JMLR: Workshop and Conference Proceedings 21:259–263, 2012 The 11th ICGI

Marginalizing Out Transition Probabilities for Several Subclasses of PFAs

Chihiro Shibata SHIBATACHH@STF.TEU.AC.JP
School of Computer Science, Tokyo University of Technology

Ryo Yoshinaka RY@I.KYOTO-U.AC.JP
Graduate School of Informatics, Kyoto University

Editors: Editor's Name

Abstract

A Bayesian manner which marginalizes transition probabilities can be generally applied to various kinds of probabilistic finite state machine models. Based on such a Bayesian manner, we implemented and compared three algorithms: variable-length gram, state merging method for PDFAs, and collapsed Gibbs sampling for PFAs. Among those, collapsed Gibbs sampling for PFAs performed the best on the data from the pre-competition stage of PAutomaC, although it consumes large computation resources.

Keywords: PFAs, collapsed Gibbs sampling, ALERGIA, variable-length grams

1. Introduction

The data used in PAutomaC were generated by various kinds of probabilistic finite state machines such as HMMs, PDFAs, Markov chains, all of which can be seen as special cases of PFAs. We implemented and compared methods which marginalize out state transition probabilities to infer such machines.

Several powerful inference methods for HMMs have been proposed and used for many applications. Since it is difficult to marginalize out state transition probabilities and sum up them with respect to hidden variables at the same time, some approximation technique is required. *Collapsed Gibbs sampling (CGS)* is a Bayesian method which samples hidden variables after marginalizing out parameters, and is thought as one of the best choices for HMM inference (Gao and Johnson, 2008). We applied CGS to PFA inference.

For PDFAs, we implemented a state-merging algorithm based on such Bayesian manner. We also implemented an inference algorithm for a variable-length gram model, which can be regarded as a special type of PDFAs.

Based on the scores obtained by executing those methods on the data sets given in the pre-competition stage of PAutomaC, we conclude that CGS performs the best.

2. Preliminaries

We conveniently assume that the sentences in a sample is jointed into a single sentence separated by a terminal symbol.[1] Throughout the paper, A denotes the size of the alphabet

1. Although the terminal symbol is specially treated in the actual implementations, the details are largely ignored in this paper due to the space limit.

including the terminal symbol, N the number of states and T the length of the jointed sentence. Let $\mathbb{a} = a_1 \cdots a_T$ and $\mathbb{z} = z_1 \cdots z_{T+1}$ be generated sequences of symbols and states from a learning target PFA, respectively, where $\mathbb{z}$ is not observable. By ξ_{iaj} we denote the transition probability where a state i is followed by a symbol a and a state j. Note that $\sum_{a,j} \xi_{i,a,j} = 1$. Here, we define functions $\mathrm{C}(\cdot)$ as:

$$\mathrm{C}(i) = \sum_t \delta \begin{pmatrix} z_t \\ i \end{pmatrix}, \quad \mathrm{C}(i,a) = \sum_t \delta \begin{pmatrix} z_t, & a_t \\ i, & a \end{pmatrix}, \quad \mathrm{C}(i,a,j) = \sum_t \delta \begin{pmatrix} z_t, & a_t, & z_{t+1} \\ i, & a, & j \end{pmatrix},$$

where $\delta \left(\begin{smallmatrix} x_1,\cdots,x_n \\ y_1,\cdots,y_n \end{smallmatrix} \right)$ is the so-called delta function, which equals 1 if $x_k = y_k$ for all $k = 1, \ldots, n$ and equals 0 otherwise. That is, $\mathrm{C}(i)$ is the number of occurrences of state i in the transition history $\mathbb{z}$ and $\mathrm{C}(i,a,j)$ counts the number of transitions from i to j with a symbol a in $\mathbb{z}$.

The probability where $\mathbb{a}$ and $\mathbb{z}$ are generated is given by

$$\Pr(\mathbb{a}, \mathbb{z} \mid \xi) = \prod_{t=1,\cdots,T} \xi_{z_t a_t z_{t+1}} = \prod_{i,a,j} \xi_{iaj}^{\mathrm{C}(i,a,j)} \ . \tag{1}$$

