Big Data Signal Processing for Communication Networks

Zijie Zheng, Peking University
Zhu Han, University of Houston
Geoffrey Ye Li, Georgia Institute of Technology

IEEE GLOBECOM 2017
Content

- Overview of Big Data
- Learning Methods
- Commercial Systems
- Large Scale Optimization
- Game Theory based Approaches
Content

- Overview of Big Data
- Learning Methods
- Commercial Systems
- Large Scale Optimization
- Game Theory based Approaches
Big Data 3V Characteristics

- Exponentially increasing data size
- Heterogeneous types of data
  - Possible to analyze ALL available data
- Need to calculate fast
- Key dimensions
  - Volume, Velocity and Variety
  - Variability and Veracity

Big Data Business
However

• Nobody knows exactly how to handle big data

• We zoom to
  □ Signal processing techniques
  □ Networking applications
Big Data Techniques

Data Mining

Big Data Storage

Big Data Sampling

Real Time Processing

Machine Learning Methods
Big Data Applications

Manufacturing

Healthcare

Media

Education

Financial Services

Sports
Communication and Networking

• Fast cloud computing vs. slow communication
• Local, fog vs. cloud
Internet of Things
Smart Grid

- In 2009, “American Recovery and Reinvestment Act”
  - $3.4 billion for SG investment grant program
  - $615 million for SG demonstration program
  - A combined investment of $8 billion in SG capabilities
Big Data Ecosystem
Big Data Landscape
Content

- Overview of Big Data
- Learning Methods
- Large Scale Optimization
- Game Theory based Approaches
- Commercial Systems
Learning Methods

- Classical Machine Learning
- Bayesian Nonparametric learning
- Deep Learning
Classical Machine Learning

“Computers: learn without being explicitly programmed”

Types:

- **Supervised learning:**
  - Example inputs (features) and their desired outputs (labels)
  - Goal: learn a general rule that maps inputs to outputs
  - SVM, neural networks, etc.
- **Unsupervised learning:**
  - No labels
  - Find structure in its input
  - Goal: discover hidden patterns in data
  - Clustering, K-means, etc.
- **Reinforcement learning:**
  - Interact with a dynamic environment (such as driving a vehicle or playing a game against an opponent)
  - Feedback in terms of rewards and punishments as it navigates its problem space
  - Active learning, Q-learning, etc.
Supervised Learning: SVM

• Distance from sample $x_i$ to the separator: $r$

• **Support vectors**: samples closest to the hyperplane

• **Margin $\rho$**: the distance between support vectors

• **Objective**: maximize the margin $\rho$

\[
\begin{align*}
    r & = \frac{y_i(w^T x_i + b)}{|w|} = \frac{1}{|w|} \\
    \rho & = \frac{2}{|w|} \\

    y = -1: w^T x_i + b \leq -\frac{\rho}{2} \\
    y = 1: w^T x_i + b \geq \frac{\rho}{2}
\end{align*}
\]
Supervised Learning: SVM Applications

- The best performers for a number of classification tasks ranging from text to genomic data.

- Complex data types beyond feature vectors (e.g. graphs, sequences, relational data) by designing kernel functions for such data.

- Extend to a number of tasks such as regression [Vapnik et al. ’97], principal component analysis [Schölkopf et al. ’99], etc.
Unsupervised Learning: K-Means

- Ask user how many clusters they’d like. *(e.g. k=5)*
- Randomly guess k cluster center locations
- Each data point: find out which center it’s closest to
- Each center: find the centroid of the points it owns
- Change center
- Repeat until terminated
Unsupervised Learning: K-Means Applications

- Data mining
- Acoustic data in speech understanding to convert waveforms into one of k categories (known as Vector Quantization or Image Segmentation)
- Also used for choosing color palettes on old fashioned graphical display devices and Image Quantization
Reinforcement Learning: Q-Learning

Today
Study & Discussion
Have Fun
Fencing Event
Submit Paper

1 -100 100
Reinforcement Learning: Q-Learning

$$Q_{t+1}(s, a) = (1 - \alpha_t)Q_t(s, a) + \alpha_t [R_t(s, a) + \beta \max_b Q_t(s', b)]$$

Learning Rate  Discount Factor

Study  Fencing

Fun  Study

$s$  $a$  $b^*$  $s'$  Fun
Reinforcement Learning: Results

- 317182 Games
- 1 million steps
- Learning rate: 0.9999954
- Discount rate: 0.8
- Epsilon: 0.1
Reinforcement Learning: Q-Learning Applications

- Online learning and recommendations systems
- Games: Alphago, Texas Holdem
- Reinforcement Learning in Vehicular Networks
Learning Methods

- Classical Machine Learning
- Bayesian Nonparametric learning
- Deep Learning


Nam Tuan Nguyen, Kae Won Choi, Lingyang Song, and Zhu Han, “Roommates: An Unsupervised Indoor Peer Discovery Approach for LTE D2D Communications,” to appear IEEE Transactions on Vehicular Technology.
Bayesian Nonparametric Learning

- For multi-dimension data
  - Model selection: The number of clusters
  - The hidden process created the observations
  - Latent parameters of the process

- Classic parametric methods (e.g. K-Means)
  - Need to estimate the number of clusters
  - Can have huge performance loss with poor model
  - Cannot scale well

- Nonparametric Bayesian Learning
  - **Nonparametric**: Number of clusters (or classes) can grow as more data are observed and need not to be known as a priori.
  - **Bayesian Inference**: Use Bayesian rule to infer about the latent variables.
Bayesian rule

\[
p(\mu|\text{Observation}) = \frac{p(\text{Observation}|\mu)p(\mu)}{p(\text{Observation})}
\]

- \(\mu\) : contain information such as how many clusters, and which sample belongs to which cluster

- \(\mu\) : nonparametric

Sample the posterior distribution \(P(\mu|\text{Observations})\), and get values of the parameter \(\mu\).
Bayesian Nonparametric Learning: Example

- A Beta distribution as prior:
  \[
  \text{Beta}(\mu|a, b) = \frac{\Gamma(a + b)}{\Gamma(a)\Gamma(b)}\mu^{a-1}(1 - \mu)^{b-1}
  \]
  - Example: a=2, b=2 (head and tail prob. are equal)

- A Binomial distribution as conjugate likelihood:
  \[
  \text{Bin}(m|N, \mu) = \binom{N}{m}\mu^m(1 - \mu)^{N-m}
  \]
  - One trial (N=1) and the result is one head (m=1)

- Lead to the Posterior:
  \[
  p(\mu|m, l, a, b) = \frac{\Gamma(m + a + l + b)}{\Gamma(m + a)\Gamma(l + b)}\mu^{m+a-1}(1 - \mu)^{l+b-1}
  \]
  - Update of parameters given the observations
  - Higher probability of head
Dirichlet Process

