# Unsupervised Structure Learning of Stochastic And-Or Grammars (Supplementary Material)

**Kewei Tu** **Maria Pavlovskaia** **Song-Chun Zhu**
Center for Vision, Cognition, Learning and Art
Department of Statistics and Computer Science
University of California, Los Angeles
{tukw,mariapavl,sczhu}@ucla.edu

## 1 Derivation of the Likelihood Gain

In this section, we derive the likelihood gain of adding an And-Or fragment into the grammar. In our learning algorithm when an And-Or fragment is added into the grammar, we try to reduce the training samples using the new grammar rules and update the top-level And-rules accordingly. We denote the set of reductions being made on the training samples by $RD$. Suppose in reduction $rd \in RD$, we replace a configuration $e$ of nodes $a_{1j_1} a_{2j_2} \dots a_{nj_n}$ with the new And-node $A$, where $a_{ij_i} (i = 1 \dots n)$ is an existing terminal or nonterminal node that can be generated by the new Or-node $O_i$ in the And-Or fragment. With reduction $rd$, the Viterbi likelihood of the training sample $x$ where $rd$ occurs is changed by two factors. First, since the grammar now generates the And-node $A$ first, which then generates $a_{1j_1} a_{2j_2} \dots a_{nj_n}$, the Viterbi likelihood of sample $x$ is reduced by a factor of:

$$P(A \to a_{1j_1} a_{2j_2} \dots a_{nj_n}) = \prod_{i=1}^{n} P(O_i \to a_{ij_i}) f_A(a_{1j_1} a_{2j_2} \dots a_{nj_n})$$

where $f_A$ is the probability distribution defined on the relations in the And-rule $A \to O_1 O_2 \dots O_n$. In the derivation below we omit the $f_A$ term for clarity, i.e., we assume deterministic relations in the And-rule. In the final likelihood gain formula, we can add the $f_A$ term back by multiplying the likelihood gain with the factor of $\prod_{rd \in RD} f_A(rd)$ where $f_A(rd)$ is the value of the $f_A$ term in reduction $rd$.

Second, the reduction may make sample $x$ identical to some other training samples, which increases the Viterbi likelihood of sample $x$ by a factor equal to the ratio of the numbers of such identical samples after and before the reduction. This factor can be computed based on the context matrix $CM$, in which each row is a configuration of existing nodes covered by the And-Or fragment, each column is a context which is the surrounding patterns of a configuration, and each element is the number of times that the corresponding configuration and context co-occur in the training set. So the second factor is equal to:

$$\frac{\sum_{e'} CM[e', x-e]}{CM[e, x-e]}$$

where $x - e$ denotes the context of configuration $e$ in sample $x$, and $e'$ in the summation or product range over all the configurations covered by the And-Or fragment.

Putting these two types of changes to the likelihood together, we can formulate the likelihood gain of learning from the And-Or fragment as follows.

$$\begin{aligned}
\frac{P(X|G_{t+1})}{P(X|G_t)} &= \prod_{rd \in RD} \left( \prod_{i=1}^{n} P(O_i \to a_{ij_i}) \right) \frac{\sum_{e'} CM[e', x-e]}{CM[e, x-e]} \\
&= \prod_{i=1}^{n} \prod_{j=1}^{m_i} P(O_i \to a_{ij})^{\|RD_i(a_{ij})\|} \times \frac{\prod_c (\sum_e CM[e,c])^{\sum_e CM[e,c]}}{\prod_{e,c} CM[e,c]^{CM[e,c]}}
\end{aligned}$$

where $G_t$ and $G_{t+1}$ are the grammars before and after learning from the And-Or fragment, $RD_i(a_{ij})$ denotes the subset of reductions in $RD$ in which the $i$-th node of the configuration being reduced is $a_{ij}$, $e'$ and $e$ in the summation or product range over all the configurations covered by the And-Or fragment, and $c$ in the product ranges over all the contexts that appear in $CM$. By applying the Lagrange multiplier method, it can be shown that the probabilities of Or-rules in the And-Or fragment must take the following values in order to maximize the likelihood gain:

$$\forall i, j \quad P(O_i \to a_{ij}) = \frac{\|RD_i(a_{ij})\|}{\sum_{j'=1}^{m_i} \|RD_i(a_{ij'})\|} = \frac{\|RD_i(a_{ij})\|}{\|RD\|}$$