As determining ξ uniquely so as to maximize (1) often causes the overfitting problem, ξ should be treated as some distribution. Let the prior $\Pr(\xi)$ be the product of Dirichlet distributions with a hyper parameter β, i.e., $c(\beta) \prod_{i,a,j} \xi_{iaj}^{\beta-1}$, where $c(\beta)$ is its normalizing constant and $0 < \beta \leq 1$. Such $\Pr(\xi)$ is called a *conjugate prior*. If $\beta = 1$, every ξ is equally likely before samples are given. By marginalizing out ξ, where $\sum_{a,j} \xi_{iaj} = 1$, we obtain

$$\Pr(\mathbb{a}, \mathbb{z}) = \int \Pr(\mathbb{a}, \mathbb{z} \mid \xi) \Pr(\xi) d\xi = \int \prod_{i,a,j} \xi_{iaj}^{\mathrm{C}(i,a,j)+\beta-1} c(\beta) d\xi = \prod_i \frac{\prod_{a,j} \Gamma(\mathrm{C}(i,a,j)+\beta)}{\Gamma(\mathrm{C}(i)+AN\beta)} c(\beta). \tag{2}$$

where $\Gamma(\cdot)$ is the gamma function.

3. Inference Methods

3.1. State-Merging Algorithms for PDFAs

ALERGIA (Carrasco and Oncina, 1994), which was a baseline algorithm of PAutomaC, constructs a PDFA starting from the probabilistic prefix tree acceptor (PPTA) for the training sample by merging states whose stochastic behaviors are similar. MDI (Thollard et al., 2000) is a modification of ALERGIA which merges states if it reduces the size of the automaton and the *Kullback-Leibler divergence (KLD)* with the initial PPTA is kept small. Our criterion for merging states is based on the marginal probability: We greedily merge states if it increases $\Pr(\mathbb{a})$. If the model is a PDFA, $\mathbb{z}$ is uniquely determined by $\mathbb{a}$. Hence, (2) is rewritten as

$$\Pr(\mathbb{a}) = \Pr(\mathbb{a}, \mathbb{z}) = \int \Pr(\mathbb{a}, \mathbb{z} \mid \xi) \Pr(\xi) d\xi = \prod_i \frac{\prod_a \Gamma(\mathrm{C}(i,a)+\beta)}{\Gamma(\mathrm{C}(i)+A\beta)} c(\beta) \ . \tag{3}$$

Both $\mathrm{C}(i)$ and $\mathrm{C}(i,a)$ are changed only when the state i is merged with other states. Thus it is enough to recalculate such local parts to update $\Pr(\mathbb{a})$ at each merging step.

3.2. Variable-Length N-Grams

Another baseline algorithm of PAutomaC was a 3-gram learner. A typical elaboration of an n-gram model is a variable-length gram model, which uses grams of different length. The literature has proposed a variety of criteria for determining the length of a gram to be used.

The criterion of our algorithm EVgram is based on marginal probabilities, which is just Eq. 3, for a variable-length gram model can be seen as a special case of a PDFA, where grams begin removed the last symbols correspond to states. We also implemented a variant of Niesler and Woodland's (1999) method, whose criterion is based on cross-validation. We call it CVgram and compared it with our method.

3.3. Collapsed Gibbs Sampling for PFAs

In the case of PFAs, we need to sum up $\Pr(\mathbb{a}, \mathbb{z})$ with respect to $\mathbb{z}$ to calculate $\Pr(\mathbb{a})$, for which, however, no feasible and rigorous technique is known. After marginalizing out ξ, we should either sample or use some approximation for $\mathbb{z}$. We focus on a sampling method in the following. Collapsed Gibbs sampling (CGS) is an algorithm that samples $\mathbb{z}$ in the following manner: Initially $\mathbb{z}$ is randomly chosen, and then z_t is sequentially and iteratively changed according to $\Pr(z_t = k \mid \mathbb{a}, \mathbb{z}^{\neg t})$, where $\mathbb{z}^{\neg t} = z_1 \cdots z_{t-1} z_{t+1} \cdots z_{T+1}$.

For PFAs, by using $\Gamma(x+1) = x\Gamma(x)$ for (2), $\Pr(z_t = k, \mathbb{z}^{\neg t}, \mathbb{a})$ is represented as

$$\Pr(z_t = k, \mathbb{z}^{\neg t}, \mathbb{a}) = g_k \prod_i \frac{\prod_{a,j} \Gamma(\mathrm{C}^{\neg t}(i,a,j) + \beta)}{\Gamma(\mathrm{C}^{\neg t}(i,a,j) + \beta)} \ ,$$