- An extension of the Beta distribution to multiple dimensions

\[ \text{Dir}(\alpha_1, \ldots, \alpha_K) = \frac{1}{B(\alpha)} \prod_{i=1}^{K} \pi_i^{\alpha_i-1} \]

\[ B(\alpha) = \frac{\prod_{i=1}^{K} \Gamma(\alpha_i)}{\Gamma(\sum_{i=1}^{K} \alpha_i)} \]

- **K**: number of clusters
- **\pi_i**: weight with marginal distribution
- **\alpha_i**: prior

- **Dirichlet Process**: \( G \sim DP(\alpha, H) \), if for every finite measurable partition \( A_1, \ldots, A_K \) of \( \Theta \)

\[ (G(A_1) \cdots G(A_K)) \sim \text{Dir}(\alpha H(A_1), \ldots, \alpha H(A_K)) \]

- \( H(\cdot) \): the mean of the DP
- \( \alpha \): strength of the prior

\[ E[G(A)] = H(A) \]
Bayesian Nonparametric Update

• Have \( t \) observation \( x_1, \ldots, x_t \) Define \( n_i = \# \{ i : x_i \in A_i \} \)

• The posterior distribution on \( \Theta \)

\[
(G(A_1) \cdots G(A_K)) \mid x_1 \cdots x_t \sim \text{Dir}(\alpha H(A_1) + n_1, \ldots, \alpha H(A_K) + n_K)
\]

• The posterior Dirichlet process

\[
G \mid x_1 \cdots x_t \sim DP(\alpha + t, \frac{\alpha}{\alpha + t} H + \frac{t}{\alpha + t} n)
\]

- Small number of observation \( t \), the prior dominates
- \( t \) increases, the prior has less and less impact
- \( \alpha \): control the balance between the impact of prior and trials
- Learn and combine any distributions
Infinite Gaussian Mixture Model

Infinite number of classes

Observations: follow a distribution such as Gaussian.

Indicators created according to multinomial distribution.

Dirichlet process

\[ \pi_k | \alpha \sim \text{Beta}(1, \alpha) \]

\[ \pi_k = \frac{\pi_k}{\prod_{l=1}^{k-1} (1 - \pi_l)}, \ k \to \infty \]
Chinese Restaurant Process

- **Goal:**
  \[ P(z_i = k | \tilde{Z}_{-i}, \alpha, \tilde{\theta}, \tilde{H}, \tilde{X}) = P(z_i = k | \tilde{Z}_{-i}, \alpha, \tilde{\theta}_k, \tilde{H}, \tilde{x}_i) \sim P(\tilde{x}_i | \tilde{\theta}_k) P(z_i = k | \tilde{Z}_{-i}, \alpha) , \]

  \[ \tilde{Z}_{-i} \text{ : set of all other labels except the current one, } i^{th} \]

- Probability assigned to a represented class
  \[ P(z_i = k | \tilde{Z}_{-i}, \alpha) = \frac{n_{k,-i}}{\alpha + N - 1} \]

- Probability assigned to an unrepresented class
  \[ P(z_i \neq z_j, \forall j \neq i | z_{-i}, \alpha) = 1 - \frac{\sum_{j=1}^{K} n_{j,-i}}{\alpha + N - 1} \]

\[ n_{k,-i} \text{ : the number of observations in the same class, } k, \text{ excluding the current one, } i^{th} \]
Inference model: Posterior Distributions

- Given the prior and the likelihood, the posterior:
  - Probability of assigning to a unrepresented cluster:
    \[ P \left( z_i \neq j, \forall j \neq i | Z_{-i}, X; \alpha, \bar{H} \right) \approx \frac{\alpha}{\alpha + N - 1} t_{v_0 - D + 1} \left( \frac{\bar{H}_0 \Lambda_0 (\kappa_0 + 1)}{(\kappa_0 (v_0 - D + 1))} \right). \] (1) \( t \) is the student-t distribution
  - Probability of assigning to a represented cluster:
    \[ P(z_i = k | Z_{-i}, X; \alpha, \bar{H}) \approx \frac{n_{k,-i}}{\alpha + N - 1} t_{v_n - D + 1} \left( \frac{\bar{\mu}_n \Lambda_n (\kappa_n + 1)}{(\kappa_n (v_n - D + 1))} \right). \] (2)

Intuitive: Provide a stochastic gradient!

\[
\begin{align*}
\bar{\mu}_n &= \frac{\kappa_0}{\kappa_0 + N} \bar{\mu}_0 + \frac{N}{\kappa_0 + N} \bar{X}, \\
\kappa_n &= \kappa_0 + N, \ \nu_n = \nu_0 + N, \\
\Lambda_n &= \Lambda_0 + S + \frac{\kappa_0 n}{\kappa_0 + N} (\bar{X} - \bar{\mu}_0)(\bar{X} - \bar{\mu}_0)^T, \\
\bar{X} &= \frac{(\bar{x}_1 + \bar{x}_2 + \cdots + \bar{x}_N)}{N},
\end{align*}
\]
Inference model: Gibbs sampler

1. Start with random indicator for each observation.
2. Remove the current $i^{th}$ observation from its cluster.
3. Update the indicator $z_i$ according to (1) and (2) given all the other indicators.
4. Converge?
   - Yes: STOP
   - No: Repeat steps 1-3.

Note: The process continues until convergence is achieved.
Chinese Restaurant Process: Results

- Two Gaussian distributed clusters with KL divergence (KLD) 4.5

  - Intuition why it works so well
    - Not the boundary or threshold. But clustering so that each cluster looks more like the distribution (Gaussian)
    - No prior information on probabilities
Bayesian Nonparametric Learning: Applications

- Smart grid
- Security for wireless devices
- Location based services
Smart Pricing for Maximizing Profit

• The profit = sum of utility bill – cost to buy power
  □ Different shape of loads cost different
  □ Incentive using pricing to change the loads
  □ The cost reduction is greater than loss of bills
Load Profiling

- From smart meter data, try to tell users’ usage behaviors
  - CEO, 1%, audience here
  - Worker, middle class, myself
  - Homeless, slave, Ph.D. students
Load Profiling Results

• Utility company wants to know benchmark distributions
  □ Nonparametric: do not know how many benchmarks
  □ Bayesian: posterior distribution might be time varying
  □ Scale: Daily, weekday, weekend, monthly, yearly
Security for Wireless Devices

• Prime User emulation (PEU) attack detection
  □ In Cognitive radio, a malicious node can pretend to be a Primary User (PU) to keep the network resources (bandwidth) for his own use
  □ Collect device dependent fingerprints and classify the fingerprints
    • The Carrier Frequency Difference (CFD)
      □ Defined as the difference between the carrier frequency of the ideal signal and that of the transmitted signal.
      □ Depends on the oscillator within each device.
    • The Phase Shift Difference (PSD)
      □ Using QPSK modulation technique.
      □ Transmitter amplifiers for I-phase and Q-phase might be different.
Location Based Services

- **Major tasks to enable LBS:**
  - Localize.
  - Estimate dwelling time
  - Prediction: Where to go next?

