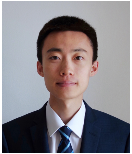
Machine learning for transmit beamforming and power control

Nikos Sidiropoulos

University of Virginia
ECE Department

ML4COM Workshop @ ICC 2018, Kansas City MO, May 24, 2018

Co-Authors



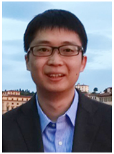
Haoran Sun
UMN

Xiangyi Chen
UMN

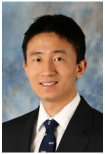
Qingjiang Shi
NUAA

Yunmei Shi
Harbin IT

Aritra Konar
UVA



Xiao Fu
Oregon State



Mingyi hong
UMN

Introduction

- **Transmit Beamforming:** [Farrokhi et al. 1998 (multiuser)], [Bengtsson-Ottersten 2001], [Sidiropoulos et al. 2006 (multicast)]
    1. Exploits CSI at base station (BS) to provide QoS, enhance throughput in multi-antenna wireless systems
    2. Exact CSIT cannot be obtained in practice
    3. Acquiring accurate CSIT is a burden, esp. for FDD, high mobility
    4. Alternative: Robust beamformer design
        - Optimize robust performance metric w.r.t. channel uncertainty

Robust Design: Prior Art

- **Worst-case design:** [Karipidis et al. 2008], [Zheng et al. 2008], [Tajer et al. 2011], [Song et al. 2012], [Huang et al. 2013], [Ma et al. 2017]
  - Downlink channels: bounded perturbations of a set of nominal channel vectors
  - Metric: worst-case QoS w.r.t. all channel perturbations
  - Can result in a very conservative design
- **Outage-based design:** [Xie et al. 2005], [Vorobyov et al. 2008], [Ntranos et al. 2009], [Wang et al. 2014], [He-Wu 2015], [Sohrabi-Davidson 2016]
  - Downlink channels: random vectors from an underlying distribution
  - Metric: QoS exceeds pre-specified threshold with high probability
  - Vary level of conservativeness by changing threshold
  - Approach adopted here

Outage-based Design

- **Prior approaches:**
  - Postulate/fit a model for the underlying probability distribution
  - Use knowledge of distribution to minimize outage probability
  - NP-hard $\rightarrow$ Approximation algorithms, still computationally demanding
- **Our approach:**
  - Knowledge of underlying distribution not required
  - Stochastic approximation - simple, online algorithms for directly minimizing outage
  - Performs remarkably well, hard to analyze

Problem Statement

- **Point-to-point MISO link:**
  - BS equipped with N transmit antennas
  - Received signal at user:

$$y = \mathbf{h}^H \mathbf{w} s + n$$

  - QoS: (normalized) receive SNR $= |\mathbf{w}^H \mathbf{h}|^2$
  - Assumption: Temporal variations of $\mathbf{h} \in \mathbb{C}^N$ are realizations of an underlying distribution
    - Example: Gaussian Mixture Model (GMM) [Ntranos et al. 2009]
    - Interpretation: Each Gaussian kernel corresponds to a different channel state

Problem Formulation

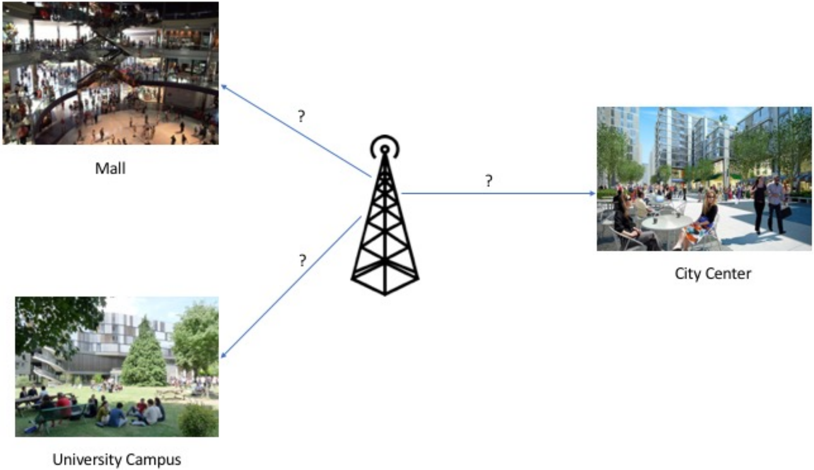
Minimize outage probability subject to power constraints:

$$\min_{\mathbf{w}\in\mathcal{W}} \left\{ F(\mathbf{w}) := \Pr\left(|\mathbf{w}^H \mathbf{h}|^2 < \gamma \right) \right\}$$

- $\mathcal{W} \subset \mathbb{C}^N$: set of power constraints
  - "simple" (easy to project onto), convex, compact
  - Example: per-antenna power constraints, sum-power constraints
- $\gamma \in \mathbb{R}_+$: Outage threshold

Problem Formulation

Equally applicable to single-group multicast beamforming [Ntranos et al. 2009]



Challenges

- Non-convex problem, NP–hard [Ntranos et al. 2009]
- Approximate minimization via simple algorithms?
  - Only for specific cases [Ntranos et al. 2009]
  - Extension to general case requires computing cumbersome integrals
- Who tells you the channel distribution?
  - Not available in practice!
  - Use data-driven approach instead?

Key Idea

Reformulate as stochastic optimization problem

$$\min_{\mathbf{w}\in\mathcal{W}} \left\{ \Pr\left(|\mathbf{w}^H\mathbf{h}|^2 < \gamma \right) = \mathbb{E}_{\mathbf{h}}[\mathbb{I}_{\{|\mathbf{w}^H\mathbf{h}|^2<\gamma\}}] \approx \frac{1}{T}\sum_{t=1}^{T} \mathbb{I}_{\{|\mathbf{w}^H\mathbf{h}_t|^2<\gamma\}} \right\}$$

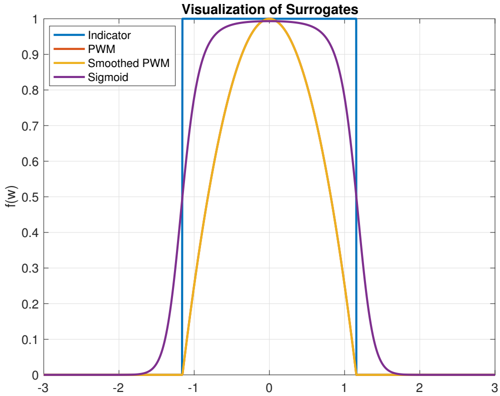
- $\mathbb{I}_{\{f(\mathbf{x})<a\}} = \begin{cases} 1, & \text{if } f(\mathbf{x}) < a \\ 0, & \text{otherwise} \end{cases}$: Indicator function
- Interpretation: minimize total # outages over (“recent”) channel “history” - very reasonable
- Use stochastic approximation [Robbins-Monro 1951], [Shapiro et al. 2009]
  - Given most recent channel realization $\mathbf{h}_t$
  - Update $\mathbf{w}$ to minimize instantaneous cost function $\mathbb{I}_{\{|\mathbf{w}^H\mathbf{h}_t|^2<\gamma\}}$

Stochastic Approximation

- **Benefits**:
  - Knowledge of channel distribution not required!
  - Online implementation
    - Low memory and computational footprint
  - Naturally robust to intermittent/stale feedback from the user
    - All channel vectors are statistically equivalent
    - Feedback requirements are considerably relaxed
  - Can also exploit feedback from "peer" users
    - "Collaborative Filtering/Beamforming"
  - **Well suited for FDD systems**
- Can it work well for our non-convex, NP-hard problem?

