CS769 Spring 2010 Advanced Natural Language Processing

Conditional Random Fields

Lecturer: Xiaojin Zhu *jerryzhu@cs.wisc.edu*

1 Information Extraction

Current NLP techniques cannot yet truly understand natural language articles. However, they can be useful on simpler tasks. One such task is Information Extraction. For example, one might want to extract the title, authors, year, and conference names from a researcher's Web page. Or one might want to identify person, location, organization names from news articles (NER, named entity recognition). The unstructured Web pages can then be converted into structured knowledge databases, forming the basis of many Web services. The basic Information Extraction technique is to treat the problem as a sequence labeling problem. The label set can be {title, author, year, conference, other}, or {person, location, organization, other}, for the examples respectively.

Being a sequence labeling tool, HMMs has been successfully applied to Information Extraction. However, HMMs have difficulty modeling *overlapping, non-independent* features. For example, an HMM might specify which word x is likely for a given label z (i.e., tag or state) via $p(x|z)$. But often the part-of-speech of the word, as well as that of the surrounding words, character n-grams, capitalization patterns all carry important information. HMMs cannot easily model these, because the generative story limits what can be generated by a state variable.

Conditional Random Fields (CRF) are discriminative graphical models that can model these overlapping, non-independent features. A special case, linear-chain CRF, can be thought of as the *undirected graphical model* version of HMM. It is as efficient as HMMs, where the sum-product algorithm and max-product algorithm still apply. Along a different dimension, HMMs are the sequence version of Naive Bayes models, while linear-chain CRFs are the sequence version of logistic regression.

2 The CRF Model

Let $x_{1:N}$ be the observations (e.g., words in a document), and $z_{1:N}$ the hidden labels (e.g., tags). A linear chain Conditional Random Field defines a *conditional probability* (whereas HMM defines the joint)

$$p(z_{1:N}|x_{1:N}) = \frac{1}{Z} \exp\left(\sum_{n=1}^{N}\sum_{i=1}^{F} \lambda_i f_i(z_{n-1}, z_n, x_{1:N}, n)\right). \tag{1}$$

Let us walk through the model in detail. The scalar Z is a normalization factor, or *partition function*, to make $p(z_{1:N}|x_{1:N})$ a valid probability over label sequences. Z is defined as

$$Z = \sum_{z_{1:N}} \exp\left(\sum_{n=1}^{N}\sum_{i=1}^{F} \lambda_i f_i(z_{n-1}, z_n, x_{1:N}, n)\right), \tag{2}$$

which has an exponential number of terms, and is difficult to compute in general. Note Z implicitly depends on $x_{1:N}$ and the parameters λ.

Within the exp() function, we sum over $n = 1, \ldots, N$ word positions in the sequence. For each position, we sum over $i = 1, \ldots, F$ *weighted features*. The scalar λ_i is the weight for feature $f_i()$. The λ_i's are the *parameters* of the CRF model, and must be learned, similar to $\theta = \{\pi, \phi, A\}$ in HMMs.

3 Feature Functions

The feature functions are the key components of CRF. In our special case of linear-chain CRF, the general form of a feature function is $f_i(z_{n-1}, z_n, x_{1:N}, n)$, which looks at a pair of adjacent states z_{n-1}, z_n, the *whole* input sequence $x_{1:N}$, and where we are in the sequence. The feature functions produce a real value.

For example, we can define a simple feature function which produces binary values: it is 1 if the current word is John, and if the current state z_n is PERSON:

$$f_1(z_{n-1}, z_n, x_{1:N}, n) = \begin{cases} 1 & \text{if } z_n = \text{PERSON and } x_n = \text{John} \\ 0 & \text{otherwise} \end{cases} \tag{3}$$

How is this feature used? It depends on its corresponding weight λ_1. If $\lambda_1 > 0$, whenever f_1 is active (i.e. we see the word John in the sentence and we assign it tag PERSON), it increases the probability of the tag sequence $z_{1:N}$. This is another way of saying "the CRF model should prefer the tag PERSON for the word John". If on the other hand $\lambda_1 < 0$, the CRF model will try to avoid the tag PERSON for John. And when $\lambda_1 = 0$, this feature has no effect whatsoever. Which way is correct? One may set λ_1 by domain knowledge (we know it should probably be positive), or learn λ_1 from corpus (let the data tell us), or both (treating domain knowledge as prior on λ_1). Note $\lambda_1, f_1()$ together play the same role as (the log of) HMM's ϕ parameter $p(x = \text{John}|z = \text{PERSON})$.