- **What’s given:**
  - Mobile devices are in indoor environments
  - WiFi scans

- **Goals:**
  - Identifying revisited location
  - Automatically profiling new location
  - Online sampling to reduce the complexity
  - Predicting the next possible locations
Location Based Services: Experimental Results

• Dataset:
  - 4 weeks long
  - 5 phones
Location Based Services: Experimental Results

- Future Location Prediction

Gibbs sampler output

- Observations inside a rectangle are from the same location.
- Observations inside an eclipse are classified as one cluster according to the Gibbs Sampler.

Two states

Same room
Learning Methods

- Classical Machine Learning
- Bayesian Nonparametric learning
- Deep Learning


Deep Learning: Basic Idea

- Add Hidden Layers in Neural Networks

\[ f(\mathbf{w}^T \mathbf{x} + b) \]

Linear transformation followed by non-linear rectification

- More parameters
- More non-linear parts
Deep Learning: Motivations

• Classic Methods
  - Do not have a lot of data, or
  - Training data have categorical features
  - A more explainable model
  - A high run-time speed

• Deep Learning
  - A lot of training data of the same or related domain
  - Improve Domain Adaptation
  - Much slower
  - Appropriate scaling and normalization have to be done
Why Now?

- Big annotated datasets become available:
  - ImageNet:
  - Google Video:
  - Mechanical Turk
  - Crowdsourcing

- GPU processing power

- Better stochastic gradient descents:
  - AdaGrad, AdamGrad, RMSProp
Typical Deep Neural Networks

- Convolutional Neural Networks (CNNs)
- Recurrent Neural Networks (RNNs)
- Deep Belief Networks
Convolutional Neural Networks (CNNs)

- **INPUT IMAGE**
- **CONVOLUTION (LEARNED)**
- **NON-LINEAR RECTIFIER**
- **POOLING**
- **FEATURE MAPS**

Work well in image processing!

Recurrent Neural Networks (RNNs)

- Produce an output at each time step and have recurrent connections between hidden units
  - Long Short-Term Memory

- Unconstrained handwriting recognition (Graves et al., 2009),
- Speech recognition (Graves et al., 2013; Graves and Jaitly, 2014)
- Handwriting generation (Graves, 2013),
- machine translation (Sutskever et al., 2014) Image captioning (Kiros et al., 2014;Vinyals et al., 2014; Xu et al., 2015)
- Parsing (Vinyals et al., 2014a).
Deep Belief Networks

• Each link associates with a probability
• Parametric

\[
E(v, h; \theta) = - \sum_{i,j} W_{i,j} v_i h_j - \sum_i b_i v_i - \sum_j a_j h_j
\]
\(\theta = \{W, a, b\}\) model parameters.

• Applied in clustering

## Comparison

<table>
<thead>
<tr>
<th></th>
<th>Similarities</th>
<th>Differences</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Convolutional Neural Networks</strong></td>
<td>1. Multiple Layers</td>
<td>1. More suitable for data with grid structures</td>
</tr>
<tr>
<td></td>
<td>2. Use Back-propagation Algorithm for training</td>
<td>2. Much fewer parameters</td>
</tr>
<tr>
<td></td>
<td>3. Can be combined together to create more powerful networks</td>
<td>3. Very efficient training with GPUs</td>
</tr>
<tr>
<td><strong>Recurrent Networks</strong></td>
<td></td>
<td>1. Having memory of past (suitable for tasks like speech recognition)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>2. Not able to take big input such as images or videos</td>
</tr>
<tr>
<td><strong>Deep Belief Networks</strong></td>
<td></td>
<td>1. Generative model (can generate realistic looking data after initializing at random variable)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>2. Used much less due to inefficiency</td>
</tr>
</tbody>
</table>
Applications

- Computer Vision
  - Object detection
  - Add & repair missing details in high-resolution photo
- Speech Recognition
  - Enable a text-to-speech system that is almost indistinguishable from human voice
  - Compose classical music that likely to be created by a human
- Natural Language Processing
- Online Recommendation Systems
- Automatic Driving
  - Google Driverless Car
- Wireless Resource Allocation
- Vehicular Communication Network
- Smart Grid
Summary

• Classical Machine Learning
• Bayesian Nonparametric learning
  □ Non-parametric
  □ Smart grid & Location Based Services
• Deep Learning
  □ Convolutional Neural Networks
  □ Recurrent Neural Networks
  □ Deep Belief Networks
Content

• Overview of Big Data
• Learning Methods
• Commercial Systems
• Large Scale Optimization
• Game Theory based Approaches
Commercial Systems

- TensorFlow
- MapReduce
TensorFlow

• A deep learning library open-sourced by Google

• Originally developed by the Google Brain Team within Google's Machine Intelligence research organization for the purposes of conducting machine learning and deep neural networks research, but the system is general enough to be applicable in a wide variety of other domains as well

• Provide primitives for defining functions on tensors and automatically computing their derivatives.

https://www.tensorflow.org/
Comparison on Deep Learning Libraries

- TensorFlow
  - Google
- Caffe
  - UC-berkeley
- MXNet
- Theano
  - Université de Montréal
- Torch
  - Facebook
- Microsoft CNTK
- Neon
  - Intel
Describe mathematical computation with a directed graph of nodes & edges

- Nodes in the graph represent mathematical operations
- Edges describe the i/o relationships between nodes
- Data edges carry dynamically-sized multidimensional data arrays, or tensors

How Does It Work?
How to Use It?

• Programming with APIs
  • Define features and labels, loss functions, select algorithm, set iteration time...

```python
import numpy as np
import matplotlib.pyplot as plt

# number of random points
num_puntos = 1000
# list of random points (pair of x and y)
conjunto_puntos = []

for i in xrange(num_puntos):
    x1 = np.random.normal(0.0, 0.55)
    y1 = x1*0.1 + 0.3 + np.random.normal(0.0, 0.03)
    conjunto_puntos.append([x1,y1])

x_data = [v[0] for v in conjunto_puntos]
y_data = [v[1] for v in conjunto_puntos]

# plot the those random points in 2D
plt.plot(x_data,y_data,'go', label='Original data')
plt.legend()
plt.show()
```
Example: Linear Regression

1 iteration

8 iterations

16 iterations
Commercial Systems

- TensorFlow
- MapReduce
MapReduce: Basic Idea

- Parallel programming model meant for large clusters
  - User implements Map() and Reduce()

- Parallel computing framework
  - Libraries take care of EVERYTHING else
    - Parallelization
    - Fault Tolerance
    - Data Distribution
    - Load Balancing

- Useful model for many practical tasks (large data)

http://hadoop.apache.org/
Processing

Very big data → Map: 
- Accepts input key/value pair 
- Emits intermediate key/value pair 

Map → Partitioning Function → Reduce: 
- Accepts intermediate key/value* pair 
- Emits output key/value pair 

Reduce → Result
Architecture
Three Steps to Use

• Indicate
  □ Input/output files
  □ M: number of map tasks
  □ R: number of reduce tasks
  □ W: number of machines