$$\text{where } g_k = \frac{\left(\mathrm{C}^{\neg t}(k, a_t, z_{t+1}) + \beta\right)\left(\mathrm{C}^{\neg t}(z_{t-1}, a_{t-1}, k) + \delta\left(\begin{smallmatrix} k, & a_t, & z_{t+1} \\ z_{t-1}, & a_{t-1}, & k \end{smallmatrix}\right) + \beta\right)}{\mathrm{C}^{\neg t}(k) + AN\beta} \ ,$$

$$\mathrm{C}^{\neg t}(i) = \mathrm{C}(i) - \delta\left(\begin{smallmatrix} z_t \\ i \end{smallmatrix}\right) \ , \quad \mathrm{C}^{\neg t}(i,a,j) = \mathrm{C}(i,a,j) - \delta\left(\begin{smallmatrix} z_t, & a_t, & z_{t+1} \\ i, & a, & j \end{smallmatrix}\right) - \delta\left(\begin{smallmatrix} z_{t-1}, & a_{t-1}, & z_t \\ i, & a, & j \end{smallmatrix}\right) \ .$$

Since $\Pr(z_t = k, \mathbb{z}^{\neg t}, \mathbb{a})/g_k$ does not depend on k, $\Pr(z_t = k \mid \mathbb{a}, \mathbb{z}^{\neg t})$ can be calculated as $g_k / \sum_i g_i$.

Calculation of g_k requires constant time. The time complexity of CGS for PFAs is $\mathcal{O}(NTL)$, where L is the number of iterations.

After $\mathbb{z}$ is sampled multiply, say $\mathbb{z}_1, \ldots, \mathbb{z}_S$, the predictive probability for an unknown sentence $\mathbb{b}$ is estimated as $\Pr(\mathbb{b} \mid \mathbb{a}) \approx \frac{1}{S} \sum_s \Pr(\mathbb{b} \mid \widetilde{\xi}^{(s)})$, where $\widetilde{\xi}^{(s)}_{iaj} = E\left[\xi_{iaj} \mid \mathbb{a}, \mathbb{z}_s\right] = (\mathrm{C}(i,a,j) + \beta)/(\mathrm{C}(i) + AN\beta)$. It is known that the distribution of $\mathbb{z}$ converges to $\Pr(\mathbb{z} \mid \mathbb{a})$ in the limit (see, e.g., Chap. 11 of (Bishop, 2006)), whereas greedy algorithms such as Baum-Welch and variational Bayesian methods may converge to a local optimal point.

4. Experiments and Results

We applied the methods proposed in Sec. 3 to the 51 problems of the pre-competition stage of PAutomaC and computed their scores in accordance with PAutomaC's official rule.

We set the hyper parameter $\beta = 0.5$ for the state-merging method (Sec. 3.1), EVgram and CVgram (Sec. 3.2). Though Niesler and Woodland (1999) proposed to adopt leaving-one-out cross-validation, CVgram employed 10-fold cross-validation, since scores with finer fragmentations were worse in preparatory experiments.

In CGS (Sec. 3.3), although the distribution of $\mathbb{z}$ should converge to $\Pr(\mathbb{z} \mid \mathbb{a})$ in the limit, in actual implementations it is hard to decide when the total amount of iterations should suffice. In our experiments, we sampled $\mathbb{z}$ for every 100 iterations between 10,100th iteration and 20,000th, hence we got $S = 100$ samples in total. Furthermore, we ran the algorithm 10 times independently. Therefore, our final answer to each problem was calculated as the average of the probabilities obtained from 1,000 samples. Fig. 1 shows how the scores of answers calculated from the latest 100 samples of 10 trials vary for different number of iterations up to 20,000. After 12,000 iterations, respective lines seem flat and close to each other. Hence 20,000 iterations seem enough. The black and bold line represents the score of the average answer of those 10 answers. Empirically, the score of averaged answers is generally better than the average of their scores.

We put before the actual training phase two preparatory phases where N and β are determined. As Fig. 2 shows, the best choice for N depends on problems. By 10-fold cross-validation, we set N to be the value among $\{10, 20, \ldots, 90\}$ that gives the largest probability, where we used $\beta = 0.5$. After determining N, we selected the best value amongst $\{0.01, 0.02, 0.05, 0.1, 0.2, 0.5\}$ for β again by 10-fold cross-validation. The effectiveness of this process is illustrated in Fig. 3, where the circles indicate the scores by the chosen values.

