Pruning and quantizing neural belief propagation decoders

Tuesday, 9 February 2021 2:18 PM

Topic: near maximum-likelihood decoding for "short" linear block code

Aim: purpose pruned-based neural belief propagation decoding to reduce the complexity
- hypothesis
- Gap
- Research question

Key result or argument

Brief evaluation

The most notable features
- Methodology
    - Consider NBP decoding over an overcomplete PC matrix, and prune by using magnitude of the weight as the measure of importance
    - Consider pruning-based neural offset min-sum (PB-NOMS) decoder
- Results
    - For given complexity, PB-NBP yield performance compared to NBP
    - 0.27-0.31 dB over NBP with up to 97% percent reduction of nodes
    - 0.5 dB from ML decoding with 5-bit quantization for RM code

(difference between NOMS and NBNP: individual offset to each edge of the unrolled graph)
Intro:
- Audience: scholar in comm engineering
- Purpose of the article: demostrate a neural

Summary
- What is the research about: topic
- What research question or hypothesis
- Has the purposed been clearly expressed
- If there are research question, hv been well articulated?

Evaluation - design
- What criteria for evaluation
    - NBP, MMBP (RM code) - near ML performance
- Is the research sound
- Has the appropriate methodology, technology or approach has been used to answer the research questions
- Is the methodology or technology innovative in trying to answer the research questions?
- Is there a better framework, model, criterion or standard benchmark that could have been used?

Evaluation- discussion
- How significant are the results
- What conclusion draw from findings
- Are recommendation valid based on the finding
- To what extend they contribute in the field

2021年2月14日 22:50

Deep learning based communication over air

Friday, 16 August 2019 4:01 PM

Keywords: over-the-air transmissions, complete communication system, neural network, receiver synchronization problem, training over actual channel, autoencoder, deep learning, end-to-end learning, modulation, software-defined radio

Key idea: instead of separating all the parts in communication sys, use deep learning model to optimize transmitter and receiver jointly without any artificially introduced block structure

2n-dimensional output - > n-dimensional complex-valued vector

power constraint

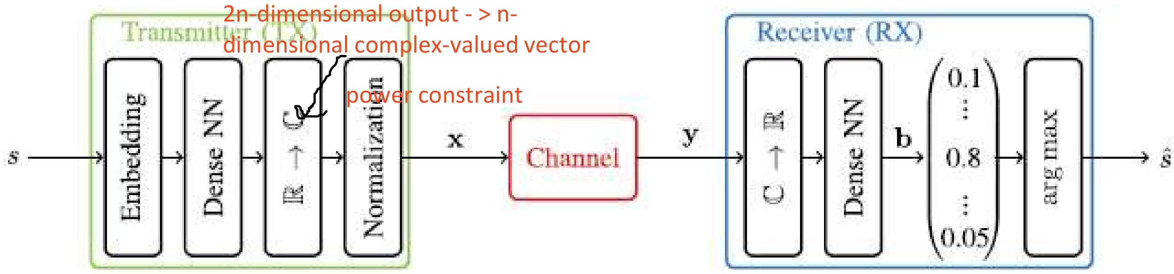


Fig. 2. Representation of a communications system as an autoencoder.