• Write map and reduce functions

• Submit the job
Example: Word Count

- **Map()**
  - Input `<filename, file text>`
  - Parses file and emits `<word, count>` pairs
    - eg. `<"hello", 1>`

- **Reduce()**
  - Sums values for the same key and emits `<word, TotalCount>`
    - eg. `<"hello", (3 5 2 7)> => <"hello", 17>`
More Distributed Systems

• Hadoop: Realize MapReduce
  □ Offline big data processing
  □ Web searching
  □ Parallel computing

• Spark: Working on Internal Storage
  □ Online big data processing, fast input and output
  □ Repeated operation for the same data

• Storm: Data Flow
  □ Realtime data streaming
Summary

• TensorFlow
  □ Deep Learning Library
• MapReduce
  □ Parallel Computing System
  □ Hadoop & Spark & Storm
Content

- Overview of Big Data
- Learning Methods
- Commercial Systems
- Large Scale Optimization
- Game Theory based Approaches
Large Scale Optimization

• Multi-block Optimization
• Compressive Sensing
• Sublinear Algorithms
Multi-block Optimization

- Generally, optimization algorithms for solving problems of such huge sizes should satisfy:
  - **Simple Sub-problem**: Each of their computational steps must be simple and easy
  - **Parallel Implementable**: Algorithm is implementable in distributed and/or parallel manner
  - **Fast Convergence**: A high-quality solution can be found using a small number of iterations

\[
\begin{align*}
\min_{x_1 \in \mathcal{X}_1, x_2 \in \mathcal{X}_2} f_1(x_1) + f_2(x_2) & \quad \text{s.t.} \quad A_1 x_1 + A_2 x_2 = c, \\
\min_{x \in \mathcal{X}} f(x) \quad \text{s.t.} \quad Ax = c & \quad \Rightarrow \quad x_i^{k+1} = \arg \min_{x_i} \mathcal{L}_\mu(x_i, \{x_j^k\}_{j \neq i}, \lambda^k) + \frac{1}{2} \|x_i - x_i^k\|_B^2, \\
& \quad \varepsilon = o(1/k)
\end{align*}
\]

Optimization Preliminary: Dual Ascent Methods

- Consider an optimization problem of the form

\[
\min_{x \in \mathcal{X}} f(x) \quad \text{s.t.} \quad Ax = c
\]

### The Lagrangian

- **Dual function**
  \[ g(\lambda) = \inf_x \mathcal{L}(x, \lambda) \]

- **Dual Problem**
  \[ \max_\lambda g(\lambda) \quad \text{Lower bound} \]

- **Optimal solution**
  \[ x^* = \arg \min_x \mathcal{L}(x, \lambda^*) \]

- **Dual Ascent (Gradient Descent)**
  \[
  \begin{align*}
  x^{k+1} &= \arg \min_x \mathcal{L}(x, \lambda^k), \\
  \lambda^{k+1} &= \lambda^k + \rho^k(Ax^{k+1} - c)
  \end{align*}
  \]

**KKT (Karush–Kuhn–Tucker) Condition – Strong Convexity**

Require an appropriate step size $\rho$ and assumptions of strong convexity of the objective function $f$. 

Optimization Preliminary: Method of Multipliers

• Introduce an augmentation $\|Ax - c\|_2^2$ to the Lagrangian:

• The Augmented Lagrangian: $\mathcal{L}_\rho(x, \lambda) = f(x) + \lambda^T(Ax - c) + \frac{\rho}{2}\|Ax - c\|_2^2$,

• Method of Multipliers: \[
\begin{align*}
x^{k+1} &= \text{arg min}_x \mathcal{L}_\rho(x, \lambda^k), \\
\lambda^{k+1} &= \lambda^k + \rho(Ax^{k+1} - c),
\end{align*}
\]

• Pros:
  - Stable, robust and fast compare with the dual ascent method.
  - No need to tune the parameter $\rho$ during each iteration.

• Cons:
  - Difficult to decouple and parallelize due to the augmentation $\|Ax - c\|_2^2$
Two Block Optimization: ADMM

- **ADMM: Alternating Direction Method of Multipliers**

- The general form of problem that ADMM can solve is expressed as:

  \[
  \min_{x_1 \in \mathcal{X}_1, x_2 \in \mathcal{X}_2} f_1(x_1) + f_2(x_2) \quad \text{s.t.} \quad A_1 x_1 + A_2 x_2 = c. \tag{1}
  \]

- The augmented Lagrangian for (1) is

  \[
  \mathcal{L}_\rho(x_1, x_2, \lambda) = f_1(x_1) + f_2(x_2) + \lambda^T (A_1 x_1 + A_2 x_2 - c) \\
  + \frac{\rho}{2} \| A_1 x_1 + A_2 x_2 - c \|^2_2,
  \]

- A Gauss-Seidel iterations of \( x_1 \) and \( x_2 \) as follows

  \[
  \begin{align*}
  x_{1}^{k+1} &= \arg \min_{x_1} \mathcal{L}_\rho(x_1, x_2^k, \lambda^k), \\
  x_{2}^{k+1} &= \arg \min_{x_2} \mathcal{L}_\rho(x_1^{k+1}, x_2, \lambda^k), \\
  \lambda^{k+1} &= \lambda^k + \rho(A_1 x_1^{k+1} + A_2 x_2^{k+1} - c).
  \end{align*}
  \]

- Global convergence for convex optimization with a convergence rate \( o\left(\frac{1}{k}\right) \)

---


Gauss-Seidel Multi-Block ADMM (Sequential)

• Consider the following convex optimization problem

\[
\min_{x_1, x_2, \ldots, x_N} f(x) = f_1(x_1) + \ldots + f_N(x_N),
\]

s.t. \( A_i x_i + \ldots + A_N x_N = c, \)

\[ x_i \in \mathcal{X}_i, \quad i = 1, \ldots, N. \tag{2} \]

• The augmented Lagrangian for (2):

\[
\mathcal{L}_\rho(\{x_i\}^N_{i=1}, \lambda) = \sum_{i=1}^{N} f_i(x_i) + \lambda^T (\sum_{i=1}^{N} A_i x_i - c) + \frac{\rho}{2} \| \sum_{i=1}^{N} A_i x_i - c \|^2_2
\]

• A Gauss-Seidel Multi-block ADMM:

\[
\begin{align*}
& x_i = \arg \min_{x_i} \mathcal{L}_\rho(\{x_j^{k+1}\}^N_{j<i}, x_i, \{x_j^{k}\}^N_{j>i}, \lambda^k), \\
& \lambda^{k+1} = \lambda^k + \rho(\sum_{i=1}^{N} A_i x_i^{k+1} - c).
\end{align*}
\]

(Block Coordinate Descent…)

Proximal Jacobian Multi-Block ADMM (Parallel)

- Recall the augmented Lagrangian:
  \[ L_p(\{x_i\}_{i=1}^N, \lambda) = \sum_{i=1}^N f_i(x_i) + \lambda^T (\sum_{i=1}^N A_i x_i - c) + \frac{\rho}{2} \| \sum_{i=1}^N A_i x_i - c \|^2 \]

- A proximal term is added to the augmented Lagrangian, and the update of \( x_i \) is performed concurrently:
  \[
  \begin{aligned}
  x_i^{k+1} &= \arg \min_{x_i} L_p(x_i, \{x_j^k\}_{j \neq i}, \lambda^k) + \frac{1}{2} \|x_i - x_i^k\|_P^2, \\
  \lambda^{k+1} &= \lambda^k + \gamma \rho (\sum_{i=1}^N A_i x_i^{k+1} - c),
  \end{aligned}
  \forall i = 1, \ldots, N.
  \]
  where \( \|x_i\|_P^2 = x_i^T P_i x_i \) for some symmetric and positive semi-definite matrix \( P_i \geq 0 \).

