Deep learning methods for improved decode of linear codes

2021年4月10日 14:14

Traditional Method

For odd layer:

$$x_{i,e=(v,c)} = l_v + \sum_{e'=(v,c'),\, c'\neq c} x_{i-1,e'}$$

For even layer:

$$x_{i,e=(v,c)} = 2\tanh^{-1}\left(\prod_{e'=(v',c),\, v'\neq v} \tanh\left(\frac{x_{i-1,e'}}{2}\right)\right)$$

Output of network:

$$o_v = l_v + \sum_{e'=(v,c')} x_{2L,e'}$$

Neural Belief Propagation Decoder

For odd layer:

$$x_{i,e=(v,c)} = \tanh\left(\frac{1}{2}\left(w_{i,v} l_v + \sum_{e'=(v,c'),\, c'\neq c} w_{i,e,e'} x_{i-1,e'}\right)\right)$$

For even layer:

$$x_{i,e=(v,c)} = 2\tanh^{-1}\left(\prod_{e'=(v',c),\, v'\neq v} x_{i-1,e'}\right)$$

Output of network:

$$o_v = \sigma\left(w_{2L+1,v} l_v + \sum_{e'=(v,c')} w_{2L+1,v,e'} x_{2L,e'}\right) \qquad \sigma(x) \equiv (1+e^{-x})^{-1} \text{ is a sigmoid function}$$

Loss function: binary cross-entropy

Neural Min-sum Decoding (to lower complexity)

For odd layer: same as (1)

For even layer:

$$x_{i,e=(v,c)} = \min_{e'=(v',c),\, v'\neq v} |x_{i-1,e'}| \prod_{e'=(v',c),\, v'\neq v} \operatorname{sign}(x_{i-1,e'}) \qquad (8)$$

Output of network: same as (3)

(normalized) min-sum small weight in range [0,1]

$$x_{i,e=(v,c)} = w \cdot \left(\min_{e'} |x_{i-1,e'}| \prod_{e'} \operatorname{sign}(x_{i-1,e'})\right), \qquad (9)$$
$$e' = (v',c),\ v' \neq v.$$

Neural Normalized Min-sum (NNMS) check to variable (even layer)

$$x_{i,e=(v,c)} = w_{i,e=(v,c)} \cdot \left(\min_{e'} |x_{i-1,e'}| \prod_{e'} \text{sign}(x_{i-1,e'}) \right), \quad e' = (v',c),\ v' \neq v. \tag{10}$$

Offset Min-sum algorithm

$$x_{i,e=(v,c)} = \max\left(\min_{e'} |x_{i-1,e'}| - \beta, 0 \right) \prod_{e'} \text{sign}(x_{i-1,e'}), \quad e' = (v',c),\ v' \neq v. \tag{11}$$

Neural Offset min-sum decoding

$$x_{i,e=(v,c)} = \max\left(\min_{e'} |x_{i-1,e'}| - \beta_{i,e=(v,c)}, 0 \right) \prod_{e'} \text{sign}(x_{i-1,e'}), \quad e' = (v',c),\ v' \neq v. \tag{12}$$

BP-RNN decoding

(V to C) at iteration t

$$x_{t,e=(v,c)} = \tanh\left(\frac{1}{2} \left(w_v l_v + \sum_{e'=(c',v),\, c' \neq c} w_{e,e'} x_{t-1,e'} \right) \right) \tag{13}$$

(C to V) at iteration

$$x_{t,e=(c,v)} = 2\tanh^{-1}\left(\prod_{e'=(v',c),\, v' \neq v} x_{t,e'} \right) \tag{14}$$

Output at t iteration

$$o_{v,t} = \sigma\left(\tilde{w}_v l_v + \sum_{e'=(c',v)} \tilde{w}_{v,e'} x_{t,e'} \right) \tag{15}$$

Multi-loss cross entropy

$$L(o,y) = -\frac{1}{N} \sum_{t=1}^{T} \sum_{v=1}^{N} y_v \log(o_{v,t}) + (1-y_v)\log(1-o_{v,t}) \tag{16}$$

