Intelligent Raido Resouce Managment: Learning And Optimization

Qingjiang Shi

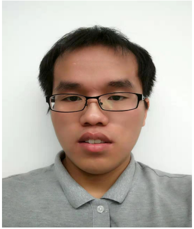
Tongji University

Nov.16, 2019 @ Hangzhou

Research Group

Hao Yu, Master

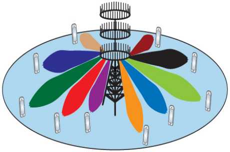
Siyuan Lu, Master



Jintai Yang, Baidu

Introduction

- Radio resource management (RRM) is a large-scale control problem involving
  - transmit power
  - beamforming
  - time-frequency channel
  - modulation-coding scheme
  - ......



- The objective is to utilize the limited radio resources to improve
  - network services
  - quality of service (QoS)
  - overall system performance
- RRM with traditional rule-based algorithms is particularly challenging
  - numerous network functionalities operating at different timescales
  - unprecedented levels of complexity in the 5G/B5G mobile system

Introduction

RRM from a bigdata perspective

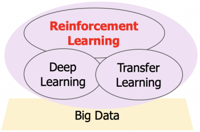
- Networks are data-rich environments
- RRM nowadays derives little insight from such data
- Data-driven approach is promising

Conventional Approach

- Simple rules
- Opt. methods with high complexity
- Become increasingly impractical as
  - more dynamic and diverse traffic
  - more complex netw. architecture
  - more resource structures

AI-Empowered Approach

- **Self-learning** and **adaptive** based on user/channel conditions
- Decision quality improves via learning

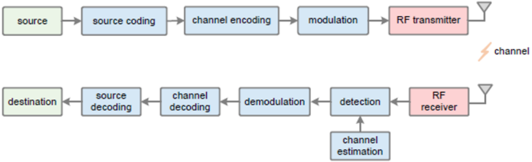


[1] D. Calabrese, et. al., "Learning radio resource management in RANs: framework, opportunities, and challenges," IEEE Communications Magazine, vol. 56, no. 9, pp. 138-145, Sept. 2018.

Introduction

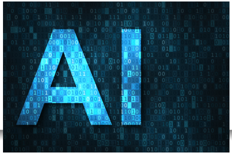
The potential applications of AI in wireless networks

- **Complex modeling**: modeling relashionships of KPIs and network parameters
- **Complex problem solving**: channel scheduling, power control, beamforming
- **AI as communication modules**: demodulation/decoding, or the whole



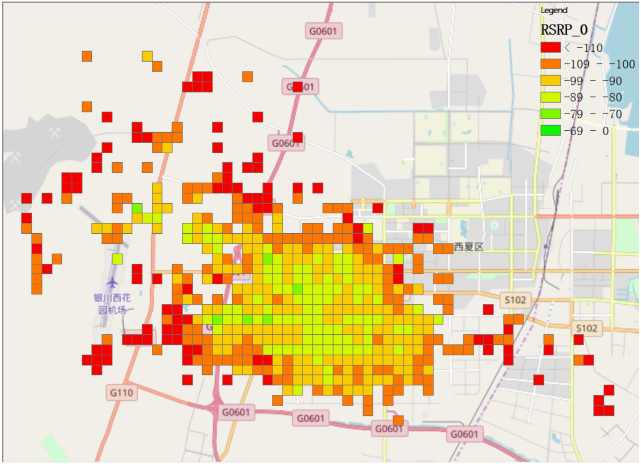
In this talk, we cover

- Data-driven network optimization
- Learning-based massive beamforming



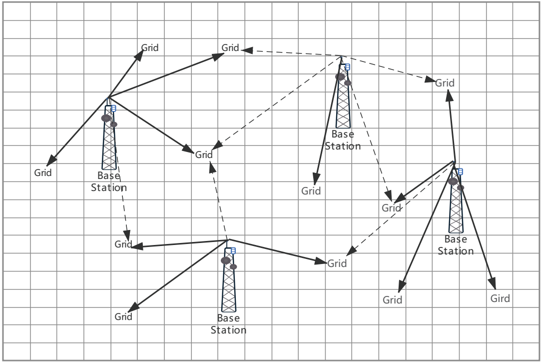
1. Data-Driven Network Optimization

Motivation



- Bad coverage if RSRP$\leq -105dBm$

Challenge



- Data-driven network optimization
  - Where is data from?
  - What is performance metric?
  - How to do parallel optimization?