Since the prior probability of the grammar does not involve the rule probabilities, the above optimal values of rule probabilities also maximize the posterior gain. Putting the optimal rule probabilities into the likelihood gain formula, we get:

$$\frac{P(X|G_{t+1})}{P(X|G_t)} = \frac{\prod_{i=1}^{n} \prod_{j=1}^{m_i} \|RD_i(a_{ij})\|^{\|RD_i(a_{ij})\|}}{\|RD\|^{n\|RD\|}} \times \frac{\prod_c (\sum_e CM[e,c])^{\sum_e CM[e,c]}}{\prod_{e,c} CM[e,c]^{CM[e,c]}}$$

## 2 Interpretation of the Likelihood Gain

Despite the complex form of the likelihood gain, it has an intuitive interpretation. We first define the multiplicative coherence of a tensor. A tensor is multiplicatively coherent when the numbers in the tensor are proportional. More specifically, given a tensor $T$ of order $n$ in which the $i$-th index ranges from 1 to $m_i$, $T$ is multiplicatively coherent iff. there exists a constant $\mu$ and a set of constants $\beta_{ij}$ $(i = 1, \ldots, n; j = 1, \ldots, m_i)$ such that

$$T[a_1, a_2, \ldots, a_n] = \mu \prod_{i=1}^{n} \beta_{ia_i}$$

The multiplicative coherence of a tensor $T$ can be measured with the following formula, which is extended from a similar measurement on matrices [1].

$$\frac{\prod_{i=1}^{n} \prod_{j=1}^{m_i} \|T_i(a_{ij})\|^{\|T_i(a_{ij})\|}}{\|T\|^{(n-1)\|T\|} \prod_e \|T(e)\|^{\|T(e)\|}}$$

where $T_i(a_{ij})$ denotes the sub-tensor of $T$ in which the $i$-th index takes the value of $a_{ij}$, $e$ in the product ranges over all the elements of $T$, and $\|\cdot\|$ denotes the summation of all the elements in the enclosed tensor. It can be shown that this formula has a larger value if the elements in the tensor are closer to being multiplicatively coherent, and the formula reaches its maximal value of 1 when the the tensor is perfectly coherent.

Now let us rewrite the likelihood gain formula. Let $RD(e)$ be the subset of reductions in $RD$ in which the configuration being reduced is $e$. Based on the definition of the context matrix $CM$, we have $\|RD(e)\| = \sum_c CM[e,c]$ and $\|RD\| = \sum_{e,c} CM[e,c]$. So we can rewrite the likelihood gain as:

$$\frac{\prod_{i=1}^{n} \prod_{j=1}^{m_i} \|RD_i(a_{ij})\|^{\|RD_i(a_{ij})\|}}{\|RD\|^{(n-1)\|RD\|} \prod_e \|RD(e)\|^{\|RD(e)\|}} \times \frac{\prod_c (\sum_e CM[e,c])^{\sum_e CM[e,c]} \times \prod_e (\sum_c CM[e,c])^{\sum_c CM[e,c]}}{\prod_{e,c} CM[e,c]^{CM[e,c]} \times (\sum_{e,c} CM[e,c])^{\sum_{e,c} CM[e,c]}}$$