Stochastic Approximation

- **Major roadblock:**
  - Indicator function is non-convex, discontinuous
- **Proposed solution:**
  - Approximate indicator function via smooth surrogates



Construction of smooth surrogates

- **Transformation to real domain:**
  - Define $\tilde{\mathbf{w}} := [\Re[\mathbf{w}]^T, \Im[\mathbf{w}]^T]^T \in \mathbb{R}^{2N}$, $\tilde{\mathbf{h}} := [\Re[\mathbf{h}]^T, \Im[\mathbf{h}]^T]^T \in \mathbb{R}^{2N}$
  - Define
$$\tilde{\mathbf{H}} := \begin{bmatrix} \Re[\mathbf{h}] & \Im[\mathbf{h}] \\ \Im[\mathbf{h}] & -\Re[\mathbf{h}] \end{bmatrix} \in \mathbb{R}^{2N \times 2}$$
  - In terms of real variables
    - Indicator function $f(\tilde{\mathbf{w}}; \tilde{\mathbf{h}}) := \mathbb{I}_{\{\|\tilde{\mathbf{H}}^T \tilde{\mathbf{w}}\|_2^2 < \gamma\}}$
    - Constraint set $\tilde{\mathcal{W}} \subset \mathbb{R}^{2N}$

Construction of smooth surrogates

- **Sigmoidal Approximation:**

$$u(\tilde{\mathbf{w}};\tilde{\mathbf{h}}) := \frac{1}{1+\exp\left(\|\tilde{\mathbf{H}}^T\tilde{\mathbf{w}}\|_2^2-\gamma\right)}$$

  - Continuously differentiable

- **Point-wise Max (PWM) Approximation:**

$$v(\tilde{\mathbf{w}};\tilde{\mathbf{h}}) := \max\left\{0, 1-\frac{\|\tilde{\mathbf{H}}^T\tilde{\mathbf{w}}\|_2^2}{\gamma}\right\} = \max_{0\le y\le 1}\left\{y\left(1-\frac{\|\tilde{\mathbf{H}}^T\tilde{\mathbf{w}}\|_2^2}{\gamma}\right)\right\}$$

  - Non-differentiable!
  - Solution: Apply Nesterov's smoothing trick [Nesterov 2005]

Construction of smooth surrogates

- **Smoothed Point-wise Max Approximation:**
  - Define smoothing parameter $\mu \in \mathbb{R}_+$
  - Define $g(\tilde{\mathbf{w}}; \tilde{\mathbf{h}}) := 1 - \dfrac{\|\tilde{\mathbf{H}}^T \tilde{\mathbf{w}}\|_2^2}{\gamma}$
  - Consider the modified PWM function

$$v^{(\mu)}(\tilde{\mathbf{w}}; \tilde{\mathbf{h}}) = \max_{0 \leq y \leq 1} \left\{ y g(\tilde{\mathbf{w}}; \tilde{\mathbf{h}}) - \frac{\mu}{2} y^2 \right\}$$
$$= \begin{cases} 0, & g(\tilde{\mathbf{w}}; \tilde{\mathbf{h}}) < 0 \\ \dfrac{1}{2\mu} \left(g(\tilde{\mathbf{w}}; \tilde{\mathbf{h}}) \right)^2, & 0 \leq g(\tilde{\mathbf{w}}; \tilde{\mathbf{h}}) \leq \mu \\ g(\tilde{\mathbf{w}}; \tilde{\mathbf{h}}) - \dfrac{\mu}{2}, & g(\tilde{\mathbf{w}}; \tilde{\mathbf{h}}) > \mu \end{cases}$$

  - Continuously differentiable!
  - Furthermore,

$$v^{(\mu)}(\tilde{\mathbf{w}}; \tilde{\mathbf{h}}) \leq v(\tilde{\mathbf{w}}; \tilde{\mathbf{h}}) \leq v^{(\mu)}(\tilde{\mathbf{w}}; \tilde{\mathbf{h}}) + \frac{\mu}{2}, \forall\, (\tilde{\mathbf{w}}; \tilde{\mathbf{h}})$$ [Nesterov 2005]

Putting it all together

- **Modified Problem(s):**

$$\min_{\tilde{\mathbf{w}}\in\tilde{\mathcal{W}}}\left\{U(\tilde{\mathbf{w}}) := \mathbb{E}_{\tilde{\mathbf{h}}}[u(\tilde{\mathbf{w}};\tilde{\mathbf{h}})]\right\} \text{ [Sigmoidal Approx.]}$$

$$\min_{\tilde{\mathbf{w}}\in\tilde{\mathcal{W}}}\left\{V^{(\mu)}(\tilde{\mathbf{w}}) := \mathbb{E}_{\tilde{\mathbf{h}}}[v^{(\mu)}(\tilde{\mathbf{w}};\tilde{\mathbf{h}})]\right\} \text{ [Smoothed PWM Approx.]}$$

- Represent both via the problem

$$\min_{\mathbf{x}\in\mathcal{X}}\mathbb{E}_{\boldsymbol{\xi}}[f(\mathbf{x};\boldsymbol{\xi})]$$

  - $\mathcal{X}\subset\mathbb{R}^d$: convex, compact and simple
  - $\boldsymbol{\xi}$: random vector drawn from unknown probability distribution with support set $\boldsymbol{\Xi}\subset\mathbb{R}^d$
  - $f(.;\boldsymbol{\xi})$: non-convex, continuously differentiable
  - Minimize by sequentially processing stream of realizations $\{\boldsymbol{\xi}_t\}_{t=0}^{\infty}$