Math Prob: find $x \in R^n$ s.t.
$f_i(x) \geq \Gamma, \forall i = 1, 2, \ldots, N$

- both n and N are very large
- f_i is a black-box function

Problem statement

- Suppose all problematic areas involve m grid points and n base stations
- The number of RF parameters is defined to be $d \triangleq 4n$.
- We use $\mathbf{x} \in \mathbb{R}^d$ to denote the RF parameters to be optimized.
- Let N_i denote the number of MRs at the i-th grid and $N \triangleq \sum_{i=1}^{m} N_i$.
- The objective to be optimized is given by

$$F(\mathbf{x}) \triangleq -\frac{1}{N} \sum_{i=1}^{m} \frac{N_i}{2} \left(\frac{1}{1 + e^{(-\lambda_1 (f_i(\mathbf{x}) - \Gamma_1))}} + \frac{1}{1 + e^{(-\lambda_2 (g_i(\mathbf{x}) - \Gamma_2))}} \right) \tag{1}$$

where both $f_i(\mathbf{x})$: RSRP and $g_i(\mathbf{x})$: SINR

- modeled from data
- have no analytical form, and are nondifferentiable

The BCD Method for Small-scale Problems

- The problem fits into the BCD framework due to the separable constraints.
- Let $\mathcal{C}_i$ denotes the box constraints on x_i.
- The problem can be equivalently as

$$\min_{\mathbf{x}_1,\cdots,\mathbf{x}_n \in C_1 \times \cdots \times C_n} F(\mathbf{x}_1, \cdots, \mathbf{x}_n)$$

The BCD Method for Small-scale Problems

- In each iteration t, we randomly select an $i_t \in \{1, 2, \cdots, n\}$ and update $\mathbf{x}_{i_t}$ via

$$\mathbf{x}_{i_t}^{(t+1)} = \arg\min_{\mathbf{x}_{i_t}} F\left(\mathbf{x}_1^{(t)}, \cdots, \mathbf{x}_{i_t}^{(t)}, \mathbf{x}_{i_t}, \mathbf{x}_{i_t+1}^{(t)}, \cdots, \mathbf{x}_n^{(t)}\right).$$

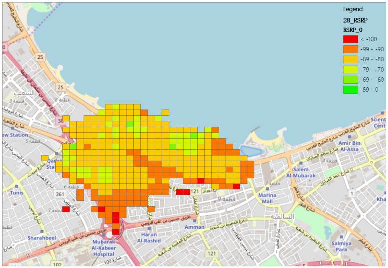
- In our simulations the update of $\mathbf{x}_{i_t}$ is done inexactly, i.e., by

$$\mathbf{x}_{i_t}^{(t+1)} = \mathcal{P}_{\mathcal{C}_{i_t}}\{\mathbf{x}_{i_t}^{(t)} - \alpha_t \nabla_{i_t} F(\mathbf{x}^{(t)})\}.$$

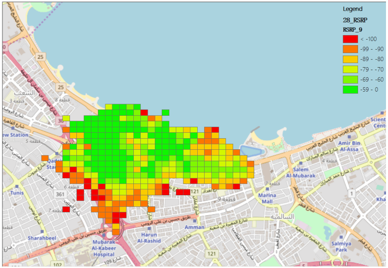
  - $\nabla_{i_t} F(x^{(t)})$ is approximately and numerically calculated
  - α_t is chosen among $\{10, 2, 1, 10^{-1}, 10^{-2}, \ldots, 10^{-7}\}$

