denotes the factors. A factor graph describes how a complex probability distribution factorizes, and thus imposes independence relations. A distribution $p(\mathbf{y})$ factorizes according to a factor graph G if there exists a set of local functions ψ_a such that p can be written as:

$$p(\mathbf{y}) = \frac{1}{Z} \prod_{a \in F} \psi_a(y_{N(a)})$$

The factors ψ_a are each only dependent on the neighbors y_i of a in the factor graph. The factors need to be larger than zero ($\psi_a(y_{N(a)}) \geq 0$), but need not have a probabilistic interpretation. The trick in using a factor graph for a task like classification is defining a set of feature functions that are nonzero only for a single class.

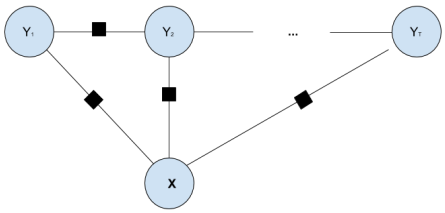

Figure 1: An instance of a linear-chain CRF

2.3 Linear-Chain CRF

A linear-chain CRF is just a way of modeling the conditional probability distribution $p(\mathbf{y} \mid \mathbf{x})$ with a specific choice of feature functions and specific independence assumptions.
If we define $\boldsymbol{\theta}$ as parameters, $\mathcal{F} = \{f_k(y, y', \mathbf{x}_t)\}_{k=1}^K$ as a set of real-valued feature functions, a *linear-chain CRF* is conditional probability distribution that factorizes according to:

$$p(\mathbf{y} \mid \mathbf{x}) = \frac{1}{Z(x)} \prod_{t=1}^{T} \underbrace{\exp\left\{\sum_{k=1}^{K} \boldsymbol{\theta}_k f_k(y_t, y_{t-1}, \mathbf{x}_t)\right\}}_{\text{Factors in a FG: } \psi_t(y_t, y_{t-1}, \mathbf{x}_t)}$$

$$Z(\mathbf{x}) = \sum_{\mathbf{y}} \prod_{t=1}^{T} \exp\left\{\sum_{k=1}^{K} \boldsymbol{\theta}_k f_k(y_t, y_{t-1}, \mathbf{x}_t)\right\}$$

This is a factor graph with factors $\psi_t(y_t, y_{t-1}, \mathbf{x}_t)$, as shown in Figure 1. The dependence of each y_t on the entire input sequence is a modeling assumption. In the equation, each factor contains a vector $\mathbf{x}_t$ that can contain all information necessary for computing the features at time step t, which can include features from all other time steps. The graphical model underlying the particular linear-chain CRF in Figure 1 implies two independence assumptions:

1. Each state is only dependent on its immediate predecessor and x: $y_t \perp\!\!\!\perp \{y_1, \ldots, y_{t-2}\} \,|\, y_{t-1}, \mathbf{x}$
2. Each observation x_t only depends on the current state y_t: $x_t \perp\!\!\!\perp \{x_i\}_{i \neq t}, \{y_i\}_{i \neq t} \mid y_t, , \mathbf{x}$

2.4 Training a Linear-Chain CRF

A linear-chain CRF can be trained with maximum likelihood estimation. The log-likelihood over some dataset with N datapoints is:

$$\begin{aligned}
\log \mathcal{L}(\boldsymbol{\theta}) &= \log \prod_{i=1}^{N} p(\mathbf{y}^{(i)} \mid \mathbf{x}^{(i)}, \boldsymbol{\theta}) \overset{\text{FG}}{=} \log \left[\prod_{i=1}^{N} \frac{1}{Z(\mathbf{x}^{(i)})} \prod_{t=1}^{T} \psi_t(y_t^{(i)}, y_{t-1}^{(i)}, \mathbf{x}_t^{(i)})\right] \\
&\overset{\text{CRF}}{=} \log \left[\prod_{i=1}^{N} \frac{1}{Z(\mathbf{x}^{(i)})} \prod_{t=1}^{T} \exp\left\{\sum_{k=1}^{K} \boldsymbol{\theta}_k f_k(y_t^{(i)}, y_{t-1}^{(i)}, \mathbf{x}_t^{(i)})\right\}\right] \\
&= \log \left[\prod_{i=1}^{N} \prod_{t=1}^{T} \exp\left\{\sum_{k=1}^{K} \boldsymbol{\theta}_k f_k(y_t^{(i)}, y_{t-1}^{(i)}, \mathbf{x}_t^{(i)})\right\}\right] + \log \left[\prod_{i=1}^{N} \frac{1}{Z(\mathbf{x}^{(i)})}\right] \\
&= \sum_{i=1}^{N} \sum_{t=1}^{T} \sum_{k=1}^{K} \boldsymbol{\theta}_k f_k(y_t^{(i)}, y_{t-1}^{(i)}, \mathbf{x}_t^{(i)}) - \sum_{i=1}^{N} Z(\mathbf{x}^{(i)})
\end{aligned}$$

We can optimize this likelihood with gradient descent, and the gradients in the parameters involve computing the factors, meaning we can use probabilistic inference methods to compute them efficiently. Training a linear-chain CRF then comes down to two things:

- Inference of $p(y_t \mid \mathbf{x})$, $p(y_t, y_{t-1} \mid \mathbf{x})$ and $Z(\mathbf{x})$ (subroutines of parameter estimation).
- Decoding: $\mathbf{y}^* = arg\max_{\mathbf{y}} p(\mathbf{y} \mid \mathbf{x})$

Inference in this model can be done with the forward-backward (belief propagation) algorithm, and decoding can be done with the Viterbi algorithm.