Online Algorithms

- **Online Gradient Descent (OGD)**
  - Given realization $\boldsymbol{\xi}_t$, define $f_t(\mathbf{x}) := f(\mathbf{x}; \boldsymbol{\xi}_t)$
  - Update:
  $$\mathbf{x}^{(t+1)} = \Pi_{\mathcal{X}}(\mathbf{x}^{(t)} - \alpha_t \nabla f_t(\mathbf{x}^{(t)})), \forall\, t \in \mathbb{N}$$
- **Online Variance Reduced Gradient (OVRG)** [Frostig et al. 2015]
  - Streaming variant of SVRG [Johnson-Zhang 2013]
  - Proceeds in stages
  - At each stage $s \in [S]$, define "centering variable" $\mathbf{y}_s$ from last stage
  - "Anchor" OGD iterates to gradient of $\mathbf{y}_s$
  - $\mathbb{E}_{\boldsymbol{\xi}}[\nabla f(\mathbf{y}_s; \boldsymbol{\xi})]$ is unavailable; form surrogate via mini-batching
  $$\hat{\mathbf{g}}_s := \frac{1}{k_s} \sum_{i \in [k_s]} \nabla f_i(\mathbf{y}_s)$$
  - Update:
  $$\mathbf{x}_s^{(t+1)} = \Pi_{\mathcal{X}}(\mathbf{x}_s^{(t)} - \alpha_s^{(t)}(\nabla f_t(\mathbf{x}_s^{(t)}) - \nabla f_t(\mathbf{y}_s) + \hat{\mathbf{g}}_s)), \forall\, t \in [T]$$
  - Set $\mathbf{y}_{s+1} = \mathbf{x}_s^{(T+1)}$

Convergence?

According to theory:

- **OGD**:
  - (a.s.)convergence to stationary point with diminishing step-size rule [Razaviyayn et al. 2016]
  - Requires $f(;\boldsymbol{\xi})$ to have L Lipschitz continuous gradients
- **OVRG**:
  - Only established for strongly convex with constant step-sizes $f(;\boldsymbol{\xi})$ [Frostig et al. 2015]
  - Extension to non-convex $f(;\boldsymbol{\xi})$ currently an open problem
- To go by the book (or not)?
  - OGD: hard to estimate L; estimates too conservative to work well in practice
  - OVRG: non-trivial to establish convergence
  - Use empirically chosen step-sizes; work well in simulations

Baseline for comparison

- **Alternative approach:**

$$\min_{\mathbf{w}\in\mathcal{W}} \Pr[|\mathbf{w}^H\mathbf{h}|^2 < \gamma] \Longleftrightarrow \max_{\mathbf{w}\in\mathcal{W}} \Pr[|\mathbf{w}^H\mathbf{h}|^2 \geq \gamma]$$

  - Ideally: Maximize lower bound of objective function
  - NP–hard to compute [Ntranos et al. 2009]
  - Construct lower bound using moment information [He-Wu 2015]
    - Entails solving non-trivial, non-convex problem
    - Not suitable for online approximation
  - Instead: Use Markov's inequality to maximize upper bound [Ntranos et al. 2009]

$$\Pr[|\mathbf{w}^H\mathbf{h}|^2 \geq \gamma] \leq \gamma^{-1}\mathbf{w}^H\mathbf{R}\mathbf{w}, \forall\ \mathbf{w} \in \mathcal{W}$$

Baseline for comparison

Online Markov Approximation

$$\max_{\mathbf{w}\in\mathcal{W}} \mathbf{w}^H\mathbf{R}\mathbf{w}$$

- Online solution:
  - Sum-power constraints: Oja's Algorithm [Oja 1982]
    - (a.s.)convergence to optimal solution
  - Per-antenna constraints: Stochastic SUM [Razaviyayn et al. 2016]
    - (a.s.)convergence to stationary point

Simulations

- **Setup:**
  - Algorithms: Sigmoid OGD & OVRG, PWM OGD & OVRG, Online Markov Approximation (OM-App)
  - Step-sizes: Diminishing rule for OGD, constant for OVRG
  - Iteration Number: fix maximum gradient budget for all methods
  - Smoothing parameter for PWM $\mu = 10^{-3}$
  - For OVRG
    - Length of each stage: $T = 1000$
    - Mini-batch sizes:

$$k_s = \begin{cases} 80, & s = 1 \\ 2k_{s-1}, & k_s < 640 \\ 640, & \text{otherwise} \end{cases}$$

  - Constraints: Per-antenna (-6dbW per antenna)
  - Channels: GMM with 4 kernels
    - Equal mixture probabilities
    - Mean of each kernel modeled using different LOS component

Illustrative Example

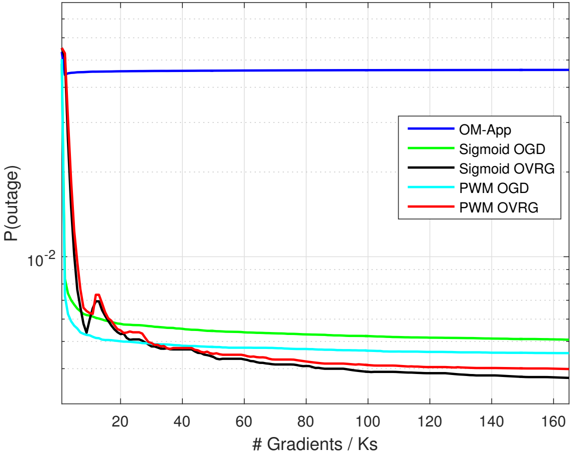
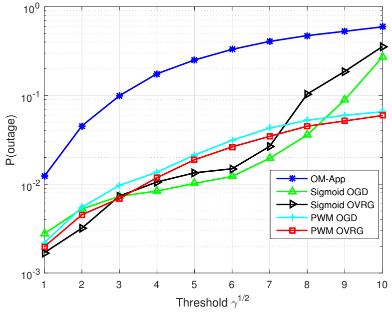
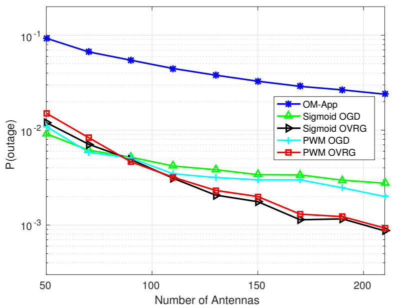


Figure: $N = 100,\ \gamma = 4,\ K_s = 200$

Detailed Results





Intermezzo - take home points

Learning to beamform for minimum outage

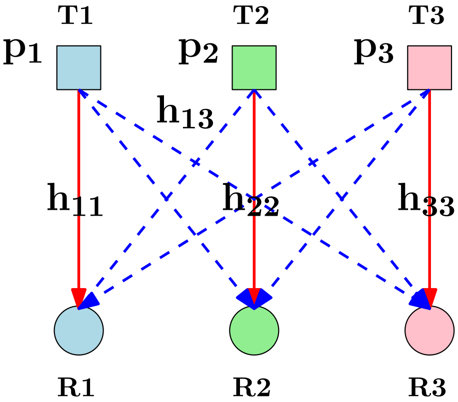
- No prior knowledge of distribution required at BS
- Reformulate as stochastic optimization problem
- Construct smooth surrogate of indicator function
- Use simple stochastic approximation based algorithms based on user feedback
  - Feedback can be intermittent/delayed/stale/from peer users
- Works remarkably well in practice (problem is NP-hard even for known channel distribution!)
- Future work: Extension to general multi-user MIMO, better theoretical understanding of WHY it works that well

Be bold!