Experiment results

- Case 1 : 28 Cells



(a) Initial RSRP values

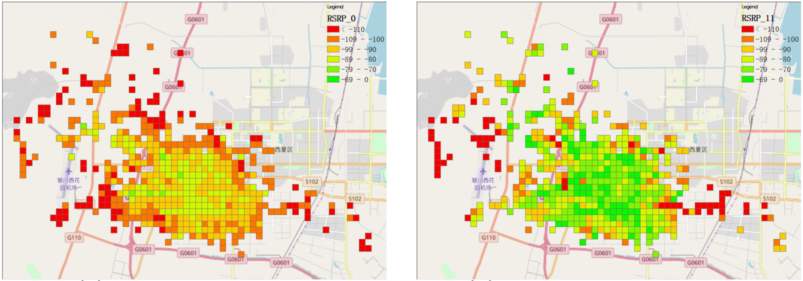


(b) Optimized RSRP values

Figure: The RSRP results before RF tuning and after in the case of 28 cells

Experiment results

- Case 2 : 293 Cells



(a) Initial RSRP values (b) Optimized RSRP values

Figure: The RSRP results before RF tuning and after in the case of 293 cells

Problem reformulation for ADMM

- Define

$$\mathbb{F}_i(\mathbf{x}) \triangleq -\frac{N_i}{2N}\left(\frac{1}{1+e^{(-\lambda_1(f_i(\mathbf{x})-\Gamma_1))}}+\frac{1}{1+e^{(-\lambda_2(g_i(\mathbf{x})-\Gamma_2))}}\right) \quad (2)$$

- Partition the grid points into K subsets, C_k's, each with almost equal size
- Denote the number of grids in subset C_k by n_k, i.e., $n_k = |C_k|$
- Define $F_k(\mathbf{x}) \triangleq \sum_{i\in C_k} \mathbb{F}_i(\mathbf{x})$

Consensus constraints under global mapping

- The RF parameter optimization problem is reformulated as (consensus form)

$$\min_{\{\mathbf{x}_k\},\{\tilde{\boldsymbol{z}}_k\}} \sum_{k=1}^{K} F_k(\mathbf{x}_k) \\ s.t. \quad \mathbf{x}_k = \tilde{\boldsymbol{z}}_k, \forall k \tag{3}$$

where

- $\mathbf{x}_k$: part of cell parameters related to F_k
- $\tilde{z}$: a global variable with $\tilde{z}_k$ corresponding to the parameters in $\mathbf{x}_k$

ADMM algorithm

- Furthermore, let us define
$L_{\rho}^{k}(\mathbf{x}_k, \boldsymbol{y_k}, \tilde{\boldsymbol{z}}_k)) \triangleq F_k(\mathbf{x}_k) + y_k^T \sqrt{w_k}(\mathbf{x}_k - \tilde{\boldsymbol{z}}_k) + (\rho/2) w_k \|\mathbf{x}_k - \tilde{\boldsymbol{z}}_k\|_2^2$

- Then the main iteration of the ADMM iteration is as follows

$$\mathbf{x}_k^{t+1} = \arg\min_{\mathbf{x}_k \in \mathcal{C}_k} (F_k(\mathbf{x}_k) + \left(\boldsymbol{y}_k^t\right)^T \sqrt{w_k}(\mathbf{x}_k - \tilde{\boldsymbol{z}}_k^t) + (\rho/2) w_k \left\|\mathbf{x}_k - \tilde{\boldsymbol{z}}_k^t\right\|_2^2) \tag{4}$$

$$\boldsymbol{y}_k^{t+1} = \boldsymbol{y}_k^t + \rho\sqrt{w_k}(\mathbf{x}_k^{t+1} - \tilde{\boldsymbol{z}}_k^{t+1}) \tag{5}$$

- Note that the update of $\mathbf{x}_k$ and $\boldsymbol{y}_k$ can be carried out in parallel.

