10-708: Probabilistic Graphical Models 10-708, Spring 2014

12 : Conditional Random Fields

Lecturer: Eric P. Xing *Scribes: Qin Gao, Siheng Chen*

1 Hidden Markov Model

1.1 General parametric form

In hidden Markov model (HMM), we have three sets of parameters,

$$\begin{aligned}&\text{transition probability matrix A}: p(y_t^j = 1|y_{t-1}^i = 1) = a_{i,j},\\ &\text{initialprobabilities}: p(y_1) \sim \text{Multinomial}(\pi_1, \pi_2, ..., \pi_M),\\ &\text{emission probabilities}: p(x_t|y_t^i) \sim \text{Multinomial}(b_{i,1}, b_{i,2}, ..., b_{i,K}).\end{aligned}$$

1.2 Inference

The inference can be done with forward algorithm which computes $\alpha_t^k \equiv \mu_{t-1\to t}(k) = P(x_1, ..., x_{t-1}, x_t, y_t^k = 1)$ recursively by

$$\alpha_t^k = p(x_t|y_t^k = 1) \sum_i \alpha_{t-1}^i a_{i,k}, \tag{1}$$

and the backward algorithm which computes $\beta_t^k \equiv \mu_{t\leftarrow t+1}(k) = P(x_{t+1}, ..., x_T|y_t^k = 1)$ recursively by

$$\beta_t^k = \sum_i a_{k,i} p(x_{t+1}|y_{t+1}^i = 1)\beta_{t+1}^i. \tag{2}$$

Another key quantity is the conditional probability of any hidden state given the entire sequence, which can be computed by the dot product of forward message and backward message by,

$$\gamma_t^i = p(y_t^i = 1|x_{1:T}) \propto \alpha_t^i \beta_t^i = \sum_j \xi_t^{i,j}, \tag{3}$$

where we define,

$$\begin{aligned}\xi_t^{i,j} &= p(y_t^i = 1, y_{t-1}^j = 1, x_{1:T}),\\ &\propto \mu_{t-1\to t}(y_t^i = 1)\mu_{t\leftarrow t+1}(y_{t+1}^j = 1)p(x_{t+1}|y_{t+1})p(y_{t+1}|y_t),\\ &= \alpha_t^i \beta_{t+1}^j a_{i,j} p(x_{t+1}|y_{t+1}^i = 1).\end{aligned}$$

The implementation in Matlab can be vectorized by using,

$$\begin{aligned}&B_t(i) = p(x_t|y_t^i = 1),\\ &A(i,j) = p(y_{t+1}^j = 1|y_t^i = 1).\end{aligned}$$

The relation of those quantities can be simply written in pseudocode as,

$$\begin{aligned}
\alpha_t &= (A^T \alpha_{t-1}) .* B_t, \\
\beta_t &= A(\beta_{t+1} .* B_{t+1}), \\
\xi_t &= (\alpha_t (\beta_{t+1} .* B_{t+1})^T) .* A, \\
\gamma_t &= \alpha_t .* \beta_t .
\end{aligned}$$

1.3 Learning

1.3.1 Supervised Learning

The supervised learning is trivial if only we know the true state path. Based on maximal likelihood estimation, we just need to count the instances to find the transition and emission probability,

$$\begin{aligned}
a_{ij}^{ML} &= \frac{\#(i \to j)}{\#(i \to \cdot)} = \frac{\sum_n \sum_{t=2}^T y_{n,t-1}^i y_{n,t}^j}{\sum_n \sum_{t=2}^T y_{n,t-1}^i} \\
b_{ik}^{ML} &= \frac{\#(i \to k)}{\#(i \to \cdot)} = \frac{\sum_n \sum_{t=1}^T y_{n,t}^i x_{n,t}^k}{\sum_n \sum_{t=1}^T y_{n,t}^i}.
\end{aligned} \tag{4}$$

Some pseudocounting technique can avoid zero probability. In the continuous case, it is no longer multinomial distribution but rather condition specific Gaussian distribution or something else. We can perform the usual methods by collecting samples from the same Gaussian distribution and calculating their mean and covariance.

