RESIDUAL ALGORITHMS VIA LINEAR PROGRAMMING NORMALIZATION

RESEARCH MOTIVATION

- How to solve Approximate Dynamic Programming problems efficiently? [2] [3]
- How to improve residual algorithms? [1] [6]
- How to combine general-purpose and problem-specific approximation algorithms?

THE NETWORK INVENTORY CONTROL PROBLEM [8]

- Given inventory \( x(t) \) and a request for resource \( a \), should the algorithm accept or reject the request?
- High impact: 1 ppt improvement has significant financial benefit
- Well studied: current best practice is LP + DP decomposition [7]

MAIN IDEA

ORIGINAL PROBLEM

CUSTOM APPROXIMATION FIRST ORDER EFFECT

DEEP NEURAL NETWORK HIGHER-ORDER EFFECT

MATH

Approximate value function with

\[ V^*(x|\theta) = U(x)W(x|\theta) \]

where

\[ U(x) = \max \mathbf{r}^T y \text{ s.t. } A y \leq x \text{ and } 0 \leq y \leq \mathbf{p}^T t \]

Is the LP normalizer [4] [5] and \( W \) is the neural network

Minimize Bellman residual

\[ U(x,t)W(x,t|\theta) - \sum_{0 \leq n \leq N} \max_{\mathbf{r} \in \mathcal{R}(x)} \left[ r_n u + U_{t+1}(x-a_n)W_{t+1}(x-a_n|\theta) \right] \]

IMPLEMENTATION DETAILS

- Calculate \( U \) on the fly using standard LP solver [4]
- Generate empirical average of Bellman residual via Monte Carlo simulation [5]
- Gradient descent via Adam
- Validation using discrete event simulation

PRELIMINARY RESULTS

- Optimality in same ballpark with current state-of-the-art approximation algorithm
- Savings on computation cost
- Scalability

NEXT STEPS

- Train general residual network across multiple problems
- Improve network architecture
- Test against pure deep neural networks
- Large scale industry experiment

REFERENCES