[2] S. Boyd, et. al., Distributed optimization and statistical learning via the alternating direction method of multipliers, Foundations and Trends in Machine Learning, 3(1):1–122, 2011

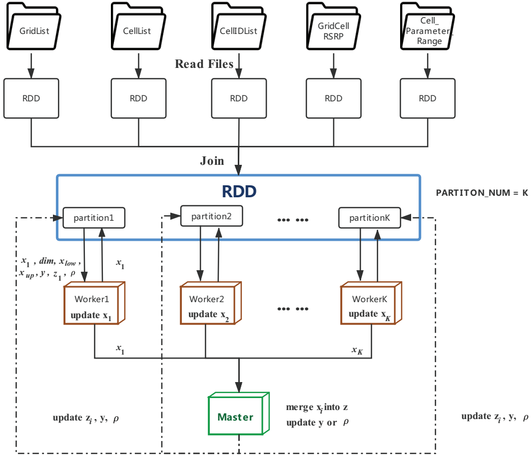
PDD—a variant of ADMM

Algorithm 1 Penalty Dual Decomposition (PDD)

initialize $\tau < 1$, ρ, $\mathbf{x}_k^0 = \arg\min_{\mathbf{x}_k}(F_k(\mathbf{x}_k))$;
set $iter = 0$ and $\eta_0 = MAX\_INT$
while $iter < MAX\_ITER$ **do**
  update $\mathbf{x}_k$, $\forall k$, by using BCD
  update $\tilde{\boldsymbol{z}}_k$, $\forall k$
  $h_{iter} = \sum_{k=1}^{K} \|\mathbf{x}_k - \tilde{\boldsymbol{z}}_k\|_2$
  if $h_{iter} <= \eta_{iter}$ **then**
    update $\boldsymbol{y}_k$, $\forall k$
  else
    increment ρ
  end if
  $iter = iter + 1$
  $\eta_{iter} = \tau \min(\eta_{iter-1}, h_{iter-1})$
end while

[3] Q Shi, M Hong, X Fu, TH Chang, Penalty dual decomposition method for nonsmooth nonconvex optimization, submited to IEEE TSP.

Architecture of PDD on Spark



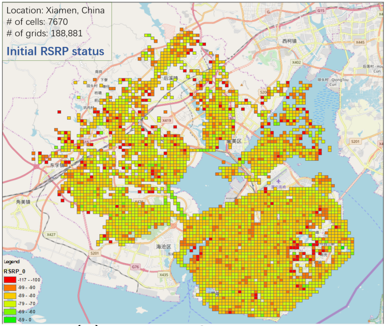
Cell Partition-based Parallel (CPP) Algorithm

- Clearly, each grid point can be interfered by neighboring cells.
- However, it is more desirable to divide the large-scale problem into small subproblems based on cell partition.
- The key to such kind of method is to partition the cells so that the interference impact is as small as possible.
- Moreover, to balance the load, size-constrained K-mean is proposed.

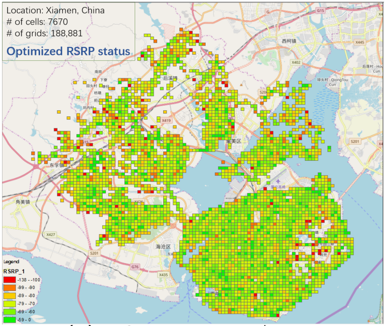
Experiment results

Clustering	# Partition	# Cell	Running Time	Score
K-Means	10	154	312s	0.8676
	15	392	1133s	0.8775
	20	579	2844s	0.873
Size-cons. K-means	10	154	235s	0.854
	20	579	1600s	0.837
	30	1413	3588s	0.861
	40	7670	15495s	0.903