For respective problems in the pre-competition stage of PAutomaC, one iteration took about 0.2 to 2.0 seconds, thus 400 to 40,000 seconds for 20,000 iterations. To determine values of N and β among 9 and 6 candidates by 10-fold cross-validation, respectively, one must run CGS 150 times in total for every problem. With those determined values, we ran CGS 10 times to obtain a final answer. The computational cost of the entire process was quite high, so we ran CGS in parallel on a grid computing environment `InTrigger`[2] with a parallel computing processing system `GXP Make`[3].

Among algorithms presented in Sec. 3, CGS performed the best. The average ratio of its scores against the theoretical minimum scores exceeded 1 by 0.00129. CVgram and EVgram in Sec. 3.2 and the state merging method in Sec. 3.1 marked 0.00642, 0.00992 and 0.0185, respectively. Fig. 4 compares the scores of those methods on the 51 problems.

References

Christopher M. Bishop. *Pattern recognition and machine learning*. Springer-Verlag, 2006.

Raphael C. Carrasco and Jose Oncina. Learning stochastic regular grammars by means of a state merging method. *ICGI*, pages 139–150, 1994.

Jianfeng Gao and Mark Johnson. A comparison of Bayesian estimators for unsupervised hidden Markov model POS taggers. *EMNLP*, pages 344–352, 2008.

Thomas Niesler and Philip C. Woodland. Variable-length category n-gram language models. *Computer Speech & Language*, 13(1):99–124, 1999.

Franck Thollard, Pierre Dupont, and Colin de la Higuera. Probabilistic DFA inference using Kullback-Leibler divergence and minimality. *ICML*, pages 975–982, 2000.

2. `http://www.intrigger.jp/`
3. `http://www.logos.ic.i.u-tokyo.ac.jp/gxp/`

Appendix A. Figures

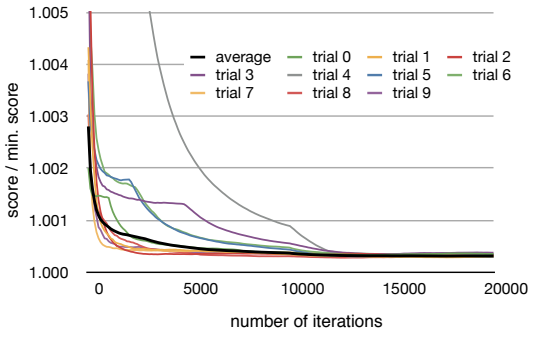

Figure 1: Scores by CGS with different number of iterations for Prob. 19 with $N = 10$ and $\beta = 0.02$

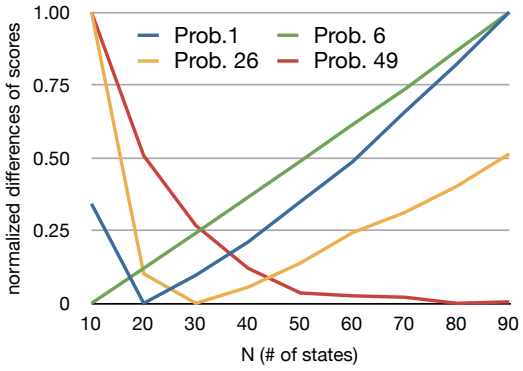

Figure 2: Variation in scores when changing the number of states for Problems 1, 6, 26 and 49

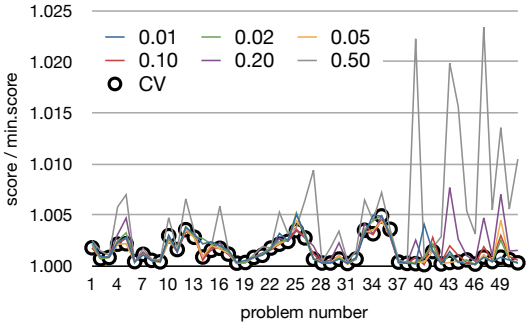

Figure 3: CGS scores with different β

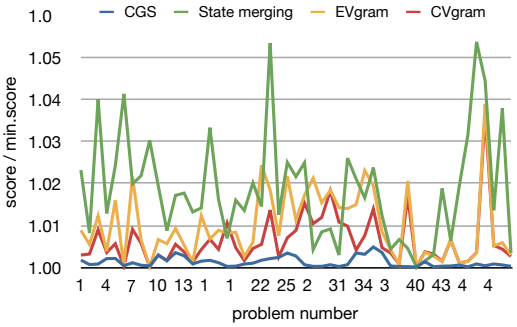

Figure 4: Scores by different methods