As another example, consider

$$f_2(z_{n-1}, z_n, x_{1:N}, n) = \begin{cases} 1 & \text{if } z_n = \text{PERSON and } x_{n+1} = \text{said} \\ 0 & \text{otherwise} \end{cases} \tag{4}$$

This feature is active if the current tag is PERSON and the *next* word is 'said'. One would therefore expect a positive λ_2 to go with the feature. Furthermore, note f_1 and f_2 can be both active for a sentence like "John said so." and $z_1 = \text{PERSON}$. This is an example of *overlapping features*. It boosts up the belief of $z_1 = \text{PERSON}$ to $\lambda_1 + \lambda_2$. This is something HMMs cannot do: HMMs cannot look at the next word, nor can they use overlapping features. On the other hand, a CRF feature can use any part of the whole sequence $x_{1:N}$.

The next feature example is rather like the transition matrix A in HMMs. We can define

$$f_3(z_{n-1}, z_n, x_{1:N}, n) = \begin{cases} 1 & \text{if } z_{n-1} = \text{OTHER and } z_n = \text{PERSON} \\ 0 & \text{otherwise} \end{cases} \tag{5}$$

This feature is active if we see the particular tag transition (OTHER, PERSON). Note it is the value of λ_3 that actually specifies the equivalent of (log) transition probability from OTHER to PERSON, or $A_{\text{OTHER, PERSON}}$ in HMM notation. In a similar fashion, we can define all K^2 transition features, where K is the size of tag set.

Of course the features are not limited to binary functions. Any real-valued function is allowed. Designing the features of an CRF is the most important task. In CRFs for real applications it is not uncommon to have tens of thousands or more features.

4 Undirected Graphical Models (Markov Random Fields)

CRF is a special case of undirected graphical models, also known as Markov Random Fields (MRFs) (their counterparts are directed graphical models, aka Bayes Networks). A *clique* is a subset of nodes in the graph that are fully connected (having an edge between any two nodes). A maximum clique is a clique that is not a subset of any other clique. Let X_c be the set of nodes involved in a maximum clique c. Let $\psi(X_c)$ be an arbitrary non-negative real-valued function, called the *potential function*. In particular $\psi(X_c)$ does not need to be normalized. The Markov Random Field defines a probability distribution over the node states as the normalized product of potential functions of all maximum cliques in the graph:

$$p(X) = \frac{1}{Z} \prod_c \psi(X_c), \tag{6}$$

where Z is the normalization factor. In the special case of linear-chain CRFs, the cliques correspond to a pair of states z_{n-1}, z_n as well as the corresponding x nodes, with

$$\psi = \exp(\lambda f). \tag{7}$$

An undirected graphical model can be easily converted to a factor graph representation. Each clique is represented by a factor node with the factor $\psi(X_c)$, and the factor node connects to every node in X_c. There is one addition special factor node which represents Z. A nice consequence is that the sum-product algorithm and max-sum algorithm immediately apply to MRFs in general, and linear-chain CRFs in particular. The factor corresponding to Z can be ignored during message passing.

5 CRF training

Training means finding the λ parameters in a CRF. For this we need fully labeled data sequences $\{(\mathbf{x}^{(1)}, \mathbf{z}^{(1)}), \ldots, (\mathbf{x}^{(m)}, \mathbf{z}^{(m)})\}$, where $\mathbf{x}^{(1)} = x^{(1)}_{1:N_1}$ the first observation sequence, and so on[1]. Since CRFs define the conditional probability $p(\mathbf{z}|\mathbf{x})$, the appropriate objective for parameter learning is to maximize the conditional likelihood of the training data

$$\sum_{j=1}^{m} \log p(\mathbf{z}^{(j)}|\mathbf{x}^{(j)}). \tag{8}$$

Often one can also put a Gaussian prior on the λ's to regularize the training (i.e., smoothing). If $\lambda \sim N(0, \sigma^2)$, the objective becomes

$$\sum_{j=1}^{m} \log p(\mathbf{z}^{(j)}|\mathbf{x}^{(j)}) - \sum_{i}^{F} \frac{\lambda_i^2}{2\sigma^2}. \tag{9}$$

The good news is that the objective is concave, so the λ's have a unique set of global optimal values. The bad news is that there is no closed-form solution.

