From Stochastic Grammar to Bayes Network:

Probabilistic Parsing of Complex Activity

CVPR14 Submission #83

Supplementary Document

In section 1, we provide a detail description of how the inference is implemented. We further discuss primitive action in section 2 and 3. Finally we show the full grammar of the toy assembly task used in our experiment.

1. Inference by Message Passing

Input:

- The constructed Bayes network.
- CPT $P(v_e \mid v_s)$ for every primitive v
- CPT $P(Z^v \mid v_s, v_e)$ for every primitive v, including special value $P(Z^v \mid !v)$
- Prior information $P(\exists M \mid \exists A)$ for every OR-rule A -> M | …
- Prior information $P(\exists S) = 1$
- Prior information $P(S_s \mid \exists S)$
- Prior information $P(Z^{end} \mid S_e, \exists S)$

Step 0 : For every composition A, recursively compute:

$$P(Z^A \mid !A) = \prod_{\text{M in A}} P(Z^M \mid !M)$$

Note: since scaling the likelihood $P(Z^v \mid v_s, v_e)$ of primitive v does not change the inference result. In our implementation, we allow the value bigger than 1 and scale it so that $P(Z^v \mid !v) = 1$ for every primitive v. Then $P(Z^A \mid !A) = 1$ for every A. Then we can safely ignore them in following calculation.

Step 1 – Forward Phase:

- Forward Phase on primitive v: assume $P(v_s, Z^{pre(v)} \mid \exists v)$ is given, compute:

$$P(v_s, v_e, Z^{pre(v),v} \mid \exists v) = P(v_s, Z^{pre(v)} \mid \exists v) P(v_e \mid v_s) P(Z^v \mid v_s, v_e)$$

$$P(v_e, Z^{pre(v),v} \mid \exists v) = \sum_{t=1}^{T} P(v_s = t, v_e, Z^{pre(v),v} \mid \exists v)$$

- Forward Phase on composition A defined by A -> M N: given $P(A_s, Z^{pre(A)} \mid \exists A)$, compute:

$$P(M_s = t, Z^{pre(M)} \mid \exists M) = P(A_s = t, Z^{pre(A)} \mid \exists A)$$

For t is between 1 and T. This will apply for every t is used in all following formulas.

Recursively perform forward phase on M to get $P(M_e, Z^{pre(M),M} \mid \exists M)$

$$P(N_s = t, Z^{pre(N)} \mid \exists N) = P(M_e = t, Z^{pre(M)}, M \mid \exists M)$$

Recursively perform forward phase on N to get $P(N_e, Z^{pre(N),N} \mid \exists N)$

$$P(A_e = t, Z^{pre(A),A} \mid \exists A) = P(N_e = t, Z^{pre(N),N} \mid \exists N)$$

- Forward Phase on composition A defined by A -> M | N: given $P(A_s, Z^{pre(A)} \mid \exists A)$, compute:

$$P(M_s = t, Z^{pre(M)} \mid \exists M) = P(A_s = t, Z^{pre(A)} \mid \exists A)$$

$$P(N_s = t, Z^{pre(N)} \mid \exists N) = P(A_s = t, Z^{pre(A)} \mid \exists A)$$

Recursively perform forward phase on M and N to get $P(M_e, Z^{pre(M),M} \mid \exists M)$ and $P(N_e, Z^{pre(N),N} \mid \exists N)$, then:

$$P(A_e = t, Z^{A,pre(A)} \mid \exists A) = P(\exists M \mid \exists A) P(Z^N \mid !N) P(M_e = t, Z^{M,pre(M)} \mid \exists M) \\ P(\exists N \mid \exists A) P(Z^M \mid !M) P(N_e = t, Z^{N,pre(N)} \mid \exists N)$$

- Start from $P(S_s \mid \exists S)$, perform forward phase on S, and recursively on other actions. The output is $P(A_s, Z^{pre(A)} \mid \exists A)$ and $P(A_e, Z^{pre(A),A} \mid \exists A)$ for every A.