1.3.2 Unsupervised Learning

When the hidden states are unobserved we use the Baum Welch algorithm where we deal with complete likelihood with expectation over the hidden variables. The likelihood is then decomposable. The counts of the hidden events is replaced by the expectation of the hidden events. Specifically, the likelihood can be written as,

$$l_c(\theta; x, y) = \log p(x, y) = \log \prod_n (p(y_{n,1}) \prod_{t=2}^T p(y_{n,t}|y_{n,t-1}) \prod_{t=1}^T p(x_{n,t}|y_{n,t})). \tag{5}$$

The expected complete log likelihood is,

$$\langle l_c(\theta; x, y) \rangle = \sum_n (\langle y_{n,1}^i \rangle_{p(y_{n,1}|x_n)} \log \pi_i) + \sum_n \sum_{t=2}^T (\langle y_{n,t-1}^i y_{n,t}^j \rangle_{p(y_{n,t-1}, y_{n,t}|x_n)} \log a_{i,j}) + \sum_n \sum_{t=1}^T (x_{n,t}^k \langle y_{n,t}^i \rangle_{p(y_{n,t}|x_n)} \log b_{i,k}) \tag{6}$$

We then can perform the E-step with,

$$\begin{aligned}
\gamma_{n,t}^i &= \langle y_{n,t}^i \rangle = p(y_{n,t}^i = 1|x_n) \\
\xi_{n,t}^{i,j} &= \langle y_{n,t-1}^i y_{n,t}^j \rangle = p(y_{n,t-1}^i = 1, y_{n,t}^j = 1|x_n).
\end{aligned} \tag{7}$$

The M-step is,

$$\pi_i = \frac{\sum_n \gamma_{n,1}^i}{N}, \quad a_{i,j} = \frac{\sum_n \sum_{t=2}^T \xi_{n,t}^{i,j}}{\sum_n \sum_{t=1}^{T-1} \gamma_{n,t}^i}, \quad b_{ik} = \frac{\sum_N \sum_{t=1}^T \gamma_{n,t}^i x_{n,t}^k}{\sum_n \sum_{t=1}^{T-1} \gamma_{n,t}^i} \tag{8}$$

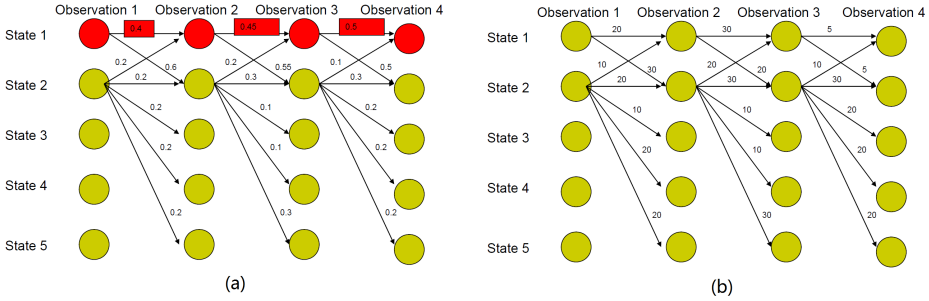

Figure 1: MEMM.

1.3.3 Shortcomings

One of the major shortcomings of HMM is that every state just generates a localized observation. For example, in a sentence segmentation task, you can only generate a single word from a given state. One observation thus depends on only one state directly and others indirectly. However, the observations in reality do not only depend on local features. How to capture the non-local features explicitly rather than implicitly?

The other issue is the mismatch between learning objective function and prediction objective function. HMM learns a joint distribution of states and observations $P(Y, X)$, but in a prediction task, we need the conditional probability $P(Y|X)$. We could directly write down $P(Y|X)$ in someway but it is not allowed in HMM.

2 Maximum Entropy Markov Model

2.1 Formalism

The first innovation over HMM is to include the global features. The maximum entropy is a way to design a loss function which we ignore the detail here. In an earlier lecture, we show that the generalized log linear model is equivalent to the maximum entropy. By design, we can write down the conditional probability as product of local transitional probabilities as,

$$P(y_{1:n}|x_{1:n}) = \prod_{i=1}^{n} P(y_i|y_{i-1}, x_{1:n}) = \prod_{i=1}^{n} \frac{\exp(w^T f(y_i, y_{i-1}, x_{1:n})}{Z(y_{i-1}, x_{1:n})}, \tag{9}$$

where Z is the normalization factor of the probability. It thus overcomes the two shortcomings of HMM, namely, including nonlocal features and directly computing the conditional probabilities.

2.2 Shortcomings

The performance of this model is not dominating HMM. People figured out the reason for this. In this model, it is shown that the word 'Rob' is more favorable than 'Rib' due to a statistical reason that is beyond our control. It is an intrinsic flaw in the model design which is called label bias and described in the example below.

In the example shown in Fig. 1(a), state 1 can only go to state 1 and 2 while state 2 can go to all the 5 states. From observation, we find that state 1 almost always prefers to go to state 2, while state 2 almost always prefer to stay in state 2. In MEMM, we find that the path $1 \to 1 \to 1 \to 1$ has probability 0.09, the path $2 \to 2 \to 2 \to 2$ has probability 0.018 and the path $1 \to 2 \to 1 \to 2$ has probability 0.06. In all these paths, the most likely path is $1 \to 1 \to 1 \to 1$, which is different from our observation that state 1 wants to go to state 2 and state 2 wants to remain in state 2.