- The involvement of the proximal term
  - Make subproblem of \( x_i \) strictly or strongly convex
  - Ensure the convergence
  - Easier to solve

Additional Issues

- Indecomposable
  \[ f(x) = f(x_1, x_2, \ldots, x_N) \neq f_1(x_1) + f_2(x_2) + \ldots f_N(x_N) \]

  Decompose to Consensus Problem
  \[ f(x) = f(x_1, x_2, \ldots, x_N) \neq f_1(x_1, x_1^1, \ldots, x_N^1) + f_2(x_2^2, x_2, \ldots) + \ldots f_N(x_N^N, x_2^N, \ldots, x_N) \]
  s.t.  \( x_i = x_i^j \)

- Non-Convexity
  - Relax to convex problems
  - BSUM(Block Successive Upper Bound Minimization)

- Mixed-Integer
  - Branch and Bound (\( x_1 \) not integer => \( \lfloor x_1 \rfloor \) & \( \lfloor x_1 \rfloor + 1 \) )


Summary of Methods

• For a non-convex, mixed-integer, indecomposable, large-scale optimization…

<table>
<thead>
<tr>
<th>Issues</th>
<th>Tricks</th>
</tr>
</thead>
<tbody>
<tr>
<td>Indecomposable</td>
<td>Consensus problem</td>
</tr>
<tr>
<td>Non-convexity</td>
<td>Relaxation, BSUM</td>
</tr>
<tr>
<td>Mixed-integer</td>
<td>Branch and Bound</td>
</tr>
<tr>
<td>Deconvergence in Parallel Update</td>
<td>Proximal</td>
</tr>
<tr>
<td>Slow in Sequential Update</td>
<td>Augmented Lagrangian</td>
</tr>
</tbody>
</table>

• Before those…
  • Simulations & Proofs case by case
  • Experienced Adoptions
    Multi-block algorithms have their own conditions in mathematics except for convexity.
Multi-Block Optimization: Applications

- Security Constrained Optimal Power Flow
- Data Offloading in Software Defined Networks
• The anticipated smart grid data deluge:
  - North American power grid: generate 4.15 TB data per day
  - 61.8 million smart meters deployed in the U.S. by the end of 2013
  - Every one million users produce 27.3TB per year
Blackout

• Increasing integration between cyber operations and physical infrastructures for generation, transmission, and distribution control

• The security and reliability are not guaranteed
Contingency Analysis

- Assume line 1-2 is disconnected.
- Generators A and B cannot change productions quickly.
- The flows over other lines would increase.
- Trigger cascading failure.
Security Constrained Optimal Power Flow (SCOPF)

• **SCOPF**: Minimizing the cost of system operation while satisfying a set of postulated contingency constraints.

\[
\begin{align*}
\min_{\{x^c\}_0^C;\{u^c\}_0^C} & \quad f^0(u^0) \quad \text{scheduling objective} \\
\text{s.t.} & \quad g^0(x^0, u^0) = 0, \text{ power flow equations} \\
& \quad h^0(x^0, u^0) \leq 0, \text{ operating limits for base case} \\
& \quad g^c(x^c, u^c) = 0, \text{ power flow equations} \\
& \quad h^c(x^c, u^c) \leq 0, \text{ operating limits for contingency } k \\
& \quad \|u^0 - u^c\|^2 \leq \Delta_c, c = 1, \ldots, C, \text{ security constrains}
\end{align*}
\]

• **Challenges:**
  - Number of constraints is prohibitive.
  - How to find the best operating point with a scalable algorithm?


DC SCOPF

• ‘N-1' contingency, corrective setting:

\[
\begin{align*}
\min_{\{\theta^c\}^C_{c=0}, \{P_{g,c}\}^C_{c=0}} & \quad \sum_{i \in G} f_i^g(P_i^{g,0}) \\
\text{subject to} & \quad B_{bus}^0 \theta^0 + P_{d,0} - A_{g,0}^0 P_{g,0} = 0, \\
& \quad B_{bus}^c \theta^c + P_{d,c} - A_{g,c}^c P_{g,c} = 0, \\
& \quad |B_i^0 \theta^0| - F_{max} \leq 0, \\
& \quad |B_i^c \theta^c| - F_{max} \leq 0, \\
& \quad P_{g,0}^0 \leq P_{g,0} \leq P_{g,0}^0, \\
& \quad P_{g,c}^c \leq P_{g,c} \leq P_{g,c}^c, \\
& \quad |P_{g,0}^0 - P_{g,c}^c| \leq \Delta_c, \\
& \quad i \in \mathcal{G}, \quad c = 1, \ldots, C,
\end{align*}
\]

where $B_{bus}$ and $B_f$ can be modified from the bus admittance matrix $Y_{bus}$. $A_{g,c}^c$ is the generator connection connection matrix.
A Distributed Approach by ADMM

- Introduce a slack variable $p^c$ to rewrite $|P^{g,0} - P^{g,c}| \leq \Delta_c$ as

\[
P^{g,0} - P^{g,c} + p^c = \Delta_c
\]

\[
0 \leq p^c \leq 2\Delta_c, \quad c = 1, \ldots, C.
\] (3)

- The partial scaled augmented Lagrangian associated with (3) can be calculated as follows:

\[
\mathcal{L}_p(\{P^{g,c}\}_{c=0}^C; \{p^c\}_{c=1}^C; \{\mu^c\}_{c=1}^C)
= \sum_{i \in G} f_i^g(P^{g,i,0}) + \sum_{c=1}^C \frac{\rho^c}{2} \|P^{g,0} - P^{g,c} + p^c - \Delta_c + \mu^c\|_2^2.
\]

- Iterate till convergence
  - 1. Update $\{P^{g,0}\}$
  - 2. Update $\{P^{g,0}, p^c\}$
  - 3. Update dual variable $\mu^c$
Distributed Implementation

- On multi-core machine
- High performance computer cluster using MPI (message passing interface)
- On cloud using Hadoop or Apache Spark
Evaluation setup: Modified data of IEEE 57 bus, IEEE 118 bus and IEEE 300 bus generated by MATPOWER
Multi-Block ADMM: Applications

- Security Constrained Optimal Power Flow
- Data Offloading in Software Defined Networks
Mobile Data Offloading

- Mobile data offloading: Offload traffic from cellular networks to alternate wireless technologies.
- Software defined network (SDN) at the edge: Dynamically route the traffic in a mobile network.

