Reinforcement Learning
for
Automated Performance Tuning:
Initial Evaluation for Sparse Matrix Format Selection

Warren Armstrong and Alistair P. Rendell

Department of Computer Science
College of Engineering and Computer Science
Australian National University
Outline

- Background on Reinforcement Learning
- Motivation
- General framework and sample problem
- Evaluation
What is Reinforcement Learning

- Learning how to act based on experience
  - Dynamic and probabilistic environments.

- Contrast to supervised learning: no examples required.
Automated Tuning as a Reinforcement Learning Problem

• Automated Tuning
  – Often requires search – exhaustive or heuristic
  – Rapidly changing conditions require many searches

• Reinforcement Learning
  – No need for heuristic: just define the goal.
  – Rather than search for an optimal solution, RL searches for an optimal mapping from states to optimal solutions
  – Our idea: Rate of change of mappings is less than rate of change of solutions.
General Framework and Sample Problem
Sample problem: Sparse Matrix Format Selection

- A sparse matrix is a matrix with zeroes which can be profitably ignored.
- There exist many different storage formats for achieving this.
  - Format efficiency varies with matrix structure
  - This is generally only knowable at runtime
- Our goal: Given a matrix:
  - Characterise it
  - Select a format for it
  - Reformat the matrix and multiply it by a vector
Existing Tuning Methods

- **ATLAS, PhiPAC**
  - Tune based on architecture – for dense matrices
- **OSKI / AcCELS**
  - Architectural tuning defines a search space
  - Format selection based on (heuristic) runtime search of this space
Applying Reinforcement Learning

- **Actions**
  - Switching to different configurations
  - CSR, BCSR-8x8, COORD

- **Sensory perceptions**
  - N sensors map onto an m-dimensional state space
  - Rows, Columns, Nonzeroes, Inter-row spread, neighbour count.

- **Reward**
  - Any measurable quantity
  - Inverse of execution time
Selecting an Action

- Two types of action: Exploitation and Exploration
- Exploration costs, so only do it when it is likely to succeed.
- Exploration Chance: \( P = E + K \), where
  - \( E \) is fixed at runtime
  - \( K \) varies based on the recent exploration success rate.
- Every time a choice must be made, exploration occurs with probability \( P \).
The Mechanics of Exploration

- Goal: Compare choices.
  - Assumption: a state persists for $D$ steps, where $D$ is a user-specified constant.
  - *In the sample problem, $D$ is the number of multiplications*

- Break into multiple actions:
  1) Execute $D/2$ steps with the current optimal choice to obtain reward $R_c$
  2) Execute $D/2$ steps with an alternative choice to obtain reward $R_a$

- If $R_a$ is better than $R_c$ then exploration succeeded.
Representing the State Space

- The state space is discretised into partitions
  - Example with $m = 2$:
Using the State Space Representation

• Initial state:
  - One partition, no recorded failures, arbitrary optimal choice

• Change results from exploration:
  - Failed exploration is remembered
  - Successful exploration results in a new partition:

• Partition width determines generalisation
Evaluation:
Does the learning algorithm make good choices?
Evaluation Strategy

- Matrix generation
- Matrix sequence characterisation
- Defining an evaluation metric
- Simulation
Matrix Generation

- Large numbers of matrices needed.
- Generation through probabilistic selection of attributes
  - size, density, pattern
  - Via uniform and normal distributions.
- How realistic are the generated matrices?
Matrix Generation

- Not identical, but reasonably similar
Matrix sequence characterisation

- Characteristics of each matrix
- Time to change each matrix into each format
- Multiplication time for each matrix in each format.
Defining an Evaluation Metric

- For each format $F$ and each matrix $m$, define \( \text{rank}(F, m) = 1 \) if the format is optimal, \( 3 \) if the format is the worst choice, \( 2 \) otherwise.
- Slice the matrix stream into windows of 100 matrices.
- For each window $W$, compute the mean rank achieved by the agent's choices.
Results

- The algorithm made the correct choices
- The performance of the algorithm depends on its ability to generalise
- Generalisation reduces exploration.
The algorithm made correct choices
Parameter Sensitivity

E = fixed exploration rate, CW = new partition proportion
Generalisation and Exploration

Exploitation with 2000 multiplications per matrix

E = epsilon_base, CW = cut_width
Best performance with least exploration: generalisation is important.
Discussion

- Execution time gains
- Place matrix generation on a sounder footing
- Models of matrix evolution
- Possible overfitting
- Tuning the learning algorithm
Questions?