Part II: Resource Management for Wireless Networks

Wireless Resource Management

Tx power allocation to optimize throughput.



Example: Formulation

For each receiver k, signal to interference-plus-noise ratio (SINR)

$$\text{SINR} = \frac{|h_{kk}|^2 p_k}{\sum_{j \neq k} |h_{kj}|^2 p_j + \sigma_k^2}$$

- h_{ij}: elements of channel matrix H
- p_k: power allocated to k-th link (optimization variable)
- σ_k^2: noise power at k-th receiver

Example: Formulation

Maximize weighted system throughput:

$$\max_{p} \; f(p;h) = \sum_{k=1}^{K} \alpha_k \log\left(1 + \frac{|h_{kk}|^2 p_k}{\sum_{j\neq k} |h_{kj}|^2 p_j + \sigma_k^2}\right)$$

$$\text{s.t. } 0 \le p_k \le P_{\max}, \forall k = 1, 2, ..., K$$

- α_k: nonnegative weights
- $P_{\max}$: max power allocated to each user
- NP hard problem [Luo-Zhang 2008]
- Lots of iterative algorithms in the literature deal with (generalized versions of) this problem, e.g., SCALE [Papandriopoulos et al 09], Pricing [Shi et al 08], WMMSE [Shi et al 11], BSUM [Hong et al 14]; See [Schmidt et al 13] for comparison of different algorithms

Introduction

- **Proposed Method: Learning to Optimize**

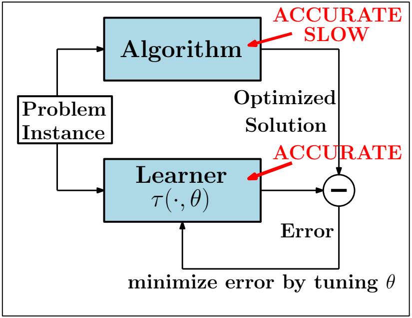


Figure: Training Stage

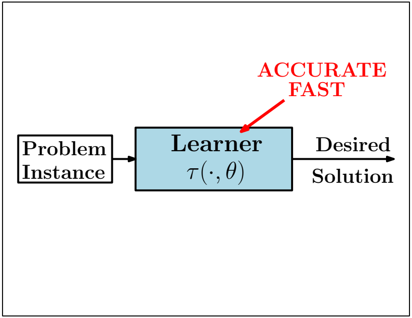


Figure: Testing Stage

Literature Review

- “Unfold” specific iterative algorithm
  - Gregor and LeCun (2010)
    - Iterative soft-thresholding algorithm (ISTA)
  - Gregor and LeCun (2010)
    - Coordinate descent algorithm (CD)
  - Sprechmann et al. (2013)
    - Alternating direction method of multipliers (ADMM)
  - Hershey et al. (2014)
    - Multiplicative updates for non-negative matrix factorization (NMF)
- Drawbacks
  - No theoretical approximation guarantees
  - Can we use fewer layers to approximate more iterations?

Can we learn the entire algorithm?

Proposed Method

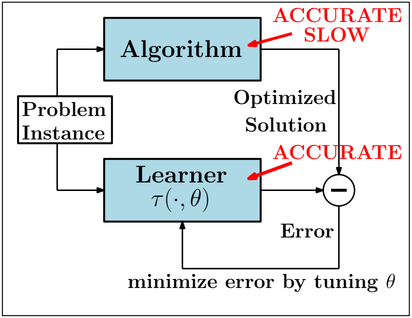


Figure: Training Stage

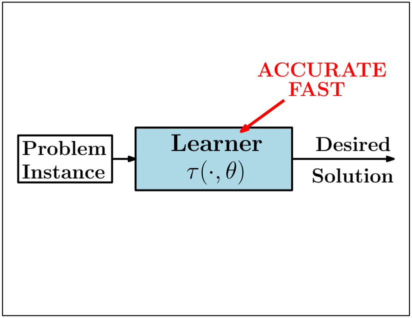


Figure: Testing Stage

- Given lots of (h, x^*) pairs, learn the nonlinear "mapping" $h \to x^*$
- **Questions:**
  - How to choose "Learner"?
  - What kinds of algorithms can we accurately learn?
  - What's the major benefit of such an approach?

Deep Neural Network

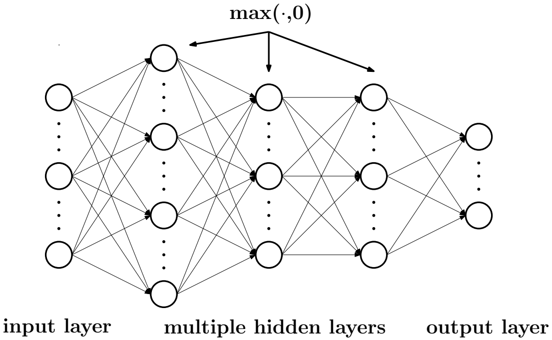


Figure: deep neural network

Deep Neural Network

- Difficult to train [Glorot and Bengio (2010)]
  - The vanishing gradient problem
  - Traditional gradient descent not work
- Recent advances
  - New initialization methods [Hinton et al. (2012)]
  - New training algorithms: ADAM [Kingma and Ba (2014)], RMSprop [Hinton et al. (2012)], ...
  - New hardwares: CPU clusters, GPUs, TPUs, ...
- DNN is more powerful than the traditional NN [Telgarsky (2016)]
- To achieve the same accuracy as shallow neural network, DNN can be exponentially faster in testing stage [Mhaskar et al. (2016)]