The first of the two factors in the formula measures the multiplicative coherence of the *$n$-gram tensor* of the And-Or fragment. The $n$-gram tensor $NA$ of an And-Or fragment is an order $n$ tensor where $n$ is the number of Or-nodes in the fragment. The $i$-th dimension of the $n$-gram tensor $NA$ is indexed by the set of nodes $a_{i1}, a_{i2}, \ldots, a_{im_i}$ that the Or-node $O_i$ can generate. The tensor element $NA[a_1, a_2, \ldots, a_n]$ is the number of times the configuration consisting of $a_1, a_2, \ldots, a_n$ as defined by the And-Or fragment appears in the training samples. It can be seen that if the generation of child nodes at each Or-node is independent of that at the other Or-nodes in the And-Or fragment, then the resulting $n$-gram tensor $NA$ is very likely to be multiplicatively coherent. Therefore, the first factor of the formula provides a surrogate measure of how much the training data support the context-freeness within the And-Or fragment. The second factor of the formula, on the other hand, measures the multiplicative coherence of the context matrix $CM$ (which is an order 2 tensor). If the generation of configurations at the And-Or fragment is independent of its context, then the resulting context matrix $CM$ is very likely to be multiplicatively coherent. Therefore, the second factor of the formula provides a surrogate measure of how much the training data support the context-freeness of the And-Or fragment against its context. By combining these two factors, the likelihood gain measures the support of the training data to the context-freeness of the new And-Or fragment when it is added into the And-Or grammar.

## 3 The Complete Algorithm

We have described the learning algorithm framework in section 3.2 and the algorithm of finding And-Or fragments in section 3.3. Here we give the pseudocode of the complete algorithm. We employ greedy search in the pseudocode, and it is straightforward to extend the code to perform beam search.

**Algorithm 1** Structure Learning of And-Or Grammars

```
Input: the training set $X$
Output: an And-Or grammar $G$
 1: $G \Leftarrow$ the initial grammar constructed from $X$
 2: loop
 3:     $F \Leftarrow \{\}$
 4:     repeat
 5:         $f \Leftarrow$ an And-Or fragment with two Or-nodes and two leaf nodes constructed from a ran-
            domly selected bigram from $X$
 6:         optimize the posterior gain of $f$ using greedy or beam search via four operators:
            adding/removing Or-nodes, adding/removing leaf nodes
 7:         if $f$ increases the posterior gain and $f \notin F$ then
 8:             add $f$ into $F$
 9:         end if
10:     until after a pre-specified number of iterations
11:     if $F$ is empty then
12:         return $G$
13:     end if
14:     $f^* \Leftarrow$ the fragment in $F$ with the highest posterior gain
15:     insert $f^*$ into $G$
16:     reduce $X$ using the grammar rules in $f^*$ and update $G$ accordingly
17: end loop
```

In computing the posterior gain of an And-Or fragment, we need to construct its context matrix. The complete context matrix can be very large and sparse, and we restrict the range of the context to compress the matrix and accelerate the computation.

Note that the grammar learned with the approach described so far does not contain any recursive grammar rule, because the new grammar rules introduced in each learning iteration only specify how the new nonterminal node generates existing terminal or nonterminal nodes but not the reverse. Recursive grammar rules, while not useful in some types of grammars (e.g., image grammars), can be important in other types of grammars (e.g., natural language grammars). In order to learn recursive grammar rules, at the end of each learning iteration we can additionally search for grammar rules that generate the new And-node from existing nonterminal nodes based on the same posterior probability objective.

## 4 Experimental Results

In section 4.2 we have described our experiments of learning image grammars and shown that our approach outperforms the competing approach. Here we illustrate a few grammars learned in the experiments to gain more insight. Figure 1 shows the top levels of the true grammar used to produce the synthetic dataset of animal face sketches [2]. Figure 2 shows the top levels of the grammar produced by our learning approach when trained on 400 synthetic images. It can be seen that the

