Open Student Projects

Marius Lindauer & Frank Hutter
General Points

• Flexible scope for most projects:
  – OK for BSc theses or MSc projects/theses

• All projects are very up-to-date
  – State-of-the-art research topics
  – Successful projects can lead to publications
    (typically requires some more work afterwards, though)

• Team work recommended for MSc projects
  – 2-3 students per team (max. 4)
  – You can get a lot more done
  – Often, team work is more efficient & rewarding

• MSc theses typically only after a MSc project in our group
Main Topics for Projects

1. Multiple Fidelities
2. Optimizing a policy network for Atari
3. Learning to Design RNA
4. Learning Moves through Neural Architecture Space
5. Learning to Learn / Optimize
6. Algorithm Parameter Control
7. Monte-Carlo Tree Search for Hyperparameter Optimization
Multiple Fidelities

- There are many ways to approximate a blackbox function
  - Initial learning curves
  - Subsample of datasets
  - Downsampling images
  - ... 

- So far, HB and BO-HB can only exploit either one of these
  - We'd like to exploit many of these multiple fidelities:
    - e.g., short runs on a subsample of downsampling images
  - Of course, the goal is still to find a configuration
    - that is good for long runs on all full-sized images
  - Simple approaches might do well (like Hyperband did);
    - Potentially could be improved by model later
  - Ultimately, one should learn exploration strategies (e.g., using RL)
Optimizing a policy network for Atari

- **RL** is the main approach for playing Atari
  - See the DQN Nature paper

- **Evolutionary strategies are an alternative**
  - All we need is a policy network that maps states to actions
  - We can optimize this network by evolutionary strategies
    - Highly parallelizable
    - Well studied
    - OpenAI published a paper about this last year
    - Previous MSc project: showed that even simpler methods work well
    - Several possibilities for follow-up work; code base available to start from
Learning to Design RNA

- Folding an RNA sequence to its secondary structure: $O(n^3)$

- Designing a sequence that folds into a given structure:
  - unknown complexity, hard problem in practice

- Approach:
  - RNN policy network that reads one structure element at a time and outputs the corresponding nucleotide
  - Score rollouts by folding designed structures and counting errors; optimize with policy gradient
Learning Moves through Neural Architecture Space

● Recall NASH: Neural Architecture Search by Hillclimbing
  ○ Each move uses a network morphism to start from good weights
  ○ But moves are completely random

● Formulate as an RL problem
  ○ State: current network
  ○ Actions: which morphism to apply
  ○ Reward: final performance
Learning to Learn / Optimize for DNNs

• View an optimizer (like SGD, Adam, etc) as a mapping from some state to the next step to be applied
  – Parameterize that mapping as a neural network
  – Use RL or gradient descent to optimize that network

• Project
  – Bootstrap from existing code
  – Can we learn cosine annealing & SGDR?
  – Can we apply this for different hyperparameters (weight decay, etc)?
  – Can we learn an optimizer that is actually better than Adam etc in practice?
Learning to Learn / Optimize for Blackbox Optimizers

• Learning to learn without gradient descent by gradient descent
  – Learns a global blackbox optimizer
  – State: set of points in the space already evaluated, and their performance
  – Action: which point in the space to evaluate next
  – Loss: performance obtained (e.g., in fixed horizon)
  – Policy network, to be optimized with SGD

• Project: reimplement this approach
  – Code is not available
Learning to Learn / Optimize for SAT

• Learning an entire algorithm component
• For basically any algorithm; here: a **SAT solver**
  – Propositional satisfiability solving: \((x_1 \lor x_2) \land (-x_1 \lor x_3) \land \ldots\)
  – Many applications in software & hardware verification or other NP-hard problems

• Represent the algorithm component as a neural net with the component’s output as the network output.
• Use **policy gradient** to optimize the network to improve the algorithm’s overall performance

• Expensive in terms of compute time, but (in principle) very cheap to set up in terms of human time
Algorithm Parameter Control

- Algorithms have many (hyper-)parameters
  - Parameter tuning often required to achieve peak performance
- Is it enough to set these parameters once? Or do we have to adapt them over time?
  - → Parameter Control
  - So far, manually engineered adaption schemes (e.g., learning rate schedules)
- Project
  - Model algorithm parameter control as RL task
  - Goal: Learn a policy for algorithm parameter control
  - State: State of the algorithm characterized by features
  - Action: Change parameter configuration
Algorithm Parameter Control

- **BSc Thesis: simplified version of parameter control:**
  - Learn a policy to select a well-performing parameter configuration from a small, finite set of configurations
  - → Contextual Bandits

- **MSc Project: easy version of parameter control**
  - Roberto Battiti et al applied “old” RL methods in 2012 on a SAT solver with a single parameter
  - We have the data and the code
  - Goal: Improve over the Battiti’s results by using state-of-the-art RL

- **MSc Thesis: full parameter control**
  - Parameter control for a full-fledged SAT solver
  - Many parameters, complex state, …
Monte Carlo Tree Search for HPO

- Configuration spaces are often complex and have hierarchy of hyperparameters
  - So far, hierarchy only implicitly handled
- Trees should be handled as trees
  - One way would be to use MCTS
  - Could be combined with ideas from AlphaGo, e.g., we learn from previous HPO runs on other datasets
- Target domain: Hyperparameter Optimization with auto-sklearn
  - In the long-term: algorithm configuration for arbitrary algorithms
Interested?

- Talk to us today

- Write us an email: aad-staff@cs.uni-freiburg.de
  - Please follow the instructions at http://ml.informatik.uni-freiburg.de/open_projects.html
    - Projects listed on this webpage are outdated!