Problem Formulation

• Utility of base stations: \( \sum_{b=1}^{B} U_b(x_b) \)
• Cost of access points: \( \sum_{a=1}^{A} L_a(y_a) \)
• Total revenue: \( \sum_{b=1}^{B} U_b(x_b) - \sum_{a=1}^{A} L_a(y_a) \)

• Equivalent revenue maximization problem:

\[
\begin{align*}
\min_{\{x_1, \ldots, x_B\}, \{y_1, \ldots, y_A\}} & \quad \sum_{a=1}^{A} L_a(y_a) - \sum_{b=1}^{B} U_b(x_b), & \text{Service revenue} \\
\text{s.t} & \quad \sum_{b=1}^{B} y_{ab} \leq C_a, & \forall a, \text{ Capacity constraint} \\
& \quad x_{ba} = y_{ab}. & \forall a, b, \text{ Consensus}
\end{align*}
\]
Proximal Jacobian ADMM

- Iterative gather-scatter scheme (Map-Reduce)
- Signaling: $p_{ab}^k = (y_{ab}^k - \frac{\lambda_{ab}^k}{\rho})$, $q_{ba}^k = (x_{ba}^k + \frac{\lambda_{ab}^k}{\rho})$
Numerical Results

Evaluation setup: $B = 5$ base stations and $A = \{5, 10\}$ access points. $Ca = 10$Mbps
Large Scale Optimization

- Multi-block Optimization
- Compressive Sensing
- Sublinear Algorithms
Sparse Signal Recovering

- Sparse $x$
- Traditional Method: Detect all $x$
- Random linear projection
- Dimension reduction from $x$ to $y$
  - $M > K\log(N/K)$
- Recovery algorithm for ill-posed problem

LASSO

- **LASSO**: Least absolute shrinkage and selection operator (R. Tibshirani, 1996)

- **Sparsity: L₀-norm Problem**

  \[
  \text{arg min}_{x \in \mathbb{R}^N} \| \Phi x - y \|_2
  \quad \text{s.t.} \quad \| x \|_0 \leq k
  \]

- **Sparsity Relaxation: L₁-norm Problem, LASSO**

  \[
  \text{arg min}_{x \in \mathbb{R}^N} \frac{1}{2} \| \Phi x - y \|_2^2 + \lambda \| x \|_1
  \]

- For a vector x

  p-norm: \( \| x \|_p = (|x_1|_p + |x_2|_p + \ldots + |x_N|_p)^{\frac{1}{p}} \)

  L₀-norm: number of non-zero elements
  L₁-norm: sum of absolute values
  L₂-norm: sqrt sum of squares
  Lₘ-norm: max (x)

Algorithms

- Classic solvers and omitted solvers (ADMM & BSUM)
- Other algorithms
  - Shrinkage Operate
  - Prox-linear Algorithms
  - Dual Algorithms
  - Bregman Method
  - Homotopy Algorithm and Parametric Quadratic Programming
  - Continuation, Varying Stepsizes and Line Search
  - Greedy Algorithms
    - Greedy Pursuit Algorithms
    - Iterative Support Detection
    - Hard Thresholding

http://www.math.ucla.edu/~wotaoyin/summer2013/lectures.html
Application: Compressive Spectrum Sensing

- Cognitive Radio Motivation:
  - Most of the licensed spectrum is not used by the licensed users

- How Cognitive Radio Work
  - Secondary (unlicensed) users detect the spectrum holes (unoccupied spectrum) and utilize the spectrum at the absence of the primary (licensed) users

- Advantage of Cognitive Radio
  - Improve radio spectrum utilization

- Key Enabler
  - Spectrum sensing (possibly wideband)

\[
\arg \min_{x \in \mathbb{R}^N} \frac{1}{2} \| \Phi x - y \|_2^2 + \lambda \| x \|_1
\]


Large Scale Optimization

- Multi-block Optimization
- Compressive Sensing
- Sublinear Algorithms
Sublinear Algorithms: Motivation & Basic Concept

Faster than linear algorithms

- $N=$ size of data, we want $o(n)$, not $O(n)$
- Sublinear Time
  - Queries
  - Samples
- Sublinear Space
  - Data Stream
  - Sketching
- Deterministic Algorithms

Quality of approximation

Resources
- number of queries
- running time
Sublinear Algorithms: Example

- Consider a tall skyscraper building and you do not know the total number of this building.
- Now you want to test how high a cat can fly.
- When you throw a cat out of window from floor 1, 2, ... n, the cat will survive;
- When you throw a cat out of window from floor n+1, n+2,..., the cat will die.
- The question to develop an efficient method to determine n given that you have 2 cats.
Sublinear Algorithms: Deterministic Algorithm

- Test each floor? 1, 2, 3, 4, 5…
  - This will lead a linear algorithm $O(n)$.

- Double floors every time? 1, 2, 4, 8, 16…
  - This will lead to $O(\log n)$, but you need $O(\log n)$ cats.

- The Two-Cat Algorithm $O(\sqrt{n})$

```
1: i = -1, l = 1
2: for the first cat is still alive do
3:   i++;
4:   test floor $l = l + i$;
5:     $l = l - i$;
6: for the second cat is still alive do
7:   l++;
8: test floor $l$;
9: Output $l$;
```

1, 3, 6, 10, 15, 21, 28,…
Future Challenges

• For large-scale communication networks:
  • Tradeoff between error & computing complexity
  • Linear or NP problem (not NP-Complete/NP-Hard) may be already hard to solve…
  • Combination with multi-block optimization

Slides from Dr. Ronnit Rubinfeld's website http://people.csail.mit.edu/ronitt/sublinear.html
Slides from Dr. Dana Ron's website http://www.eng.tau.ac.il/~danar/talks.html
Summary

• Multi-block Optimization
  - Convergence, Distributed, & Parallel
  - Smart Grid & Mobile Data Offloading

• Compressive Sensing
  - Sparsity & LASSO
  - Cognitive Spectrum Sensing

• Sublinear Algorithms
  - Sublinear Convergence
Content

- Overview of Big Data
- Learning Methods
- Commercial Systems
- Large Scale Optimization
- Game Theory based Approaches
Game Theory for Big Data Processing

- Introduction of Game Theory
- Two Specific Games & Applications
Introduction

• **John von Neuman** (1903-1957) co-authored, Theory of Games and Economic Behavior, with Oskar Morgenstern in 1940s, establishing game theory as a field.

• **John Nash** (1928-2015) developed a key concept of game theory (Nash equilibrium) which initiated many subsequent results and studies.

• Since 1970s, game-theoretic methods have come to dominate microeconomic theory and other fields.

• **Nobel Prizes**
  
  • Nobel prize in Economic Sciences 1994 awarded to Nash, Harsanyi (Bayesian games) and Selten (subgame perfect equilibrium).
  