Step 2 – Backward Phase: similar to Forward Phase

- Backward Phase on primitive v: assume $P(Z^{post(v)} \mid v_e, \exists v)$ is given, compute:

$$P(v_e, Z^{v,post(v)} \mid v_s, \exists v) = P(Z^{post(v)} \mid v_e, \exists v) P(v_e \mid v_s) P(Z^v \mid v_s, v_e)$$

$$P(Z^{v,post(v)} \mid v_s, \exists v) = \sum_{t=1}^{T} P(v_e = t, Z^{v,post(v)} \mid v_s, \exists v)$$

- Backward Phase on composition A defined by A -> M N: given $P(Z^{post(A)} \mid A_e, \exists A)$, compute:

$$P(Z^{post(N)} \mid N_e = t, \exists N) = P(Z^{post(A)} \mid A_e = t, \exists A)$$

Recursively perform backward phase on N to get $P(Z^{N,post(N)} \mid N_s, \exists N)$

$$P(Z^{post(M)} \mid M_e = t, \exists M) = P(Z^{N,post(N)} \mid N_s = t, \exists N)$$

Recursively perform backward phase on N to get $P(Z^{M,post(M)} \mid M_s, \exists M)$. Then:

$$P(A^{A,post(A)} \mid A_s = t, \exists A) = P(Z^{M,post(M)} \mid M_s = t, \exists M)$$

- Backward Phase on composition A defined by A -> M | N: given $P(Z^{post(A)} \mid A_e, \exists A)$, compute:

$$P(Z^{post(M)} \mid M_e = t, \exists M) = P(Z^{post(A)} \mid A_e = t, \exists A)$$

$$P(Z^{post(N)} \mid N_e = t, \exists N) = P(Z^{post(A)} \mid A_e = t, \exists A)$$

Recursively perform backward phase on M and N to get $P(Z^{M,post(M)} \mid M_s, \exists M)$ and $P(Z^{N,post(N)} \mid N_s, \exists N)$. Then:

$$P(Z^{A,post(A)} \mid A_s = t, \exists A) = P(\exists M \mid \exists A) P(Z^{N} \mid !N) P(Z^{M,post(M)} \mid M_s = t, \exists M)$$

$$+ P(\exists N \mid \exists A) P(Z^{M} \mid !M) P(Z^{N,post(N)} \mid N_s = t, \exists N)$$

- Start from $P(Z^{end} \mid S_e, \exists S)$, perform backward phase on S and recursively on other action. The output is $P(Z^{post(A)} \mid A_e, \exists A)$ and $P(Z^{A,post(A)} \mid A_s, \exists A)$.

Step 3 – Compute Posterior Probability: by multiplying forward and backward messages:

$$P(A_s, Z^{pre(A),A,post(A)} \mid \exists A) = P(A_s, Z^{pre(A)} \mid \exists A) P(Z^{A,post(A)} \mid A_s, \exists A)$$

$$P(A_e, Z^{pre(A),A,post(A)} \mid \exists A) = P(A_e, Z^{pre(A),A} \mid \exists A) P(Z^{post(A)} \mid A_e, \exists A)$$

$$P(A_s, Z \mid \exists A) = P(A_s, Z^{pre(A),A,post(A)} \mid \exists A) \prod_{\text{M not in pre(A), A, post(A)}} P(Z^{M} \mid !M)$$

$$P(A_e, Z \mid \exists A) = P(A_e, Z^{pre(A),A,post(A)} \mid \exists A) \prod_{\text{M not in pre(A), A, post(A)}} P(Z^M \mid !M)$$

If v is a primitive we can have the joint distribution:

$$P(v_s, v_e, Z^{pre(v),v,post(v)} \mid \exists v) = P(v_s, v_e, Z^{pre(v),v} \mid \exists v) P(Z^{post(v)} \mid v_e, \exists v)$$

$$P(v_s, v_e, Z \mid \exists v) = P(v_s, v_e, Z^{pre(v),v,post(v)} \mid \exists v) \prod_{\text{M not in pre(v), v, post(v)}} P(Z^M \mid !M)$$

Step 4 – Compute the happening probability: Start from $P(\exists S \mid Z) = P(\exists S) = 1$

- For AND-rule A -> M N, assume $P(\exists A \mid Z)$ is given, compute:

$$P(\exists M \mid Z) = P(\exists N \mid Z) = P(\exists A \mid Z)$$

- For OR-rule A -> M | N, given $P(\exists A \mid Z)$, compute:

$$P(\exists M, Z \mid \exists A) = P(\exists M \mid \exists A) \sum_{t=1}^{T} P(M_e = t, Z \mid \exists M)$$

$$P(\exists N, Z \mid \exists A) = P(\exists N \mid \exists A) \sum_{t=1}^{T} P(N_e = t, Z \mid \exists N)$$

$$P(\exists M \mid Z) = P(\exists A \mid Z) \frac{P(\exists M, Z \mid \exists A)}{P(\exists M, Z \mid \exists A) + P(\exists N, Z \mid \exists A)}$$

$$P(\exists N \mid Z) = P(\exists A \mid Z) \frac{P(\exists N, Z \mid \exists A)}{P(\exists M, Z \mid \exists A) + P(\exists N, Z \mid \exists A)}$$

Output: For every action A in the grammar: $P(\exists A \mid Z)$, $P(A_s, Z \mid \exists A)$ and $P(A_e, Z \mid \exists A)$. If v is a primitive we can have the joint: $P(v_s, v_e, Z \mid \exists v)$.