Example: Approximate iterative algorithm

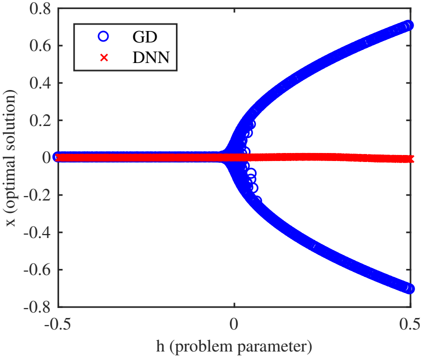


Figure: The learned model (the red line) with random initialization

- Learning the mapping $h \to x_T$?
- **Issue:** Cannot learn the behavior of the algorithm well
- Three layers DNN; 50 K training samples

Example: Approximate iterative algorithm

- **Reason: Non-convexity** results in multiple local solutions

Example: Approximate iterative algorithm

- **Reason:** **Non-convexity** results in multiple local solutions
- **Solution:** **Add init as features:** Learn the mapping $(x_0, h) \rightarrow x_T$

Example: Approximate iterative algorithm

- **Reason: Non-convexity** results in multiple local solutions
- **Solution: Add init as features:** Learn the mapping $(x_0, h) \to x_T$

Example: Approximate iterative algorithm

- **Reason: Non-convexity** results in multiple local solutions
- **Solution: Add init as features:** Learn the mapping $(x_0, h) \rightarrow x_T$
- The model learned in this way

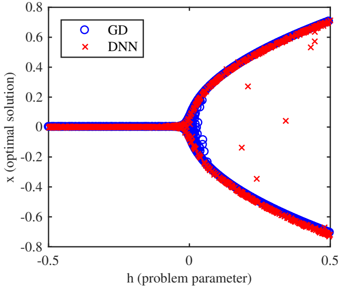


Figure: The learned model (the red line)

Universal approximation theorem for iterative algorithm

Theorem 1 [Sun et al 17]

- Given a T iteration algorithm whose input output relationship is:

$$x_T = g_T(g_{T-1}(\ldots g_1(g_0(x_0, h), h)\ldots, h), h) \triangleq G_T(x_0, h) \quad (1)$$

where h is problem parameter; x_0 is initialization; $g_k(x_{k-1}; h)$ is a continuous mapping, representing the algorithm at kth iteration

Universal approximation theorem for iterative algorithm

Theorem 1 [Sun et al 17]

- Given a T iteration algorithm whose input output relationship is:

$$x_T = g_T(g_{T-1}(\ldots g_1(g_0(x_0, h), h) \ldots, h), h) \triangleq G_T(x_0, h) \quad (1)$$

  where h is problem parameter; x_0 is initialization; $g_k(x_{k-1}; h)$ is a continuous mapping, representing the algorithm at kth iteration

- Then for any $\epsilon > 0$, there exist a three-layer neural network $NET_{N(\epsilon)}(x_0, h)$ with $N(\epsilon)$ nodes in the hidden layer such that

$$\sup_{(x_0,h)\in X_0\times H} \|NET_{N(\epsilon)}(x_0, h) - G_T(x_0, h)\| \leq \epsilon. \quad (2)$$

  where H and initialization X_0 are any compact sets.

Universal approximation theorem for iterative algorithm

Theorem 1 [Sun et al 17]

- Given a T iteration algorithm whose input output relationship is:

$$x_T = g_T(g_{T-1}(\dots g_1(g_0(x_0, h), h) \dots, h), h) \triangleq G_T(x_0, h) \quad (1)$$

  where h is problem parameter; x_0 is initialization; $g_k(x_{k-1}; h)$ is a continuous mapping, representing the algorithm at kth iteration
- Then for any $\epsilon > 0$, there exist a three-layer neural network $NET_{N(\epsilon)}(x_0, h)$ with $N(\epsilon)$ nodes in the hidden layer such that

$$\sup_{(x_0,h)\in X_0\times H} \|NET_{N(\epsilon)}(x_0, h) - G_T(x_0, h)\| \leq \epsilon. \quad (2)$$

  where H and initialization X_0 are any compact sets.
- Extension of the classical result [Cybenko (1989)]

Universal approximation theorem for iterative algorithm

- **Key point:** It is possible to learn an iterative algorithm, represented by the mapping $(x_0, h) \to x_T$
- Assumptions on the algorithm:
  - For iterative algorithm:

    $$x_{k+1} = g_k(x_k, h)$$

    where $h \in H$ is the problem parameter; $x_k, x_{k+1} \in X$ are the optimization variables.
  - The function g_k is a continuous mapping
  - X and H are compact sets

Case Study: Resource Management for Wireless Networks

Maximize weighted system throughput:

$$\max_{p} \; f(p;h) = \sum_{k=1}^{K} \alpha_k \log\left(1 + \frac{|h_{kk}|^2 p_k}{\sum_{j \neq k} |h_{kj}|^2 p_j + \sigma_k^2}\right)$$

$$\text{s.t. } 0 \leq p_k \leq P_{\max}, \forall k = 1, 2, ..., K$$

- α_k: nonnegative weights
- $P_{\max}$: max power allocated to each user

Case Study: Existing Methods

- We will attempt to learn a popular method called Weighted Minimum Mean Square Error (WMMSE) [Shi et al. (2011)]
- Transform the problem into one with three sets of variables (v, u, w)
- Optimize in a coordinate descent manner

Case Study: Existing Methods

input: $H, \{\sigma_k\}, P_{max}$, **output:** $\{p_k\}$

Initialize v_k^0 such that $0 \leq v_k^0 \leq \sqrt{P_{max}}, \forall\ k;$

Initialize $u_k^0 = \frac{|h_{kk}|v_k^0}{\sum_{j=1}^{K} |h_{kj}|(v_j^0)^2 + \sigma_k^2}$, $w_k^0 = \frac{1}{1 - u_k^0 |h_{kk}| v_k^0}, \forall\ k;$

repeat

Update v_k: $v_k^t = \left[\frac{\alpha_k w_k^{t-1} u_k^{t-1} |h_{kk}|}{\sum_{j=1}^{K} \alpha_j w_j^{t-1} (u_j^{t-1})^2 |h_{jk}|^2} \right]_0^{\sqrt{P_{\max}}}, \forall\ k;$

Update u_k: $u_k^t = \frac{|h_{kk}|v_k^t}{\sum_{j=1}^{K} |h_{kj}|^2 (v_j^t)^2 + \sigma_k^2}, \forall\ k;$

Update w_k: $w_k^t = \frac{1}{1 - u_k^t |h_{kk}| v_k^t}, \forall\ k;$

until $\left| \sum_{j=1}^{K} \log\left(w_j^t\right) - \sum_{j=1}^{K} \log\left(w_j^{t-1}\right) \right| \leq \epsilon;$

output $p_k = (v_k)^2, \ \forall\ k;$

Figure: Pseudo code of WMMSE for the scalar IC.

