Machine Learning for RAN: Delusion or Salvation?

Slawomir Stanczak
Why ML for Communications (=MLC)?

• **Entry points for ML-based improvements**
  1. high complexity (bad models)
  2. inefficient computation (limited resources)
  3. slow convergence (low latency applications)

• **Potential benefits**
  1. enable to cope with increased complexity
  2. enhance efficiency
  3. facilitate cognitive network management
  4. provide robust predictions
Load Learning

Problem
What are users’ rates as a function of the load at each base station?

General case: \( M \) cells, \( N \) test points

Reliable rate-load mapping estimates/predictions are key to reliable QoS predictions

Performance Improvement due to Predictions

# Classic vs. ML Approach (inspired by David Wipf)

## Problem
Find a load vector $x^*$ given users’ rates $\theta$ and network configuration

<table>
<thead>
<tr>
<th>Classic approach (model-driven)</th>
<th>ML approach (data-driven)</th>
</tr>
</thead>
<tbody>
<tr>
<td>• Modeling $f_\theta(x)$</td>
<td>• Choose a function set ${g_\omega}_{\omega \in \Omega}$</td>
</tr>
<tr>
<td>• Simplification: $\hat{f}_\theta(x)$</td>
<td>• Learn $\omega$ offline (e.g. from data)</td>
</tr>
<tr>
<td>• Human-designed algorithm</td>
<td>• $\hat{x} = g_\omega(\theta) \approx x^*$</td>
</tr>
<tr>
<td>input: $\theta$</td>
<td></td>
</tr>
<tr>
<td>while &lt;some condition is met&gt;</td>
<td></td>
</tr>
<tr>
<td>$x^{(n+1)} = T(x^{(n)})$; end</td>
<td></td>
</tr>
<tr>
<td>output: $\hat{x} = x^*$</td>
<td></td>
</tr>
</tbody>
</table>

→ **ML is much more than neural networks!**
Which Tools for MLC?

Key issues:
- Energy efficiency neglected
- Domain knowledge ignored ➔ Function properties not preserved
- Choice of performance metrics
- Amount of training data

According to Andrew Ng

Lower layers (PHY/MAC)
Collection of training data is limited
- Fast time-varying channels and interference
- **Short stationarity interval (V2X: 10-40ms)**
- Distributed data
- Limitations on computational power/energy

Higher layers
Huge datasets are available but
- Incomplete data (missing measurements for long periods)
- Erroneous data (e.g. software bugs)
- Misaligned data (different times)
- Time series (i.i.d. unrealistic)
Load Learning (cont.)

**Challenge:** The rate-load mapping (RLM) is highly dynamic and nonlinear owing to interference

➔ training must be short
➔ important properties must be preserved

• Model-based approaches require too much a priori information
  **But we should not ignore models**

• The RLM has a rich structure (e.g., *monotonicity and Lipschitz*)
  They are hard to exploit in typical machine learning tools
Robust Online Load Learning

Hybrid-driven robust methods under uncertainty (e.g., few training samples)

Demands on MLC

- **Robust online** ML with good tracking capabilities
  - ML with small (uncertain) data sets

- Exploit **domain knowledge** (e.g. models, correlations, AoA)
  - Hybrid-driven ML approaches (e.g. use production data)
  - Learn features that change slowly over frequency, time...
  - Preserve important function properties

- **Distributed learning** under communication constraints
  - New functional architectures for Big Data analytics

- Low-complexity, **low-latency implementation**
  - New algorithms, massive parallelization
Learning in (Reproducing Kernel) Hilbert Spaces

Use projection methods in RKHS:

➔ Easy to exploit side information

➔ Initial fast speed

➔ Low complexity

➔ Convergence guarantees

➔ Massive parallelization via APSM for fast learning on GPUs

Learning-based Reception for 5G NOMA

Can we design better neural networks?
Sparsity in Communication Systems

- Sparsity in the data (soft sparsity)
- Sparsity in the channel (soft sparsity)
- Sparsity in the user activity (hard sparsity)
- Sparsity in the network flow (hard sparsity)

We aren’t likely to get a 1000X improvement in compute with the traditional, pure hardware improvements, or even better software and communication to put more chips together. It will need co-design of algorithms and compute e.g. can we create a model with a 1000X more parameters, but using only 10X more compute? I believe sparse models that address this issue and systems that can take advantage of these constraints will make a big difference.

Rajat Monga, Google Brain, Lead Developer of TensorFlow
Sparsity in Communication Systems

- We can use $B_p$-balls to model sparse signals $B_p = \{ x : \sum_{i=1}^{N} |x_i|^p \leq 1 \}$

Sparse Recovery via a Deep Neural Network

- CS methods are *not* suitable for low-latency applications
- Training must be short
  ➔ **Design** a good DNN for sparse recovery
Optimization for MMSE Recovery

\[ A \in \mathbb{R}^{3 \times 6}, \text{ activation } \sigma \text{ with polynomial degree 9} \]

- proposed (solid/dashed): linear estimator + \( \epsilon \) \( \rightarrow \) online feasible
- LASSO (dotted): many iterations \( \rightarrow \) online infeasible

Designing DNNs via Laplace Techniques

- Input uniformly distributed on
  \[ B_1 = \{ x \geq 0 : \sum_{i=1}^{N} x_i \leq 1 \} \]
- The conditional MMSE estimator is a polytope centroid under certain conditions.
  ➔ Volume and moment computation
- Implementable using the DL architecture

\[ x^T A^T y^T \]

\[ \text{vol}(\mathcal{P}_y)^{-1} \mu = \hat{x} \]
Numerical Experiments with Training

Real data

Take-away Message

- ML/AI might be a “salvation” for industrial communication
- But there is a strong need for robust online ML methods
  ➔ Exploit domain knowledge: Hybrid-driven distributed ML
  ➔ Learn feature insensitive to frequency bands, phases …
- No time and data for extensive training of DNN
  ➔ Design good NN architectures for a given task
Exploiting „Interference“ for Learning

\[ y = h^T x + n \Rightarrow \psi\left(\sum_{k=1}^{K} \phi(x_k)\right) \]

- Bjelakovic, M. Frey and S. Stanczak (2019). Distributed Approximation of Functions over Fast Fading Channels with Applications to Distributed Learning and the Max-Consensus Problem. 57th Annual Allerton Conference on Communication, Control, and Computing, 24-27 Sept. 2019 in Urbana, IL, USA
References