v-th component t-timestep

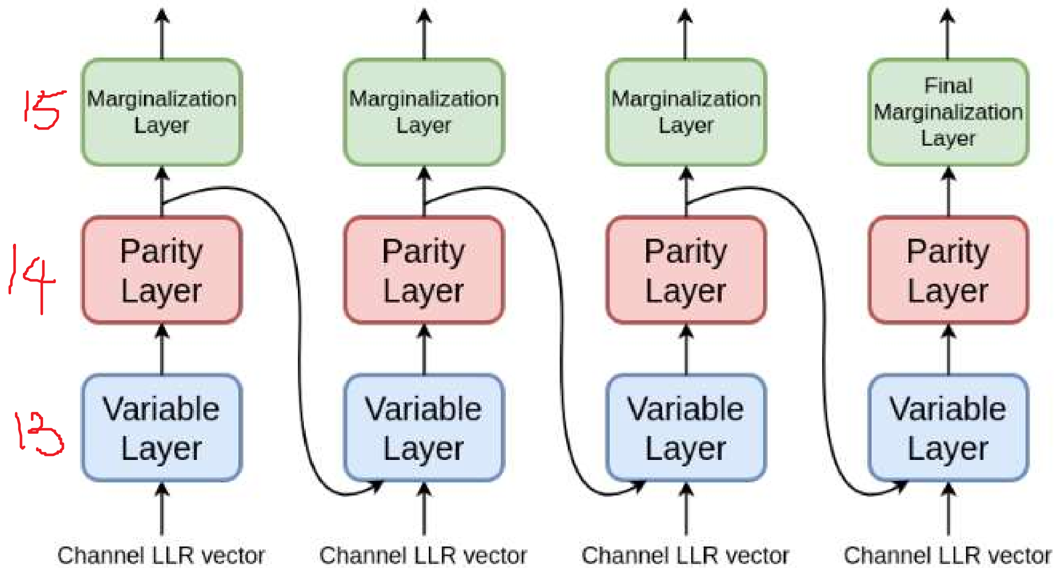


Fig. 2. Recurrent Neural Network Architecture with unfold 4 which corresponds to 4 full BP iterations.

RNN neural min-sum decoder (w is same in all iterations)

Variable layer:

$$x_{i,e=(v,c)} = w_{e=(v,c)} \cdot \left(\min_{e'} |x_{i-1,e'}| \prod_{e'} \operatorname{sign}(x_{i-1,e'}) \right), \qquad (17)$$
$$e' = (v',c),\ v' \neq v,$$

Parity layer:

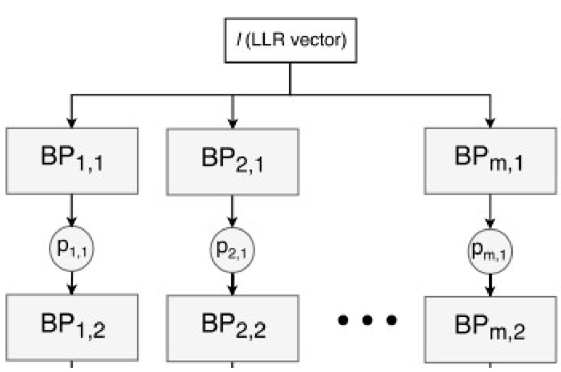
$$x_{i,e=(v,c)} = \max\left(\min_{e'} |x_{i-1,e'}| - \beta_{e=(v,c)}, 0 \right) \prod_{e'} \operatorname{sign}(x_{i-1,e'}),$$
$$e' = (v',c),\ v' \neq v. \qquad (18)$$

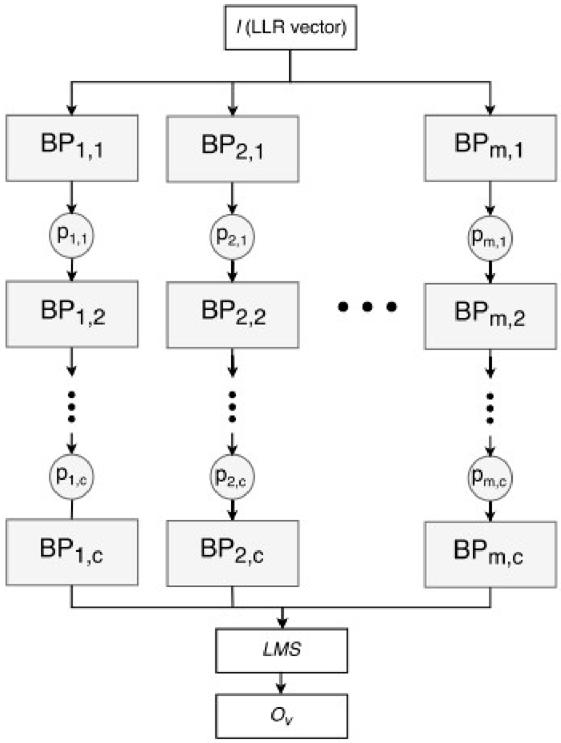
Successive relaxation

$$m'_t = \gamma m'_{t-1} + (1-\gamma) m_t \qquad (19)$$

Relaxation factor

Modified random redundant iterative algorithm




I (LLR vector)
BP1,1
BP2,1
BPm,1
p1,1
p2,1
pm,1
BP1,2
BP2,2
BPm,2
p1,c
p2,c
pm,c
BP1,c
BP2,c
BPm,c
LMS
Ov