Case Study: Proposed Approach

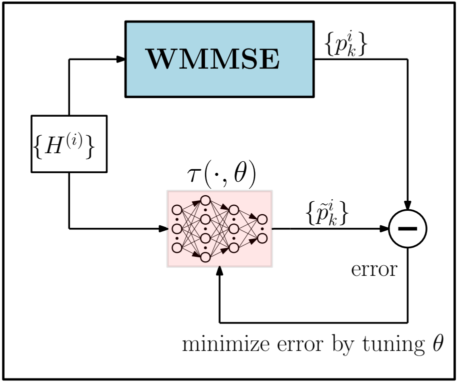


Figure: Training Stage

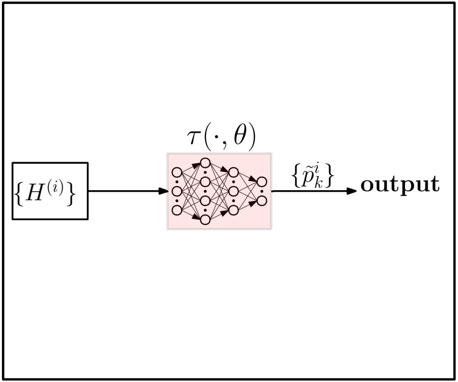


Figure: Testing Stage

Approximation of WMMSE by deep neural networks

Theorem 2 [Sun et al 17]

Suppose WMMSE is initialized with $p_k = P_{\max}, \forall k$. Define

$$\mathcal{H} := \left\{ h \mid H_{\min} \leq |h_{jk}| \leq H_{\max}, \forall j,k \quad \sum_{i=1}^{K} v_i^t(h) \geq P_{\min} > 0, \forall t \right\}.$$

Given $\epsilon > 0$, there exists a neural network $NET(h)$ consisting of

$$O\left(T^2 \log\left(\max\left(K, P_{\max}, H_{\max}, \frac{1}{\sigma}, \frac{1}{H_{\min}}, \frac{1}{P_{\min}}\right)\right) + T\log\left(\frac{1}{\epsilon}\right)\right)$$ layers

$$O\left(T^2 K^2 \log\left(\max\left(K, P_{\max}, H_{\max}, \frac{1}{\sigma}, \frac{1}{H_{\min}}, \frac{1}{P_{\min}}\right)\right) + TK^2\log\left(\frac{1}{\epsilon}\right)\right)$$

ReLUs and Binary units,

such that $\max_{h \in \mathcal{H}} \max_i |(p_i^T(h))^2 - NET(h)_i| \leq \epsilon$

IMAC - Data Generation

For problem with N base stations and total K users

- Channels are generated according to 3GPP standards
- Fix other values, i.e., $P_{max} = 1, \sigma_k = 1$
- Given tuple $(\{\tilde{H}^{(i)}\}, P_{\max}, \{\sigma_k\})$, run WMMSE get $\{p_k^i\}$, $\forall i, k$
- 10^6 training samples $(H^{(i)}, \{p_k^i\}), \forall i \in \mathcal{T}$
- 10^4 testing samples $H^{(i)}, \forall i \in \mathcal{V}$

IMAC - Training Stage

Training Deep Neural Network

- We pick a three-hidden-layer DNN with 200-80-80 neurons
- Implemented by Python 3.6.0 with TensorFlow 1.0.0
- Training using two Nvidia K20 GPUs
- Training is based on optimizing the loss function

$$\min_{\theta} \sum_{i \in \mathcal{T}} \|\tau(H^{(i)}, \theta) - \{p_k^i\}\|^2$$

IMAC - Testing Stage

Testing Deep Neural Network

- DNN appraoch: implemented by Python
- WMMSE algorithm: implemented in C
- Testing only using CPU
- Objective function

$$f = \sum_{k=1}^{K} \log \left(1 + \frac{|h_{kk}|^2 p_k}{\sum_{j \neq k} |h_{kj}|^2 p_j + \sigma_k^2} \right)$$

- Evaluate ratio of the per-testing sample sum-rates

$$\frac{f(H^{(i)}, \{\tilde{p}_k^i\}, \{\sigma_k\}) \Rightarrow \text{DNN}}{f(H^{(i)}, \{p_k^i\}, \{\sigma_k\}) \Rightarrow \text{WMMSE}}, \forall i$$

IMAC - Larger Problem

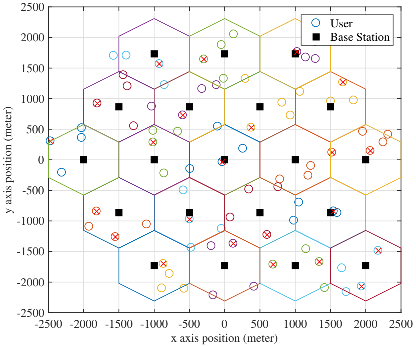


Figure: radius = 500 m, MD = 0 m

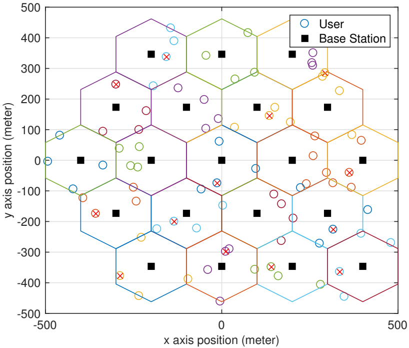
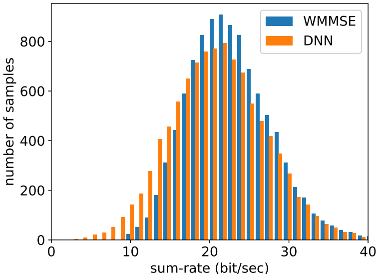


Figure: radius = 100 m, MD = 20 m

Figure: IMAC: $N = 20,\ K = 80$

IMAC - Results



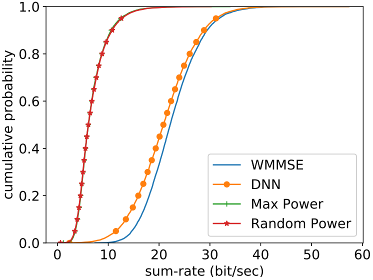


Figure: IMAC: $N = 20$, $K = 80$, radius = 100m

IMAC - Larger Problem

Table: Relative CPU Time and Sum-Rate for IMAC

network structure	training samples	sum-rate r=500m	sum-rate r=100m	computational time r=500m	computational time r=100m
200-200-200	2 million	**98.44%**	88.46%	0.7%	0.4%
200-200-200	1 million	97.03%	**89.59%**	0.7%	0.4%
200-80-80	2 millions	95.58%	87.44%	0.6%	0.5%
200-80-80	1 million	95.39%	86.70%	0.6%	0.3%
200-80-80	0.5 million	95.39%	85.35%	0.6%	0.3%
200-80-80	0.1 million	94.71%	81.28%	0.6%	0.3%