The standard parameter learning approach is to compute the gradient of the objective function, and use the gradient in an optimization algorithm like L-BFGS. The gradient of the objective function is computed as follows:

$$\frac{\partial}{\partial \lambda_k} \sum_{j=1}^{m} \log p(\mathbf{z}^{(j)}|\mathbf{x}^{(j)}) - \sum_{i}^{F} \frac{\lambda_i^2}{2\sigma^2} \tag{10}$$

$$= \frac{\partial}{\partial \lambda_k} \sum_{j=1}^{m} \left(\sum_{n} \sum_{i} \lambda_i f_i(z^{(j)}_{n-1}, z^{(j)}_n, \mathbf{x}^{(j)}, n) - \log Z^{(j)} \right) - \sum_{i}^{F} \frac{\lambda_i^2}{2\sigma^2} \tag{11}$$

$$= \sum_{j=1}^{m} \sum_{n} f_k(z^{(j)}_{n-1}, z^{(j)}_n, \mathbf{x}^{(j)}, n) - \sum_{j=1}^{m} \sum_{n} E_{z'_{n-1}, z'_n}[f_k(z'_{n-1}, z'_n, \mathbf{x}^{(j)}, n)] - \frac{\lambda_k}{\sigma^2}, \tag{12}$$

[1] Unlike HMMs which can use the Baum-Welch (EM) algorithm to train on unlabeled data $\mathbf{x}$ only, CRFs training on fully unlabeled sequence is difficult.

where we used the fact

$$\frac{\partial}{\partial \lambda_k} \log Z = E_{\mathbf{z}'}[\sum_n f_k(z'_{n-1}, z'_n, \mathbf{x}, n)] \quad (13)$$

$$= \sum_n E_{z'_{n-1}, z'_n}[f_k(z'_{n-1}, z'_n, \mathbf{x}, n)] \quad (14)$$

$$= \sum_n \sum_{z'_{n-1}, z'_n} p(z'_{n-1}, z'_n|\mathbf{x}) f_k(z'_{n-1}, z'_n, \mathbf{x}, n). \quad (15)$$

Note the edge marginal probability $p(z'_{n-1}, z'_n|\mathbf{x})$ is under the current parameters, and this is exactly what the sum-product algorithm can compute. The partial derivative in (12) has an intuitive explanation. Let us ignore the term λ_k/σ^2 from the prior. The derivative has the form of

$$\text{(observed counts of } f_k) - \text{(expected counts of } f_k).$$

When the two are the same, the derivative is zero. Therefore we see that training can be thought of as finding λ's that match the two counts.

6 Feature Selection

A common practice in NLP is to define a very large number of candidate features, and let the training data select a small subset to use in the final CRF model, in a process known as *feature selection*:

1. Initially the CRF model has no features, and thus makes uniform predictions. Let $M = \emptyset$ be the set of features in the CRF.

2. Let the candidate feature set $C =$"Atomic features". These are usually predicates on simple combination of words and tags, e.g.($x =$John, $z =$PERSON), ($x =$John, $z =$LOCATION), ($x =$John, $z =$ORGANIZATION), etc. There are VK such "word identity" candidate features, which is obviously a large number. It should be noted that these atomic predicates *involve the tag*. Similarly one can test whether the word is capitalized, the identity of the neighboring words, the part-of-speech of the word, and so on. The state transition features are also atomic.

3. Build an individual CRF with features $M \cup \{f\}$ for each candidate feature $f \in C$. Select the candidate feature f^* which improve the CRF model the most (e.g., by the increase in tuning set likelihood). Let $M = M \cup \{f^*\}$, and $C = C - \{f^*\}$.

4. "Growing" candidate features. It is often natural to combine simple features to form more complex features. For example, one can test for current word being capitalized, the next word being "Inc.", and both tags being ORGANIZATION. However, the number of complex features can grow quickly. A compromise is to only grow candidate features on selected features so far, by extending them with one atomic additions. That is, $C = C \cup \{f^* \bigodot f | f \in \text{atomic}\}$ where $\bigodot$ denotes some simple Boolean operations like AND, OR.

5. Go to step 3 until enough features have been added to the CRF model, or its tuning set likelihood peaks.