  • 2005, Auman and Schelling got the Nobel prize for having enhanced our understanding of cooperation and conflict through game theory.
  
  • 2007, Leonid Hurwicz, Eric Maskin and Roger Myerson won Nobel Prize for having laid the foundations of mechanism design theory.
  
  • 2012, Alvin Elliot Roth, and Lloyd Shapley won Nobel Prize for the theory of stable allocations and the practice of market design.
  
  • 2014, Jean Tirole, won Nobel Prize for the analysis of market power and regulation.
Introduction

- Game theory - mathematical models and techniques developed in economics to analyze interactive decision processes, predict the outcomes of interactions, identify optimal strategies.

- Fundamental component of game theory is the notion of a game.

A game is described by a set of rational players, the strategies associated with the players, and the payoffs/utilities for the players. A rational player has his own interest, and therefore, will act by choosing an available strategy to achieve his interest (maximize/minimize utilities).

A player is assumed to be able to evaluate exactly or probabilistically the outcome or payoff (usually measured by the utility) of the game which depends not only on his action but also on other players’ actions.
Example: Nash Equilibrium & Prisoner’s Dilemma

- Two suspects in a major crime held for interrogation in separate cells
  - If they both stay quiet, each will be convicted with a minor offence and will spend 1 year in prison
  - If one and only one of them finks, he will be freed and used as a witness against the other who will spend 4 years in prison
  - If both of them fink, each will spend 3 years in prison
- Components of the Prisoner’s dilemma
  - Rational Players: the prisoners
  - Strategies: Stay quiet (Q) or Fink (F)
  - Solution: What is the Nash equilibrium/Pareto optimum of the game?

- Pareto Optimum:
  Cannot Improve oneself & not damage others.
- Nash Equilibrium: Best response.

<table>
<thead>
<tr>
<th></th>
<th>P2 Quiet</th>
<th>P2 Fink</th>
</tr>
</thead>
<tbody>
<tr>
<td>P1 Quiet</td>
<td>1,1</td>
<td>4,0</td>
</tr>
<tr>
<td>P1 Fink</td>
<td>0,4</td>
<td>3,3</td>
</tr>
</tbody>
</table>

Nash Equilibrium

Pareto Optimum
Rich Game Theoretical Approaches

- **Non-cooperative Static Game**: play once
- **Repeated Game**: play multiple times
- **Dynamic Game**: optimization utility over time
  - Evolutinal game
  - Stochastic game
- **Cooperative Game**
  - Nash Bargaining Game
  - Coalitional Game
- **Other Economic Approaches**: relevant to the game
  - Contract Theory
  - Auction Theory
  - Mean Field Theory

D. Fudenberg and J. Tirole, Game Theory, Cambridge, MA, USA: MIT Press
Comparison with Optimization Methods

• Broad the scope to model a problem
  \[ \min f(x) \]

  • **Non-cooperative Static Game:**
    Nash equilibrium, Pareto optimum, Stackelberg equilibrium…
  
  • **Dynamic Game:**
    Bayesian Nash equilibrium, subgame equilibrium…
  
  • **Cooperative Game:**
    Nash Bargaining solution, core, kernel, nucleus…

  • Some can / cannot be written as an simple optimization problems

• **Explain or Deal with the interactions between different entities**

  • Different entities: service providers, base stations, users, …
  
  • Explain: Not necessary to solve the problem in a distributed/parallel manner
  
  • Deal with: Usually need to solve the problem in a distributed/parallel manner
Two Specific Games

• Explain & Deal with the interactions between different entities
• For big data/ large-scale network: fast & scalable

• Matching Theory
• Hierarchical Game


Matching Theory: Introduction

• The Nobel Prize in Economic Sciences 2012

Lloyd S. Shapley
Developed the theory in the 1960s

Alvin E. Roth
Generated further analytical development
practical design of market institutions
Stable Marriage Problem

- Women and men be matched
- Respecting their individual preferences

Example of preferences:
Adam: Geeta, Heiki, Irina, Fran

Blocking Pair & Stable Matching

Adam \xrightarrow{X} Geeta

Geeta prefers Carl to Adam!

Blocking Pair

Bob \xrightarrow{X} Irina

Carl \xrightarrow{X} Fran

Carl likes Geeta better than Fran!

Stable Matching: No blocking pair.
The Gale-Shapley algorithm can be set up in two alternative ways:
- men propose to women
- women propose to men

Each man proposing to the woman he likes the best
- Each woman looks at the different proposals she has received (if any)
- retains what she regards as the most attractive proposal (but defers from accepting it) and rejects the others

The men who were rejected in the first round
- Propose to their second-best choices
- The women again keep their best offer and reject the rest

Continues until no men want to make any further proposals
Each of the women then accepts the proposal she holds
The process comes to an end
GS Algorithm

Adam
- Geeta, Heiki, Irina, Fran

Bob
- Irina, Fran, Heiki, Geeta

Carl
- Geeta, Fran, Heiki, Irina

David
- Irina, Heiki, Geeta, Fran

Fran
- Adam, Bob, Carl, David

Geeta
- Carl, David, Bob, Adam

Heiki
- Carl, Bob, David, Adam

Irina
- Adam, Carl, David, Bob
GS Algorithm

Adam
Geeta, Heiki, Irina, Fran

Bob
Irina, Fran, Heiki, Geeta
This is a stable matching
Geeta, Fran, Heiki, Irina

Carl
Irina, Heiki, Geeta, Fran

David

Fran

Carl > Adam

Geet

Heiki

David > Bob

Irina
Properties of GS Algorithm

• Complete matching: everyone gets married
• Stable matching: no blocking pair
• Runtime complexity: $O(n^2)$
Application: Full-Duplex OFDMA Networking

- One BS, M transmitters (TXs), M receivers (RXs), and K subcarriers
- A transceiver unit
- Self interference between antennas
- Transceiver-subcarrier pairing
  - One TX can only be paired with one RX
  - One subcarrier can only be paired with one TX-RX pair
- The transmit power of the BS is fixed

Objective: matching the TXs, RXs, and subcarriers to each other and adjust the BS power level in each subcarrier such that the total network’s sum-rate is maximized.