Key observations:

- Increase training samples helps
- Increase number of neurons helps

Problem Setup - VDSL channel

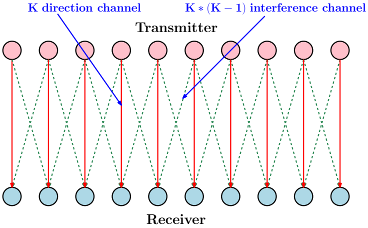


Figure: cast as a 28-user IC problem

- Data collected by France Telecom R&D [Karipidis et al. (2005)]
- Measured lengths: **75** meters, **150** meters, and **300** meters
- far-end crosstalk (**FEXT**) vs. near-end crosstalk (**NEXT**)
- Total of 6955 channel measurements

VDSL - Procedure & Results

- 6955 *real data* $= 5000$ *validation* $+ 1955$ *testing*
- $50,000$ *training*: **computer-generated** following *validation* statistics
- **Same** training and testing procedure

Table: Sum-Rate and Computational Performance for Measured VDSL Data

(length, type)	sum-rate DNN/WMMSE	computational time DNN/WMMSE(C)
(75, FEXT)	99.96%	42.18%
(150, FEXT)	99.43%	50.98%
(300, FEXT)	99.58%	57.78%
(75, NEXT)	99.85%	3.16%
(150, NEXT)	98.31%	7.14%
(300, NEXT)	94.14%	5.52%

Recap, take home, road forward

- Two very (NP-)hard problems: BMF for min outage; max sum rate power control for multiuser interference channel
- Boldly using ML (staples): SGD, DNN, ...
- Some things we can prove, design currently an art, not difficult to tune
- As engineers, we have to appreciate opportunities, understand why
- Updates
  - Lee *et al*, Deep Power Control: Transmit Power Control Scheme Based on Convolutional Neural Network, IEEE Communications Letters (2018) extend our approach, using sum rate for training in second stage (can improve upon WMMSE);
  - de Kerret *et al*, Decentralized Deep Scheduling for Interference Channels, arXiv:1711.00625 (2017) consider user scheduling for the IC using multiple collaboratively trained DNNs

Thank You!

Paper: Y. Shi, A. Konar, N. D. Sidiropoulos, *et al.*, "Learning to Beamform for Minimum Outage", in review.

Paper: H. Sun, X. Chen, Q. Shi, M. Hong, X. Fu, and N. D. Sidiropoulos, "Learning to Optimize: Training Deep Neural Networks for Wireless Resource Management", `https://arxiv.org/abs/1705.09412`

Code: `https://github.com/Haoran-S/TSP-DNN`

References

- F. Rashid-Farrokhi, L. Tassiulas, and K. J. R. Liu, “Joint optimal power control and beamforming in wireless networks using antenna arrays,” *IEEE Trans. Commun.*, vol. 46, no. 10, pp. 1313-1324, Oct. 1998.
- M. Bengtsson, and B. Ottersten, “Optimal and suboptimal transmit beamforming,” in *Handbook of Antennas in Wireless Communications*, L. C. Godara, Ed. Boca Raton, FL, USA: CRC Press, Aug. 2001, ch. 18.
- M. J. Lopez, “Multiplexing, scheduling, and multicasting strategies for antenna arrays in wireless networks,” Ph.D. dissertation, Elect. Eng. and Comp. Sci. Dept., MIT, Cambridge, MA, 2002.
- N. D. Sidiropoulos, T. Davidson, and Z.-Q. Luo, “Transmit beamforming for physical-layer multicasting,” *IEEE Trans. Signal Process.*, vol. 54, no. 6, pp. 2239–2251, June 2006.
- E. Karipidis, N. D. Sidiropoulos, and Z.-Q. Luo, “Quality of service and max-min-fair transmit beamforming to multiple co-channel multicast groups,” *IEEE Trans. Signal Process.*, vol. 56, no. 3, pp. 1268–1279, Mar. 2008.
- G. Zheng, K.-K. Wong, and T.-S. Ng, “Robust linear MIMO in the downlink: A worst-case optimization with ellipsoidal uncertainty regions,” *EURASIP J. Adv. Signal Process.*, vol. 2008, pp. 1-15, June 2008.

References

- A. Tajer, N. Prasad, and X. Wang, "Robust linear precoder design for multi-cell downlink transmission," *IEEE Trans. Signal Process.*, vol. 59, no. 1, pp. 235-251, Jan. 2011.
- E. Song, Q. Shi, M. Sanjabi, R.-Y. Sun, and Z.-Q. Luo, "Robust SINR constrained MISO downlink beamforming: When is semidefinite programming relaxation tight?," *EURASIP J. Wireless Commun. Netw.*, vol. 1, no. 1, pp. 1-11, Dec. 2012.
- Y. Huang, D. P. Palomar, and S. Zhang, "Lorentz-positive maps and quadratic matrix inequalities with applications to robust MISO transmit beamforming," *IEEE Trans. Signal Process.*, vol. 61, no. 5, pp. 1121-1130, Mar. 2013.
- W.-K. Ma, J. Pan, A. M.-C. So, and T.-H. Chang, "Unraveling the rank-one solution mystery of robust MISO downlink transmit optimization: A verifiable sufficient condition via a new duality result", *IEEE Trans. Signal Process.*, vol. 65, no. 7, pp. 1909-1924, Apr. 2017.
- Y. Xie, C. N. Georghiades, and A. Arapostathis, "Minimum outage probability transmission with imperfect feedback for MISO fading channels," *IEEE Trans. Wireless Commun.*, vol. 4, no. 3, pp. 1084–1091, May 2005.