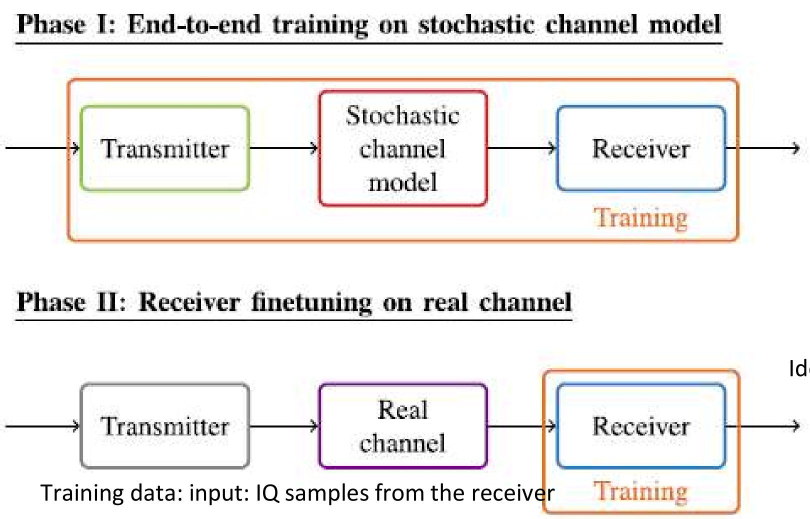
Embedding = one-hot

Point of view: channel acts as a form of regularization which makes is **impossible** for the NN to **overfit**?? Because of the training sample will never see again

Modulation:
for 2bits, learned constellation points are not structured as in regular QAM(non-uniform)
Transmitted symbols are correlated over time

Hardware challenges:
- Unknown channel transfer function
    - Two phases training strategy: train the autoencoder on a stochastic channel and then finetune the receiver with training data obtained from over- the-air trainsmissions
- Hardware effects
    - Real system operates on samples and not symbols
    - Pulse shaping, quantization, hardware imperfections, CFO, offset (require to model them)
- Continuous transmissions
    - Cannot be large block of messages, due to the exponential training complexity
    - Need to transmit a continuous sequence of a smaller number of possible messages. => require timing synchronization, have ISI problem

Two-phase training strategy
-



Idea similar as transfer learning

Fig. 3. Two-phase training strategy.

PHASE 1: train the autoencoder using stochastic channel model that should approximate the expected channel
PHASE 2: we compensete the mismatch between stochastic and actual channel
For this model: the corresponding IQ-samples = input, message indices = output, to **fine-tuning** the receiver (similar as transfer learning)

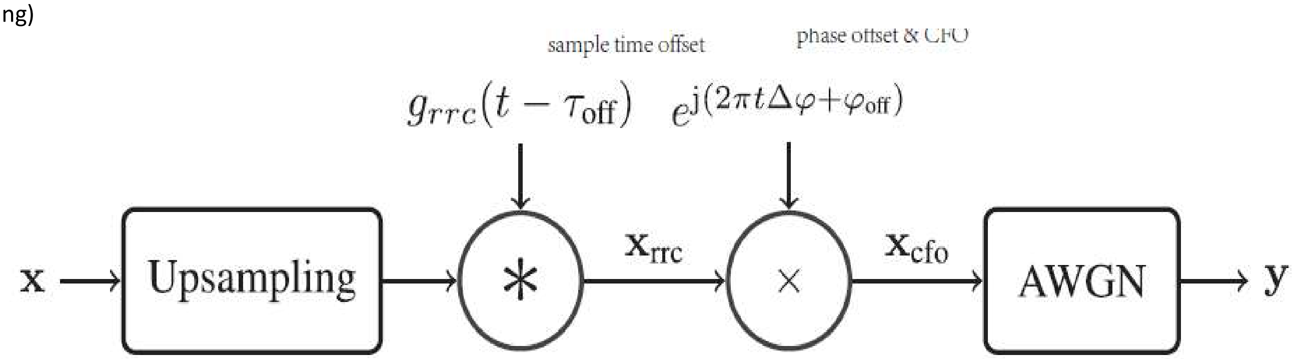


Fig. 4. Stochastic channel model for end-to-end training of the autoencoder.

ISI:

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

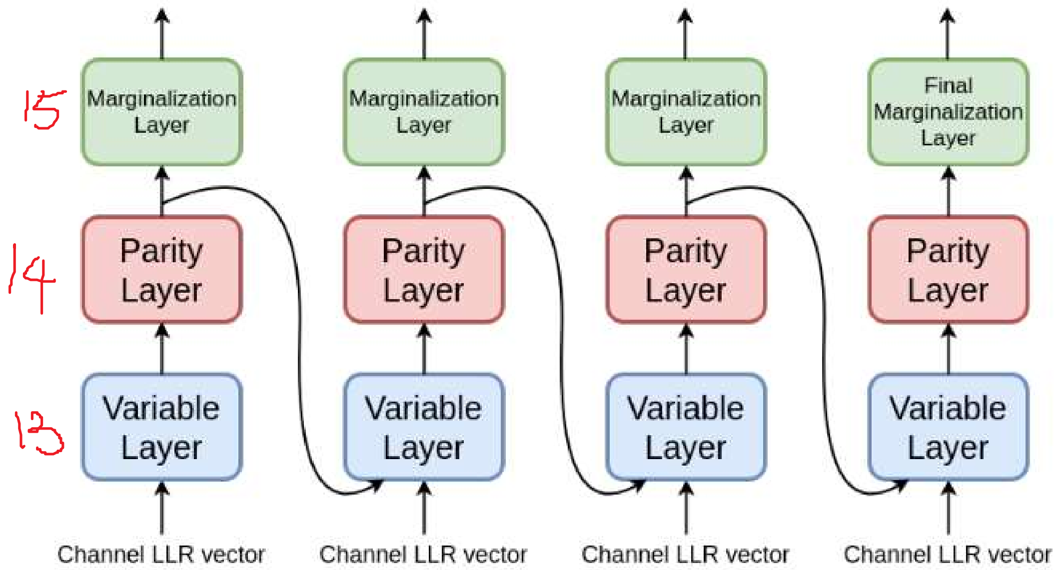


Fig. 2. Recurrent Neural Network Architecture with unfold 4 which corresponds to 4 full BP iterations.

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

Variable layer:

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

Parity layer:

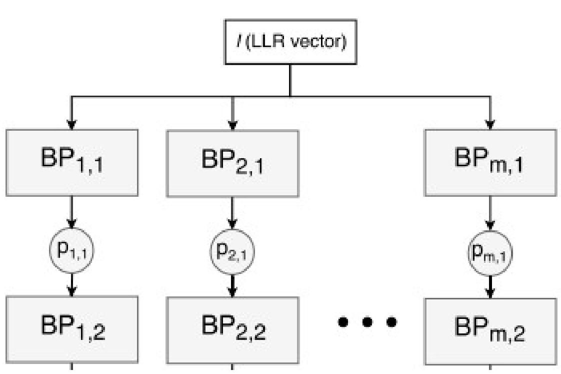
$$x_{i,e=(v,c)} = \max\left(\min_{e'} |x_{i-1,e'}| - \beta_{e=(v,c)}, 0 \right) \prod_{e'} \text{sign}(x_{i-1,e'}), \qquad e' = (v',c),\ v' \neq v. \tag{18}$$

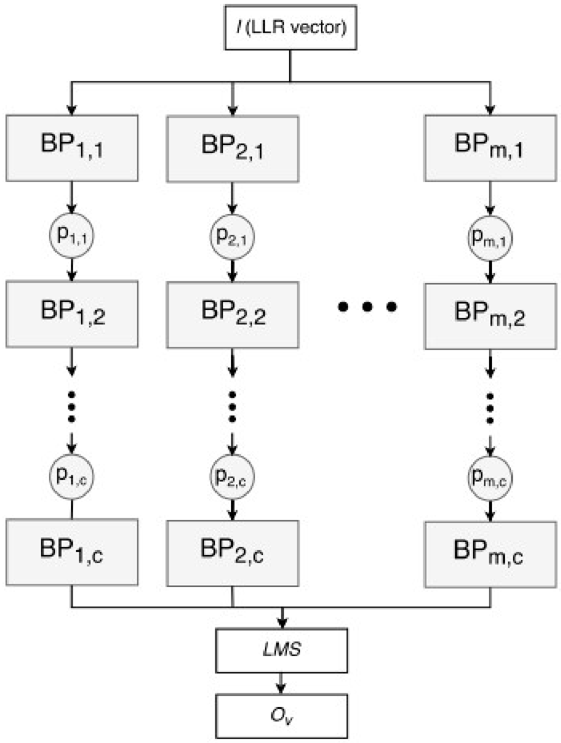
Successive relaxation

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

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