2.4.1 Forward-Backward Algorithm for a Linear-Chain CRF

The forward-backward algorithm defines an efficient way of calculating marginals from joints ($p(x) = \sum_y p(x, y)$)[1].

Finish from http://www.cs.columbia.edu/~mcollins/fb.pdf

2.4.2 Viterbi for a Linear-Chain CRF

2.5 General CRF

The general definition of a CRF:
Let G be a factor graph over $X$and Y, then (X, Y) is a conditional random field if for any value $\mathbf{x} \in X$, the distribution $p(\mathbf{y} \mid \mathbf{x})$ factorizes according to G.
The conditional distribution for a CRF is then:

$$p(\mathbf{y} \mid \mathbf{x}) = \frac{1}{Z(\mathbf{x})} \prod_{a=1}^{A} \psi_a(\mathbf{y}_a, \mathbf{x}_a)$$

Where the only difference with a general factor graph is the dependence of the partition function on the input $\mathbf{x}$. If we take log-linear factors (meaning that $\log(\psi_a(\mathbf{y}_a, \mathbf{x}_a))$ will be linear over a set of feature functions), the CRF becomes:

$$p(\mathbf{y} \mid \mathbf{x}) = \frac{1}{Z(\mathbf{x})} \prod_{\psi_A \in \mathcal{F}} \exp\left\{ \sum_{k=1}^{K(A)} \boldsymbol{\theta}_{a,k} f_{a,k}(\mathbf{x}_a, \mathbf{y}_a) \right\}$$

2.6 Training a General CRF

The methods used to train a linear-chain CRF also apply to a general CRF. We can train it with maximum likelihood, and in order to do so we need the partition function, the marginals and an efficient way of decoding. There are many algorithms for efficient inference in graphical models, like the junction tree algorithm for exact inference and Monte Carlo methods or Variational Inference for approximate inference. In the case where we are dealing with complete graphs, we can use Tutte's *Matrix Tree Theorem* (MTT) to evaluate the marginals and partition function efficiently. The decoding problem can be solved with different algorithms based on the specification of the CRF.
The log likelihood for our general CRF is almost identical to the linear case, so we won't derive it again:

$$\begin{aligned} \log \mathcal{L}(\boldsymbol{\theta}) &= \sum_{i=1}^{N} \log p(\mathbf{y}^{(i)} \mid \mathbf{x}^{(i)}) \\ &= \sum_{i=1}^{N} \sum_{\psi_a \in \mathcal{F}} \sum_{k=1}^{K(A)} \boldsymbol{\theta}_{a,k} f_{a,k}(\mathbf{x}_a^{(i)}, \mathbf{y}_a^{(i)}) - \sum_{i=1}^{N} \log Z(\mathbf{x}^{(i)}) \end{aligned}$$

Calculating the loss function involves evaluating the partition function, which can be done efficiently with the MTT as described in the next section. The partial derivative w.r.t. the parameters is[1]:

$$\frac{\partial \mathcal{L}(\boldsymbol{\theta})}{\partial \theta_{p,t}} = \sum_{i=1}^{N} \sum_{\psi_a \in \mathcal{F}} f_{p,t}(\mathbf{x}_a^{(i)}, \mathbf{y}_a^{(i)}) - \sum_{i=1}^{N} \sum_{\psi_a \in \mathcal{F}} \sum_{\mathbf{y}' \in Y} f_{p,t}(\mathbf{x}_a^{(i)}, \mathbf{y}_a'^{(i)}) p(\mathbf{y}' \mid \mathbf{x}^{(i)})$$

[1] Taken from http://www.cs.columbia.edu/~mcollins/fb.pdf

Which involves the marginals $\sum_{\mathbf{y}' \in Y} p(\mathbf{y}' \mid \mathbf{x}^{(i)})$ that can be computed with the MTT for every data point.

2.6.1 The Matrix Tree Theorem

Kirchoff's matrix tree theorem is a theorem from linear algebra that defines an efficient way to count spanning trees in a graph, and Tutte extended it to counting directed trees which is exactly what we need to calculate the partition function[2].

3 Part-of-Speech Tagging with a Linear-Chain CRF

[2]

4 Dependency Parsing with a General CRF

[3]

[2]Taken from https://personalpages.manchester.ac.uk/staff/mark.muldoon/Teaching/DiscreteMaths/LectureNotes/IntroToMatrixTree.pdf

References

[1] Terry Koo, Amir Globerson, Xavier Carreras, and Michael Collins. Structured prediction models via the matrix-tree theorem. In *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL)*, 2007.

[2] Xuezhe Ma and Eduard H. Hovy. End-to-end sequence labeling via bi-directional lstm-cnns-crf. *CoRR*, abs/1603.01354, 2016.

[3] Xuezhe Ma and Eduard H. Hovy. Neural probabilistic model for non-projective MST parsing. *CoRR*, abs/1701.00874, 2017.

[4] Charles Sutton and Andrew McCallum. An introduction to conditional random fields. *Found. Trends Mach. Learn.*, 4(4):267–373, April 2012.