Optionally we can have 2 more steps:

Step 5: For every action A, we can compute $P(A_s \mid Z)$ and $P(A_e \mid Z)$. If v is a primitive then we also have $P(v_s, v_e \mid Z)$.

Probability of label of a time step t being action A: $P(label_t = A \mid Z)$ can also be derived.

The calculation is shown in section 4.5 in the paper.

Step 6 (optional & not in the paper) Compute the joint of the start and the end for every action: This can be done if needed. However the computational complexity will change.

- For every primitive v, compute $P(v_e, Z^v \mid v_s) = P(v_e \mid v_s) P(Z^v \mid v_s, v_e)$
- For every composition A -> M, N:

  Recursively compute $P(M_e, Z^M \mid M_s)$, $P(N_e, Z^N \mid N_s)$. Then:

$$P(A_e = \beta, Z^A \mid A_s = \alpha) = \sum_{t=1}^{T} P(M_e = t, Z^M \mid M_s = \alpha) P(N_e = \beta, Z^N \mid N_s = t)$$

- For every composition A -> M | N:

  Recursively compute $P(M_e, Z^M \mid M_s)$, $P(N_e, Z^N \mid N_s)$. Then:

$$P(A_e = \beta, Z^A \mid A_s = \alpha) = P(\exists M \mid \exists A) P(Z^N \mid !N) P(M_e = \beta, Z^M \mid M_s = \alpha)$$

$$+ P(\exists N \mid \exists A) P(Z^M \mid !M) P(N_e = \beta, Z^N \mid N_s = \alpha)$$

  For every value of α, β between 1 and T.

- Given $P(A_e, Z^A \mid A_s)$ for every A, we can compute the joint:

$$P(A_s, A_e, Z^{pre(A),A,post(A)} \mid \exists A) = P(A_s, Z^{pre(A)} \mid \exists A) P(A_e, Z^A \mid A_s) P(Z^{post(A)} \mid A_e)$$

$$P(A_s, A_e, Z \mid \exists A) = P(A_s, A_e, Z^{pre(A),A,post(A)} \mid \exists A) \prod_{\text{M not in pre(A), A, post(A)}} P(Z^M \mid !M)$$

- Then for every A, we can compute $P(A_s, A_e \mid Z)$ similar to step 5.

Computational Complexity: The inference process starts from S and then performs on all symbols recursively like a depth-first-search travel on the AND-OR tree representation of the grammar. (Note that if the grammar does not have any OR-rule, it becomes a traditional message passing algorithm on a linear chain). Each symbol is "visited" 4 times (4 above steps), there are calculations of vectors of size Tx1 and matrices of size TxT in the step 1 and step 2. Overall the complexity is $O(KT^2)$ where K is the number of symbol in the compiled grammar. With K=50 and T=1000, our Matlab implementation runs in 0.1 second on an average machine (CPU 2.5GHz, RAM 6GB).

If step 6 is performed, the complexity becomes $O(KT^3)$.

Note that even running in streaming mode, each inference is independent of each other. Hence the inference rate does not need to be the same as video rate. In fact one can choose to only perform inference when needed.

2. Primitive action

Calculating D_v for primitive action v is only ingredient for the Bayes network that makes use of the test sequence and can be the trickiest one to compute. We assume, for each primitive, there is a detector that will output the TxT "heatmap" D_v of the likelihood of the action for every possible interval. Ideally if the action starts at α_0 and ends at β_0 then $D_v[\alpha_0, \beta_0]$ would be high and $D_v[\alpha, \beta]$ would be low for every other α, β value.

The detector is assumed to be black-box. It can be driven by explicitly detecting the start and the end of the action (experiment in section 6.2 and 5). An alternative way is to perform sliding-window-detection using statistics/features computed over the $[\alpha, \beta]$ segment (experiment in section 6.1). Note that the calculation of $D_v[\alpha, \beta]$ can use the information of the entire input sequence if desired, not just the $[\alpha, \beta]$ segment.

As the factor P(v.end | v.start) accounts for the duration of the action and the factors $P(Z^v \mid v.start, v.end)$ accounts for the visual information of the action, the visual detector do not need to concern about duration. Although one could derive a detector like that (or combine the 2 factors into 1 single factor $P(v.end, Z^v \mid v.start)$), we find that keeping these 2 factors separate is more flexible. That way we could change them independently, and we can run in streaming mode, where visual information is feed sequentially.