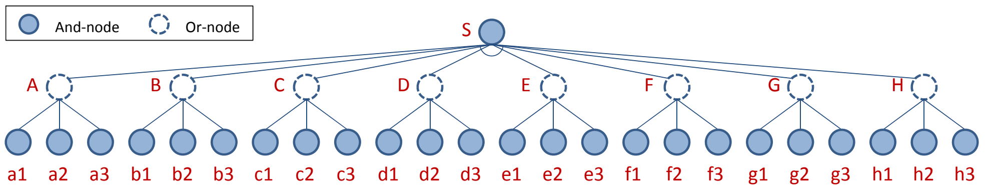

Figure 1: The top three levels of the true grammar that produces the synthetic dataset of animal face sketches [2]. The eight Or-nodes in the second level represent different parts of the animal face, e.g., ears, eyes, nose and mouth. The spatial relations specified at the And-nodes are not shown.

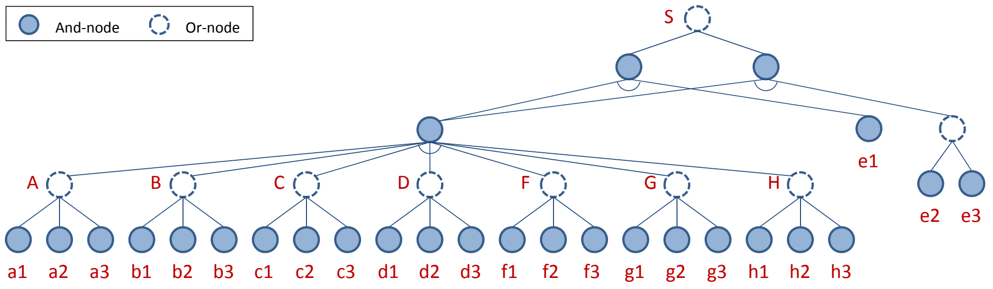

Figure 2: The top levels of the grammar produced by our learning algorithm which was trained on 400 synthetic images.

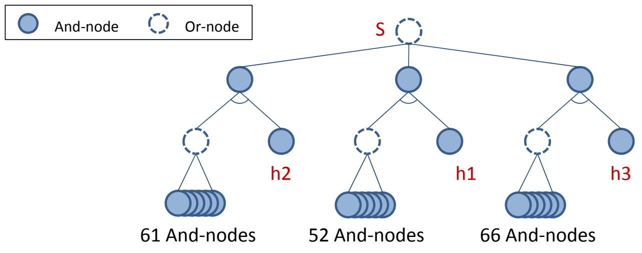

Figure 3: The top levels of the grammar produced by the competing algorithm [3] which was trained on the same set of 400 synthetic images.

learned grammar is equivalent to the true grammar, although it unnecessarily separates the three configurations of node H in the true grammar into two groups. Figure 3 shows the top levels of the grammar produced by the competing approach [3] which was trained on the same set of 400 synthetic images. This grammar has a very different structure from the true grammar in that it enumerates a large number of sub-grammars each representing a small subset of valid compositions, which are then grouped under a few top-level Or-nodes. As a result, the size of the grammar is about 100 times larger than the true grammar and the grammar learned by our approach. It is also less general than the true grammar, as indicated by the low recall. We believe that the separation of learning And-nodes and Or-nodes in [3] is to blame. In contrast, our approach learns And-nodes and Or-nodes in a unified manner via And-Or fragments, which leads to a more compact and accurate grammar.

## References

[1] S. C. Madeira and A. L. Oliveira, “Biclustering algorithms for biological data analysis: A survey.” *IEEE/ACM Trans. on Comp. Biol. and Bioinformatics*, vol. 1, no. 1, pp. 24–45, 2004.

[2] A. Barbu, M. Pavlovskaia, and S. Zhu, “Rates for inductive learning of compositional models,” in *AAAI Workshop on Learning Rich Representations from Low-Level Sensors (RepLearning)*, 2013.

[3] Z. Si and S. Zhu, “Learning and-or templates for object modeling and recognition,” *IEEE Trans on Pattern Analysis and Machine Intelligence*, 2013.