The problem is that state 1 has only 2 transition events, however, state 2 has 5 transition states. Therefore, even though the 2 to 2 movement has high probability locally, but if you compare paths with 2 and path $1 \to 1 \to 1 \to 1$ they still have smaller probability. Basically we are in an unfair game to compare the probabilities with state 1 only and with 2. Thus, the reason is that the local information is improperly put into the global contest. Similarly, if 'i' has fewer ways to transit out, even 'R' to 'o' has large probability, but 'o' to anything also has a lot of different options. Even if 'o' to 'b' has relative high probability the magnitude might not be as high as 'i' to 'b'. If you are in 'i' already no matter what it transits to the whole path has a large probability. That's an artifact. What you believe is that 'o' to 'b' is more likely than 'i' to 'b' but they are calibrated in different ways in the model.

3 Conditional Random Fields

One solution to the problem is not normalizing the states locally but just put weights on transitions shown in Fig. 1(b). This is the basis of the early design of random Markov field that we use potentials instead of local probabilities to capture the local properties.

3.1 Generic form

The potential is a pre-probabilistic notion which does not need to be normalized. Mathematically, the probability can be written as

$$\begin{aligned} P(Y|X) &= \frac{1}{Z(X)} \prod_{i=1}^{n} \phi(y_i, y_{i-1}, X) \\ &= \frac{1}{Z(X,\lambda,\mu)} \exp(\sum_{i=1}^{n} (\sum_{k} \lambda_k f_k(y_i, y_{i-1}, X) + \sum_{l} \mu_l g_l(y_i, X))), \end{aligned}$$

where $Z(X,\lambda,\mu) = \sum_y \exp(\sum_{i=1}^{n} (\sum_k \lambda_k f_k(y_i, y_{i-1}, X) + \sum_l \mu_l g_l(y_i, X)))$. We see that it is a global normalization rather than a local normalization. The structure of CRF sees Figure 2. This method thus has the true local property, which is the key of graphic model, but still preserves the global effect that you desired. CRF overcomes the label bias problem of MEMM by using a global normalizer $Z(X)$.

3.2 Inference

The inference task is that given CRF parameter λ and μ, we want to find y^* that maximize $P(y|x)$, that is:

$$y^* = \arg\max_y \exp(\sum_{i=1}^{n} (\sum_{k} \lambda_k f_k(y_i, y_{i-1}, X) + \sum_{l} \mu_l g_l(y_i, X))).$$

Note that we ignore the normalization term $Z(x)$ because it is not a function of y. To solve the optimization problem, we run the max-product algorithm on the junction tree of CRF, which turns out to be the same as Viterbi decoding used in hidden Markov models.

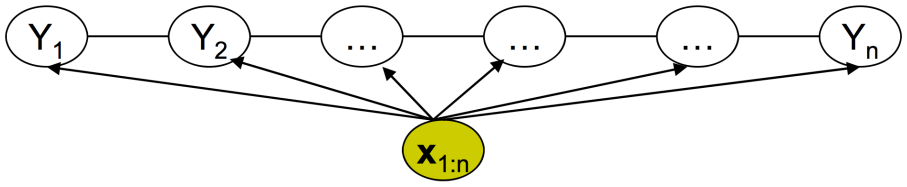

Figure 2: Conditional random field.

3.3 Learning

Although the whole graph is fully observed, the learning procedure of CRF is still tricky. The reason is that the learning procedure involves the inference subroutines. The details are as follows. Given the training set $\{x_d, y_d\}_{d=1}^N$, the goal is to find the optimal parameters λ^*, μ^*, such that

$$\begin{aligned}
\lambda^*, \mu^* &= \arg\max_{\lambda,\mu} L(\lambda, \mu) \\
&= \arg\max_{\lambda,\mu} \prod P(y_d|x_d, \lambda, \mu) \\
&= \arg\max_{\lambda,\mu} \frac{1}{Z(x_d, \lambda, \mu)} \exp(\sum_{i=1}^{n} (\lambda^T f(y_{d,i}, y_{d,i-1}, x_d) + \mu^T g(y_{d,i}, x_d)) \\
&= \arg\max_{\lambda,\mu} \sum_{d=1}^{N} (\sum_{i=1}^{n} (\lambda^T f(y_{d,i}, y_{d,i-1}, x_d) + \mu^T g(y_{d,i}, x_d)) - \log Z(x_d, \lambda, \mu))
\end{aligned}$$