Experiment results



(a) Initial RSRP values



(b) RSRP by CPP/PDD

Figure: The RSRP results before CPP/PDD and after in the case of 7670 cells

Remarks

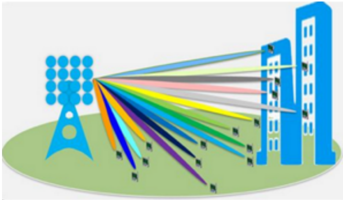
- PDD yields better score than CPP but CPP is much more efficient
- We can run one round BCD to improve the performance of the PDD/CPP
- For example, the performance of CPP can be improved as shown below

Partition	Cell	Running Time	Score before BCD	Score
10	154	642s	0.88	0.928

2 Learning-based Massive Beamforming

Motivation

- A basic problem of massive MU-MIMO is beamformer design to achieve downlink system throughput maximization
- The classical WMMSE algorithm has complexity of $O(N_T^3)$



- Deep learning can well approximate iterative optimization methods with lower complexity
- We here consider using deep learning to learn 'WMMSE' in the massive MU-MIMO case

[4] H. Sun, X. Chen, Q. Shi, et. al., "Learning to optimize: training deep neural networks for interference management," IEEE Trans. Signal Processing, vol. 66, no. 20, pp. 5438-5453, Oct.15, 2018.
[5] W. Xia, G. Zheng, Y. Zhu, et. al., "Deep learning based beamforming neural networks in downlink MISO systems," 2019 IEEE ICC Workshops, pp. 1-5.

Problem statement

- Consider a single cell K-users massive MIMO system
- The BS is equipped with N_T antennas, each user with N_R antennas.
- The received signal $\mathbf{y}_k \in \mathbb{C}^{N_R \times 1}$ at user k can be written as

$$\begin{aligned}\mathbf{y}_k &= \mathbf{H}_k\mathbf{x} + \mathbf{n}_k \\ &= \underbrace{\mathbf{H}_k\mathbf{V}_k\mathbf{s}_k}_{\text{desired signal of user } k} + \underbrace{\sum_{j=1, j\neq k}^{K} \mathbf{H}_k\mathbf{V}_j\mathbf{s}_j}_{\text{multi-user interference}} + \mathbf{n}_k, \forall k.\end{aligned}$$

where

- $\mathbf{H}_k \in \mathbb{C}^{N_R \times N_T}$: the channel matrix from the BS to user k
- $\mathbf{V}_k \in \mathbb{C}^{N_T \times d_k}$: the transmit beamformer of user k
- $\mathbf{s}_k \in \mathbb{C}^{d_k \times 1}$: the transmitted symbols of user k
- $\mathbf{n}_k \in \mathbb{C}^{N_R \times 1}$: the AWGN with distribution $\sim \mathcal{CN}(0, \sigma_k^2\mathbf{I})$

Problem statement

- The system weighted sum-rate maximization can be written as follows

$$\begin{aligned} \max_{\{\mathbf{V}_k\}} \quad & \sum_{k=1}^{K} \alpha_k R_k \\ \text{s.t.} \quad & \sum_{k=1}^{K} \mathrm{Tr}\left(\mathbf{V}_k \mathbf{V}_k^H\right) \leq P_{max}, \end{aligned}$$

where

- P_{max} denotes the BS power budget
- the weight α_k represents the priority of user k
- R_k is the rate of user k given by

$$R_k \triangleq \log\det\left(\mathbf{I}+\mathbf{H}_k\mathbf{V}_k\mathbf{V}_k^H\mathbf{H}_k^H\left(\sum_{m\neq k}\mathbf{H}_k\mathbf{V}_m\mathbf{V}_m^H\mathbf{H}_k^H+\sigma_k^2\mathbf{I}\right)^{-1}\right).$$