References

- S. A. Vorobyov, H. Chen, and A. B. Gershman, “On the relationship between robust minimum variance beamformmers with probabilistic and worst-case distortionless response constraints,” *IEEE Trans. Signal Process.*, vol. 56, pp. 5719–5724, Nov. 2008.
- V. Ntranos, N. D. Sidiropoulos, and L. Tassiulas, “On multicast beamforming for minimum outage”, *IEEE Trans. Wireless Commun.*, vol. 8, no. 6, pp. 3172–3181, June 2009.
- K.-Y. Wang, A. M.-C. So, T.-H. Chang, W.-K. Ma, and C.-Y. Chi, “Outage constrained robust transmit optimization for multiuser MISO downlinks: Tractable approximations by conic optimization,” *IEEE Trans. Signal Process.*, vol. 62, no. 21, pp. 5690-5705, Sep. 2014.
- X. He and Y.-C. Wu, “Tight probabilistic SINR constrained beamforming under channel uncertainties,” *IEEE Trans. Signal Process.*, vol. 63, no. 13, pp. 3490–3505, July 2015.
- F. Sohrabi and T. N. Davidson, “Coordinate update algorithms for robust power loading for the MU-MISO downlink with outage constraints,” *IEEE Trans. Signal Process.*, vol. 64, no. 11, pp. 2761-2773, June 2016.
- H. Robbins and S. Monro, “A stochastic approximation method,” *Ann. Math. Statist.*, vol. 22, no. 3, pp. 400-407, Sep. 1951.

References

- A. Shapiro, D. Dentcheva, and A. Ruszczynski, *Lectures on stochastic programming: Modeling and theory*, SIAM, 2009.
- Y. Nesterov, “Smooth minimization of non-smooth functions,” *Math. Program.*, vol. 103, no. 1, pp 127–152, May 2005.
- R. Frostig, R. Ge, S. M. Kakade, and A. Sidford, “Competing with the empirical risk minimizer in a single pass”, in *Proc. Conf. Learn. Theory*, Paris, France, July 2015, pp. 728–763.
- R. Johnson, and T. Zhang, “Accelerating stochastic gradient descent using predictive variance reduction,” *Adv. Neural Info. Process Syst.*, Lake Tahoe, CA, USA, Dec. 2013, pp. 315–323.
- E. Oja, “Simplified neuron model as a principal component analyzer”, *J. Math. Biology*, vol. 15, no. 3, pp. 263–273, Nov. 1982.
- M. Razaviyayn, M. Sanjabi, and Z.-Q. Luo, “A stochastic successive minimization method for nonsmooth nonconvex optimization with applications to transceiver design in wireless communication networks,” *Math. Prog.*, vol. 157, no. 2, pp. 515-545, June 2016.

References

- Gregor, Karol, and Yann LeCun. “Learning fast approximations of sparse coding.” Proceedings of the 27th International Conference on Machine Learning. 2010.
- Daubechies, Ingrid, Michel Defrise, and Christine De Mol. “An iterative thresholding algorithm for linear inverse problems with a sparsity constraint.” Communications on pure and applied mathematics 57.11 (2004): 1413-1457.
- Sprechmann, Pablo, et al. “Supervised sparse analysis and synthesis operators.” Advances in Neural Information Processing Systems. 2013.
- Hershey, John R., Jonathan Le Roux, and Felix Weninger. “Deep unfolding: Model-based inspiration of novel deep architectures.” arXiv preprint arXiv:1409.2574 (2014).
- Andrychowicz, Marcin, et al. “Learning to learn by gradient descent by gradient descent.” Advances in Neural Information Processing Systems. 2016.
- Li, Ke, and Jitendra Malik. “Learning to optimize.” arXiv preprint arXiv:1606.01885 (2016).
- Liang, Shiyu, and R. Srikant. ”Why Deep Neural Networks for Function Approximation?.” (2016).

References

- Hinton, Geoffrey, et al. "Deep neural networks for acoustic modeling in speech recognition: The shared views of four research groups." IEEE Signal Processing Magazine 29.6 (2012): 82-97.
- Krizhevsky, Alex, Ilya Sutskever, and Geoffrey E. Hinton. "Imagenet classification with deep convolutional neural networks." Advances in neural information processing systems. 2012.
- Liu, Bing. "Sentiment analysis and opinion mining." Synthesis lectures on human language technologies 5.1 (2012): 1-167.
- Cybenko, George. "Approximation by superpositions of a sigmoidal function." Mathematics of Control, Signals, and Systems (MCSS) 2.4 (1989): 303-314.
- Sun, Haoran, et al. "Learning to optimize: Training deep neural networks for wireless resource management." arXiv preprint arXiv: 1705.09412 (2017).
- Glorot, Xavier, and Yoshua Bengio. "Understanding the difficulty of training deep feedforward neural networks." Aistats. Vol. 9. 2010.

References

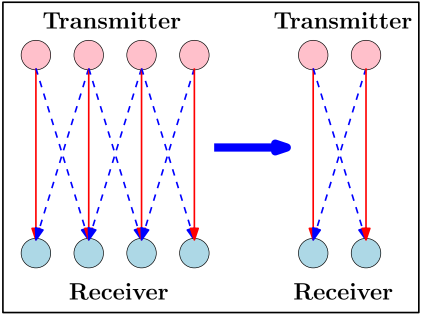
- Kingma, Diederik, and Jimmy Ba. "Adam: A method for stochastic optimization." arXiv preprint arXiv:1412.6980 (2014).
- Mhaskar, Hrushikesh, Qianli Liao, and Tomaso Poggio. "Learning functions: When is deep better than shallow." arXiv preprint arXiv:1603.00988 (2016).
- Luo, Zhi-Quan, and Shuzhong Zhang. "Dynamic spectrum management: Complexity and duality." IEEE Journal of Selected Topics in Signal Processing 2.1 (2008): 57-73.
- Shi, Qingjiang, et al. "An iteratively weighted MMSE approach to distributed sum-utility maximization for a MIMO interfering broadcast channel." IEEE Transactions on Signal Processing 59.9 (2011): 4331-4340.
- Gjendemsj, Anders, et al. "Binary power control for sum rate maximization over multiple interfering links." IEEE Transactions on Wireless Communications (2008).
- Hinton, Geoffrey, NiRsh Srivastava, and Kevin Swersky. "Neural Networks for Machine Learning Lecture 6a Overview of mini-batch gradient descent." (2012).

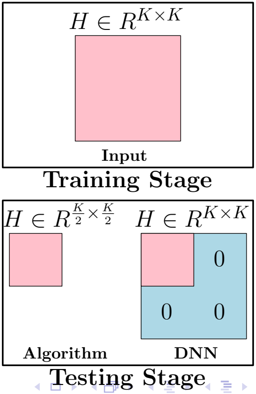
Interference Channel (IC) - Generalization

Issues:

- Same number of users for both training and testing
- In practice, what if K in testing is different from training?

Half-user simulation setup:





Interference Channel (IC) - Generalization

- Half-user results

Table: Relative CPU Time and Sum-Rate for Gaussian IC half-user

	sum-rate		computational time	
# of users (K)	full-user	**half-user**	full-user	**half-user**
10	97.92%	**99.22%**	0.32%	**0.96%**
20	92.65%	**92.78%**	0.16%	**0.48%**
30	85.66%	**87.77%**	0.12%	**0.37%**