The third equation follows by the definition of the conditional probability and the fourth equation follows by taking log. We then compute the gradient with respect to λ as follows:

$$\nabla_\lambda L(\lambda, \mu) = \sum_{d=1}^{N} (\sum_{i=1}^{n} f(y_{d,i}, y_{d,i-1}, x_d) - \sum_y (P(y|x_d) \sum_{i=1}^{n} f(y_{d,i}, y_{d,i-1}, x_d))). \quad (10)$$

We see that the first term is really a feature and the second term is the expectation of that feature, which follows by the fact that the gradient of the log-partition function in an exponential family is the expectation of the sufficient statistics.

It seems that to compute the model expectations requires exponentially large number of summations. We then use the message passing algorithm to compute the pairwise potentials and get a closed form pairwise conditional probability as did in inference; that is:

$$\begin{aligned}
\sum_y (P(y|x_d) \sum_{i=1}^{n} f(y_i, y_{i-1}, x_d)) &= \sum_{i=1}^{n} (\sum_y f(y_i, y_{i-1}, x_d) P(y|x_d)) \\
&= \sum_{i=1}^{n} (\sum_{y_i, y_{i-1}} f(y_i, y_{i-1}, x_d) P(y_i, y_{i-1}|x_d))
\end{aligned}$$

It implies that learning involves inference subroutines. By the message passing algorithm, the learning procedure can be done in a polynomial time.

We now compute feature expectations using calibrated potentials:

$$\nabla_\lambda L(\lambda, \mu) = \sum_{d=1}^{N} (\sum_{i=1}^{n} f(y_{d,i}, y_{d,i-1}, x_d) - \sum_y (P(y|x_d) f(y_{d,i}, y_{d,i-1}, x_d)))$$

$$= \sum_{d=1}^{N}(\sum_{i=1}^{n} f(y_{d,i}, y_{d,i-1}, x_d) - \sum_{y_i, y_{i-1}} \alpha'(y_i, y_{i-1}) f(y_{d,i}, y_{d,i-1}, x_d))),$$

where $\alpha'(y_i, y_{i-1}, x_d) = P(y_i, y_{i-1}|x_d)$. We then learn parameters by using gradient ascent as follows:

$$\begin{aligned} \lambda^{(t+1)} &= \lambda^{(t)} + \eta \nabla_\lambda L(\lambda^{(t)}, \mu^{(t)}) \\ \mu^{(t+1)} &= \mu^{(t)} + \eta \nabla_\mu L(\lambda^{(t)}, \mu^{(t)}). \end{aligned}$$

In practice, a Gaussian regularizer is used for the parameter vector to improve generalizability

$$\lambda^*, \mu^* = \arg\max_{\lambda,\mu} \sum_{d=1}^{N} \log P(y_d|x_d, \lambda, \mu) - \frac{1}{2\sigma^2}(\lambda^T\lambda + \mu^T\mu).$$

The second term is Gaussian prior, because we want to push λ^*, μ^* to be zeros, which leads to fewer feature. It could be Laplacian prior as well. In conditional probability, however, we do not want to see zeros because zero probability is ill-posed. Gradient ascend converges slowly, faster alternatives could be conjugate gradient method or Quasi-Newton methods.

From empirical performances, CRF gets only a little improvement over HMM and MEMM, especially when the nonlocal effect is obvious. Although the improvement is not drastic, it opens a new paradigm for wide range of problems. Another advantage of CRF is that it has large flexibility for users to design arbitrary features.

3.4 Discussion

The previous sections describe the model of CRFs and the way to do inference and learning. We discuss several interesting topics related to CRFs.

3.4.1 When CRFs meet computer vision

Computer vision community tends to emphasize feature design and empirical performances; on the other hand, machine learning community tends to build models and be away from real data and real applications. Because of the different goals, there is little interaction between two communities for a long time. CRF, however, is an unique successful product by machine learning that has been practiced by computer vision community because it has a perfect interface of both modeling and feature design.

3.4.2 Model evaluation

We can extend CRF from a chain to arbitrary graph structures. For example, in grid CRFs, the exact inference is intractable when the graph is huge, so we need to use approximate techniques. There are various inference algorithm to choose, such as MCMC sampling variational inference, and loopy belief propagation. Each algorithm has its own advantage. It is sometimes ambiguous to evaluate a method. The reason is that the inference algorithm that has been chosen influences the performance of the model. One solution is that, given a model, we choose a bunch of inference algorithms and collect the performance. If even exact inference does not perform well, it then means that the model is not correct. On the other hand, if a good inference algorithm leads to a great performance, it then means that the model is correct but the inference algorithm is terrible. For instance, people may get bad results by applying the mean field algorithm on grid CRFs and argue the model of CRF is bad. The truth is, however, that the mean field algorithm works bad in grid case because of the assumption of independence. It is worth to noticing there is an interplay between model and algorithm.