WMMSE

- Equivalent problem

$$R_k \triangleq \log\det\left(\mathbf{I}+\mathbf{H}_k\mathbf{V}_k\mathbf{V}_k^H\mathbf{H}_k^H\left(\sum_{m\neq k}\mathbf{H}_k\mathbf{V}_m\mathbf{V}_m^H\mathbf{H}_k^H+\sigma_k^2\frac{\sum_{k=1}^{K}\operatorname{Tr}\left(\mathbf{V}_k\mathbf{V}_k^H\right)}{P_{max}}\mathbf{I}\right)^{-1}\right).$$

- Define $\tilde{\mathbf{H}}_k = \sqrt{\frac{P_{max}}{\sigma_k^2}}\mathbf{H}_k$

$$R_k \triangleq \log\det\left(\mathbf{I}+\tilde{\mathbf{H}}_k\mathbf{V}_k\mathbf{V}_k^H\tilde{\mathbf{H}}_k^H\left(\sum_{m\neq k}\tilde{\mathbf{H}}_k\mathbf{V}_m\mathbf{V}_m^H\tilde{\mathbf{H}}_k^H+\sum_{k=1}^{K}\operatorname{Tr}\left(\mathbf{V}_k\mathbf{V}_k^H\right)\mathbf{I}\right)^{-1}\right).$$

WMMSE

- Define

$$\mathbf{E}_k \triangleq (\mathbf{I} - \mathbf{U}_k^H \mathbf{H}_k \mathbf{V}_k)(\mathbf{I} - \mathbf{U}_k^H \mathbf{H}_k \mathbf{V}_k)^H + \sum_{m \neq k} \mathbf{U}_k \mathbf{H}_k \mathbf{V}_m \mathbf{V}_m^H \mathbf{H}_k^H \mathbf{U}_k^H + \sum_{i=1}^{K} \mathrm{Tr}\,(\mathbf{V}_k \mathbf{V}_k^H) \mathbf{U}_k^H \mathbf{U}_k$$

- Equivalent WMMSE form

$$\min_{\{\mathbf{W}_k, \mathbf{U}_k, \mathbf{V}_k\}} \quad \sum_{k=1}^{K} (\log\det(\mathbf{W}_k) - \mathrm{Tr}\,(\mathbf{W}_k \mathbf{E}_k))$$

- Update of $\mathbf{V}_k$ in WMMSE

$$\mathbf{V}_k = \left(\sum_{j=1}^{K} \alpha_j \mathrm{Tr}\,(\mathbf{U}_j \mathbf{W}_j \mathbf{U}_j^H) \mathbf{I} + \sum_{j=1}^{K} \alpha_j \mathbf{H}_j^H \mathbf{U}_j \mathbf{W}_j \mathbf{U}_j^H \mathbf{H}_j \right)^{-1} \alpha_k \mathbf{H}_k^H \mathbf{U}_k \mathbf{W}_k$$

- The complexity of each iteration is at least $O(N_T^3)$.

Technical Challenges For Deep Learning

- Challenge 1: High dimensional matrix, not easy to train
- Challenge 2: The weights α_k's often change with time
- Challenge 3: Sometimes only single stream transmission is scheduled for some user

The Proposed Solution to Challenge 1: Reduced WMMSE (R-WMMSE)

- Define $\mathbf{H} \triangleq \left[\mathbf{H}_1^H \ \ \mathbf{H}_2^H \ \ \ldots \mathbf{H}_K^H\right]^H \in \mathbb{C}^{KN_R \times N_T}$
- It can be proven $\mathbf{V}_k = \mathbf{H}^H \mathbf{X}_k$ for some $\mathbf{X}_k \in \mathbb{C}^{KN_R \times d_k}$
- Update of $\mathbf{X}_k$ is given by

$$\mathbf{X}_k = \left(\sum_{j=1}^{K} \alpha_j \mathrm{Tr}\left(\mathbf{U}_j \mathbf{W}_j \mathbf{U}_j^H\right)(\mathbf{H}\mathbf{H}^H) + \sum_{j=1}^{K} \alpha_j \mathbf{H}\mathbf{H}_j^H \mathbf{U}_j \mathbf{W}_j \mathbf{U}_j^H \mathbf{H}_j \mathbf{H}^H \right)^{-1} \times \alpha_k \mathbf{H}\mathbf{H}_k^H \mathbf{U}_k \mathbf{W}_k$$