Interpretation of the special value $D_v[-1,-1]$

This value represents how likely the action v does not happen. Traditionally, non-maxima suppression and thresholding are performed on the heatmap D_v to obtain a set of detections. One can interpret the $D_v[-1,-1]$ as the threshold: a high value means the action more likely does not happen. Informally $D_v[\alpha, \beta] / D_v[-1,-1] > 1$ means segment $[\alpha, \beta]$ is a positive and the confidence is proportional with that ratio value. Where $D_v[\alpha, \beta] / D_v[-1,-1] = 1$ basically means "nothing is known about $[\alpha, \beta]$" (we made use of this in streaming mode). In our implementation, we choose $D_v[-1,-1]$ to be about the expected detection score so that it has above properties (though if one has a way to check if the action does not happen, it could also be incorporated).

Intuitively, $D_v[\alpha, \beta]$ and $D_v[-1,-1]$ put relative weights on the probability of sequence where the action happens and the sequences where it does not, respectively, when we are considering the OR-rule. For example if $D_v[\alpha,\beta]/D_v[-1,-1]$ is very big for some value α,β and these values are also temporally consistent with overall activity's structure, this would contribute to increase the posterior probability of the sequence where action v happens. Note that if the grammar does not have any OR-rules, then the value $D_v[-1,-1]$ will not affect the inference result.

3. Special primitive action

One can include primitive actions with special duration factor or visual observation factor to serve specific purpose.

0-Duration Action: This action always has duration to be 0; and $D_v[-1,-1]=1$, and $D_v[\alpha,\beta]=f(Z^v)$. It will not affect the inference result of action localization within a sequence of actions. However its visual observation factor will affect the relative posterior probabilities between sequences where it happens and the sequences where it doesn't.

Dummy Action: This action does not have a visual observation factor (or equivalently constant likelihood value: $D_v[\alpha,\beta]=D_v[-1,-1]=1$). It can serve as the gap between 2 actions in case we assume the start time of the next action is not the same as the end time of the current one.

Negative-Duration Dummy Action: similar to above dummy action, except its duration is allowed to be negative. Including this between 2 actions allows them to overlap each other. This will be useful for approaches that recognize activity by recognizing overlapping segments of that activity.

Waiting Action: in the Human-Robot collaboration application that we applied our method, the robot delivers the bins to the human operator. In case the human needs a specific bin that the robot has not yet delivered, he will have to wait. Therefore we designed a special "waiting" action that can starts at any moment in time but only ends when that bin is delivered (if the bin is already available, the action, if starts, will end immediately and have duration of 0).

4. Toy assembly grammar

S	→	Body, (Wheel | null), NWT, sticker
NWT	→	NWT_AB ~ 60% | NWT_C ~ 40%
NWT_C	→	Nose_C, ((Wing_C, Tail_C) | (Tail_C, Wing_C))
NWT_AB	→	Nose_AB, (WT_A | WT_B)
WT_A	→	(Wing_A, Tail_A) | (Tail_A, Wing_A)
WT_B	→	(Wing_B, Tail_B) | (Tail_B, Wing_B)
Body	→	body1, body2, body3, body4
Wheel	→	wheel1, wheel2
Nose_AB	→	nose_ab1, nose_ab2, nose_ab3, nose_ab4

Nose_C	→	nose_c1, nose_c2, nose_c3
Wing_A	→	wing_a1, wing_a2, wing_a3
Wing_B	→	wing_b1, wing_b2, wing_b3, wing_b4
Wing_C	→	wing_c1, wing_c2, wing_c3, wing_c4, wing_c5, wing_c6
Tail_A	→	tail_a1, tail_a2, tail_a3
Tail_B	→	tail_b1, tail_b2, tail_b3, tail_b4
Tail_C	→	tail_c1, tail_c2, tail_c3, tail_c4, tail_c5, tail_c6

Map between primitive actions and corresponding bins:

body1	5	
body2	5	
body3	3	
body4	4	
wheel1	3	
wheel2	3	
nose_ab1	3	
nose_ab2	4	
nose_ab3	3	
nose_ab4	3	
nose_c1		3
nose_c2		4
nose_c3		3
wing_a1		3
wing_a2		1
wing_a3		4
wing_b1		3
wing_b2		1
wing_b3		1
wing_b4		4
wing_c1		3
wing_c2		2
wing_c3		2
wing_c4		1
wing_c5		1
wing_c6		2
tail_a1	3	
tail_a2	5	
tail_a3	4	
tail_b1	3	

218	tail_b2	5
219	tail_b3	5
220	tail_b4	4
221	tail_c1	3
222	tail_c2	2
223	tail_c3	2
224	tail_c4	5
225	tail_c5	5
226	tail_c6	2
227	sticker	2
228		
229		
230		
231		
232		