3.4.3 Other CRFs

We compare CRF with two other models, Markov random field (MRF) and discriminative random field (DRF). Note that all of them are actually based on the model of Markov random field and MRF here is a specific way to implement that model.

MRF is a generative model by assuming that $P(y_i|y_{S-\{i\}}) = P(y_i|y_{N_i})$ for all $i \in S$. The model is as follows:

$$P(X,Y) \;=\; P(X|Y)P(Y) = \prod_{i\in S} P(x_i|y_i) \cdot \frac{1}{Z} \prod_{c\in C} \Psi(y_c). \tag{11}$$

We see that MRF is implemented in two steps: it first constructs the likelihood $P(X|Y)$ and the prior $P(Y)$; in the second step, it uses Bayes theorem to determine posterior $P(Y|X)$. As a generative model, MRF handles the discriminative tasks by using Bayes theorem to determine the posterior as follows:

$$P(Y|X) \;=\; \frac{P(X,Y)}{P(X)} \propto \; P(X,Y) \;=\; P(X|Y)P(Y). \tag{12}$$

One popular formulation of MRF is as follows:

$$P(Y|X) = \frac{1}{Z} \exp(\sum_{i\in S} \log p(x_i|y_i) + \sum_{i\in S_i} \sum_{i'\in N_i} V_2(y_i, y_{i'})). \tag{13}$$

CRF is one-step discriminative model, which directly infers posterior $P(Y|X)$. It assumes that $P(y_i|X, y_{S-\{i\}}) = P(y_i|X, y_{N_i})$. The model is as follows:

$$P(Y|X) \;=\; \frac{1}{Z(X)} \prod_{i=1}^{n} \phi(y_i, y_{i-1}, X). \tag{14}$$

One popular formulation of CRF is as follows:

$$P(Y|X) = \frac{1}{Z} \exp(-\sum_{i\in S} V_1(y_i|X) + \sum_{i\in S_i} \sum_{j\in N_i} V_2(y_i, y_j|X)). \tag{15}$$

The biggest difference between CRF and MRF is the way to normalize the joint probability distribution. Since they formulate in two different ways, MRF focuses on local features while CRF has more flexibility to design features.

DRF is a special type of CRF, which has a specific way to design feature train feautre in the preprocessing step. The model is as follows:

$$P(Y|X) = \frac{1}{Z} \exp(\sum_{i\in S} A_i(y_i, X) + \sum_{i\in S_i} \sum_{j\in N_i} I_{i,j}(y_i, y_j|X)). \tag{16}$$

The first term is called association potential and the second term is called interaction potential. Association potential is a local discriminative model for site i, which uses logistic link. The form is as:

$$A_i(y_i, X) = \log P(y_i|f_i(X)), \text{ where } P(y_i = 1|f_i(X)) = \log \frac{1}{1 + \exp(-w^T f_i(X))}. \tag{17}$$

Interaction potential measures how likely site i and j have the same label given all data X. The form is as:

$$I_{i,j}(y_i, y_j, X) = k y_i y_j + (1-k)(2\sigma(y_i y_j \mu_{ij}(X) - 1)). \tag{18}$$

The first term is a data-independent smoothing prior. It means that before seeing anything, we believe that labels that are similar to each other should take some weights. The second term is data-dependent pairwise logistic function.

3.4.4 Applications

CRFs have a wide range of applications in computer vision, including image segmentation [1], object detection [2], and body pose estimation [3, 4]. For instance, image segmentation is a pixel labeling problem to identify each pixel belongs to background or foreground. It is equivalent to maximize the conditional probability $Y^* = \arg\min_Y P(Y|X)$. We then can use either MRF or DRF to solve the problem. From the empirical results, we conclude that DRF did better job because MRF lacks of neighborhood interaction of data, which matches the properties of each model.

4 Reference

[1] S. Nowozin and C. H. Lampert, "Structured Prediction and Learning in Computer Vision", Foundations and Trends in Computer Graphics and Vision, Volume 6, Number 3-4.

[2] S. Kumar and M. Hebert, "Discriminative random field", IJCV, 2006.

[3] V. Ferrari, M. Marin-Jimenez, A. Zisserman, "Progressive search space reduction for human pose estimation", CVPR, 2008.

[4] D. Ramanan, "Learning to Parse Images of Articulated Bodies", NIPS, 2006.