- R-WMMSE with $O(K^3)$ Vs. the classical WMMSE with $O(N_T^3)$

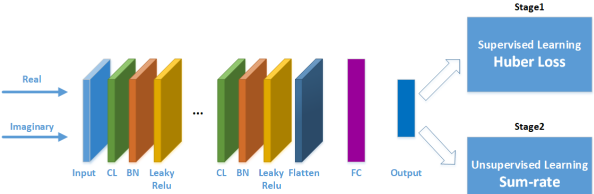
Learning Scheme

- Supervised Learning
  - CNN
  - DNN
- Unsupervised Learning

$$L(\theta;h) \triangleq -\sum_{k=1}^{K} \log\det\left(\mathbf{I}+\mathbf{H}_k\mathbf{V}_k\mathbf{V}_k^H\mathbf{H}_k^H\left(\sum_{m\neq k}\mathbf{H}_k\mathbf{V}_m\mathbf{V}_m^H\mathbf{H}_k^H+\sum_{k=1}^{K}\mathrm{Tr}\left(\mathbf{V}_k\mathbf{V}_k^H\right)\mathbf{I}\right)^{-1}\right)$$

where $\mathbf{V}_k = Net(\theta;h)$ and h denotes the input channels

- Supervised (pre-training) + Unsupervised (further optimization) learning



Learning Scheme

Algorithm 2 Supervised + Unsupervised Learning Algorithm

```
Data preprocessing.
Divide the data set into training set and test set.
for i = 1 : num_epoch do
    Perform training epoch with Huber loss.
end for
Perform training epoch with unsupervised loss for one epoch.
```

Design of Input and Output

- The update of $\mathbf{X}_k$ is

$$\mathbf{X}_k = \left(\sum_{j=1}^{K} \mathrm{Tr}\left(\mathbf{U}_j \mathbf{W}_j \mathbf{U}_j^H\right)(\mathbf{H}\mathbf{H}^H) + \sum_{j=1}^{K} \mathbf{H}\mathbf{H}_j^H \mathbf{U}_j \mathbf{W}_j \mathbf{U}_j^H \mathbf{H}_j \mathbf{H}^H \right)^{-1} \mathbf{H}\mathbf{H}_k^H \mathbf{U}_k \mathbf{W}_k$$

- Input

Input	Dimension
$\mathbf{H}_k$	$2 \times (KN_R \times N_T)$
$\mathbf{H}\mathbf{H}^H$	$2 \times (KN_R \times KN_R)$
$\mathbf{H}\mathbf{H}^H$(exploit symmetry)	$KN_R \times KN_R$

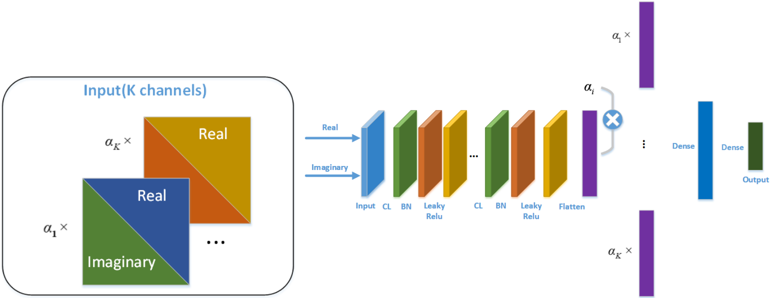
- Output

Output	Dimension
$\mathbf{V}_k$	$2 \times (N_T \times d_k)$
$\mathbf{X}_k$	$2 \times (KN_R \times d_k)$
$\mathbf{U}_k$ and $\mathbf{W}_k$	$2 \times (N_R \times d_k + d_k \times d_k)$
$\mathbf{U}_k$ and $\mathbf{W}_k$ (exploit symmetry)	$2 \times (N_R \times d_k) + d_k \times d_k$