Simulation Result

- Random matching algorithm: the TXs, the RXs, and subcarriers are randomly matched with each other.
- Complexity level: the iteration number is much smaller than that of the centralized algorithm.
- The performance is close to the centralized algorithm.
Two Specific Games

- Explain & Deal with the interactions between different entities
- For big data/ large-scale network: fast & scalable

- Matching Theory
- Hierarchical Game
Hierarchies in Large-Scale Networks

- Ubiquitous Hierarchies in Large-Scale Networks

Hierarchies in Large-Scale Networks

Controller: Task to achieve

Agents: With selfish objectives

Challenge:

• Agents optimize selfish objectives rather than controller’s
• Need incentive mechanisms
Hierarchical Game with One Leader & One Follower

- Controller’s/ Leader’s Objective: $\min g(x)$
- Agent’s/ Follower’s Objective: $\min h(x)$
- Incentive: Let follower achieves leader’s optimum rather than its own

Problem Formulation as a Game

- Leader’s game:
  $$\arg \min_{\theta} g(x) + \theta x$$

- Follower’s game:
  $$\arg \min_x h(x) - \theta x$$

$x$: variable controlled by follower
$\theta$: price from leader to follower
Hierarchical Game with One Leader & One Follower

- Leader: give price & ask follower to minimize an incentive function
- Incentive Function Design
  \[ \min \Phi(\theta, x) = g(x) + h(x) - \theta x \]

  Lear’s utility  Follower’s utility  Payment

- Hierarchical Game Update
  - Leader’s Update
    \[ \theta_{p+1} = \frac{dh(x_p)}{dx} \]
  - Follower’s Update
    \[ x_{p+1} = \arg \min_x \Phi(\theta_{p+1}, x) \]

  Marginal Cost
Optimum Properties

- **Relaxed Stackelberg Equilibrium**
  - Optimum of Leader’s Original Objective: \( \min_x g(x) \)
  - Optimum of Incentive Function: \( \min_x \Phi(\theta_p, x) = g(x) + h(x) - \theta_p x \)
  - Optimum of Follower’s Utility: \( \min_x h(x) - \theta_p x \)
  - Same Optimum: \( \begin{cases} 
    \min_x g(x) \\
    \min_x h(x) - \theta_p x 
  \end{cases} \)
  - Not Optimum of Leader’s Utility: \( \min_\theta g(x(\theta)) + \theta \cdot (x(\theta)) \)

- **Conditions:**
  - Strongly convex leader’s utility function
  - Convex follower’s utility function
  - Uniform Lipschitz gradient for follower’s utility function
Convergence & Scalability Properties

• Just like Gradient Descent/Ascent: First-order numerical optimization

  □ Linear Speed: \( \varepsilon = o(1/p) \)

  □ Scalability

\[
\begin{align*}
\arg \min_{\theta_i} \sum_{i=1}^{N} g(x_i) \\
\arg \min_{x_i} h_i(x_i) - \theta_i x_i
\end{align*}
\]
Hierarchical Game with One Leader & Multiple Followers

• Leader’s game:

\[
\arg \min_{\{\theta_i\}} \sum_{i=1}^{N} g_i(x_i) + \sum_{i=1}^{N} \theta_i x_i
\]

• Followers’ game:

\[
\arg \min_{x_i} h_i(x_i) - \theta_i x_i
\]

• Constraints:

\[
\sum_{i=1}^{N} A_i x_i = B
\]


• Leader’s Update
  \[ \theta_i^{p+1} = \frac{dh_i(x_i^p)}{dx_i} \]

• Followers & Leader’s Coordination

  □ **Sequential Follower’ Update**

  \[ x_i(t + 1) = \arg \min_{x_i} \Phi_i(\theta_i^{p+1}, x_i) + \lambda A_i x_i + \frac{\rho}{2} \left[ A_i x_i + \sum_{k=1}^{i-1} A_k x_k(t+1) + \sum_{i+1}^{N} A_k x_k(t) - B \right]^2 \]

  □ **Leader’s Dual Update**

  \[ \lambda(t + 1) = \lambda(t) - \rho \left( \sum_{i=1}^{N} A_i x_i(t+1) - B \right) \]
Optimal Properties

- **Relaxed Stackelberg Equilibrium**
  - Optimum of Leader’s Original Objective
    \[ \arg\min_{\theta_i} \sum_{i=1}^{N} g_i(x_i) \]
  - Optimum of Follower’s Incentive Function
  - Optimum of Follower’s Utility
  - Not Optimum of Leader’s Utility
    \[ \arg\min_{\theta_i} \sum_{i=1}^{N} g_i(x_i) + \sum_{i=1}^{N} \theta_i x_i \]

- **Conditions:**
  - Decomposed utility functions for the leader
  - Linear constraints (Convex can be right)
  - Strongly convex leader’s utility function
  - Convex follower’s utility function
  - Uniform Lipschitz gradient for follower’s utility function
Convergence & Scalability

- Linear Speed on Outer Loop: \( \varepsilon = o(1/p) \)
- Scalability on Outer Loop

\[
\sum_{i=1}^{N} g_i(x_i)
\]

\[
g_i(x_i) = |x_i|^2
\]

\[
h_i(x_i) = \exp(x_i)
\]
Hierarchical Game with Multiple Leaders & Multiple Followers

- Utility of Each Leader

\[
\min_{x_{i,*}} G_i(x_{i,*}) = \sum_{j=1}^{N} g_{i,j}(x_{i,j})
\]

- Utility of Each Follower

\[
\min_{x_{*},j} H_j(x_{*},j) = \sum_{i=1}^{K} h_{i,j}(x_{i,j})
\]

\(x_{i,j}\) : resources provided from agent \(j\) to controller \(i\)

Z. Zheng, L. Song, Z. Han, G. Li, and V. Poor, “Multi-leader multi-follower game-based ADMM for big data processing,”
MLMF Game-based ADMM

Outer Loop: Leaders’ Price Update

Leaders: Parallel Dual Update
- Leader 1
- Leader 2
- ... Leader K

Followers: Sequential Primal Update
- Follower 1
- Follower 2
- ... Follower N

Inner Loop: ADMM
Game Theory for Big Data Processing

• Motivation
• Introduction of Game Theory
• A General Hierarchical Game
• Applications
Hierarchical Game: Application for Edge Caching

- Limited backhaul resources vs excessive streaming
- Anticipated file demand popularity by service provider (SP)
- SP proactively transmits popular files to a large number of edge nodes (ENs)
  - ENs - base stations, small-cell base stations, or WiFi access points


Stackelberg Game-based ADMM

- **Objective**
  - Minimize total backhaul for SP

- **Application of the Stackelberg game-based ADMM**
  - One SP (controller) and multiple ENs (agents)
    MLMF game-based ADMM reduces to Stackelberg game-based ADMM
  - SP pays for the ENs’ backhaul & storage resources
Backhaul Utility

Average backhaul resources vs demands change times

- 20 Files
- 100 ENs
- 1000 Users

- Popularity-based caching: cache the most popular files on ENs
- Random caching: cache files randomly

- Cost less backhaul resources than popularity-based caching & random caching
- Cost more backhaul resources when users change file demands more frequently
Convergence Speed

Average iteration time vs Number of users

- Iteration times: change sub-linearly with the number of Users & T
- Iteration times: keep below 2.

- 20 Files
- 10, 50, 100 ENs
- 1, 6, 12, 24 Demand Change
- Accuracy: $10^{-2}$
Summary of Game Theory

- Provide more possible approaches in modeling and solving
- More specific algorithms are needed in the future
Summary of Tutorial

- Signal Processing Methods for Future Communication Works
  - Learning Methods
  - Commercial Systems
  - Large Scale Optimization
  - Game Theory based Approaches

www2.egr.uh.edu/~zhan2/big_data_course/
wireless.egr.uh.edu/research.html
Thanks