- For regression, usually the smaller the size of output/input, the easier the training

The Proposed Solution To Challenge 2

- For varying α_k's, the network structure should be carefully redesigned



(a) Merge weight into input as channels

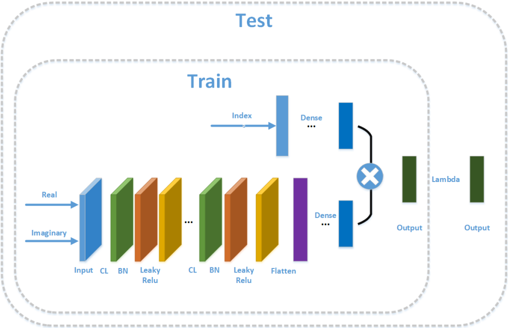
(b) Concatenate weight after conv.

Figure: Methods of merging weights into the network.

- Finally we take $\tilde{\mathbf{H}}\tilde{\mathbf{H}}^H$ as the network input
  where $\tilde{\mathbf{H}}_k = \sqrt{\alpha_k}\mathbf{H}_k$ and $\tilde{\mathbf{H}} \triangleq \left[\tilde{\mathbf{H}}_1^H \quad \tilde{\mathbf{H}}_2^H \quad \ldots \tilde{\mathbf{H}}_K^H\right]^H$

The Proposed Solution To Challenge 3

- The number of streams d_k is also varying but the network output is fixed.
- $\mathbf{U}_k$ and $\mathbf{W}_k$ should contain zeros when $d_k = 1$.
- An indexNet (upper branch) is proposed for end-to-end training

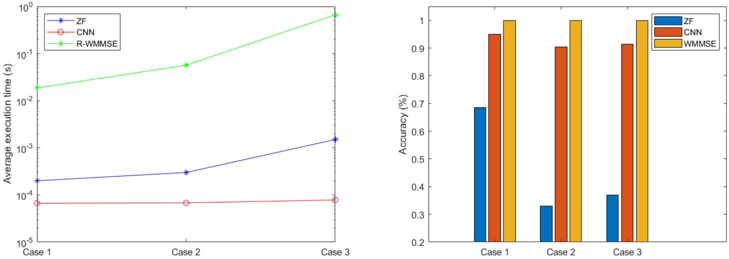


- At the testing stage, 'zero elements' should be assigned with 0 at the last layer

Simulation Setup

- $N_R = 2$
- Case 1: $N_T = 8, K = 2$
- Case 2: $N_T = 8, K = 4$
- Case 3: $N_T = 32, K = 12$
- α_k follows uniform distribution $[0\ 1]$
- $d_k = 1$ (or 2) with probability 0.5

Experiment Results



(a) Average running time (b) Average accuracy

Figure: Comparison of CNN with R-WMMSE and zero-forcing (ZF)

Experiment Results

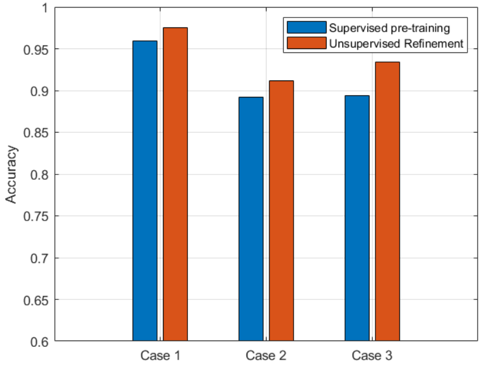


Figure: Unsupervised learning further improves supervised learning

Summary

- We have presented
  - data-driven network optimization
  - learning-based massive beamforming
  - network-level performance prediction
  - reinforcement learning-based MCS scheduling
- Our experiment results show that machine learning is a powerful tool for RRM, which sometimes can replace the role of optimization methods.

Thanks for your attention!