CS 598CM: ML for Compilers and Architecture
Instructor: Charith Mendis
Brief Announcements

- **Paper Reviews**: hotCRP access

- **Paper Presentations**: Meet the week before, use Piazza or email to schedule

- **Piazza**: on-line discussions of papers; let’s be active!

- **COVID19**: If you feel sick, please stay at home.
Recap

- Neural Networks Primer
  - Perceptrons
  - Multilayer perceptrons
  - Convolutional Neural Networks
  - Recurrent Neural Networks
  - Graph Neural Networks
  - Transformers
Lecture 6: ML Techniques + Auto-tuning
Genetic Algorithms

Survival of the Fittest

“Survival of the form that will leave the most copies of itself in successive generations.”

Charles Darwin
Genetic Algorithms

- Find the set of genes (parameters settings) that are the fittest (optimizes an objective) using genetic evolution.
Genetic Algorithms

• Find the set of genes (parameters settings) that are the fittest (optimizes an objective) using genetic evolution.

Repeat until budget exhausted or population meets convergence criteria.
Evolution

Mutations

Randomly mutate parts of the gene

Crossovers

Mix of two Genes
Evolution

Population i → Evolutions → Compute Fitness → Keep the Fittest → Population I+1
Can we train NNs using GAs?

YES!

Weights

Evolutions

Change weights

Loss function

Compute Fitness

Keep the Fittest

New weights

Population i

Population i+1

Why aren’t we using GAs? Completely Random; Inefficient
Performance Tuning

Program Configuration

Population i

Evolutions
Change configuration

Cost Model / Runtime
Compute Fitness
Keep the Fittest

New Program Configuration

Population I+1
Simulated Annealing

• Optimization algorithm / meta-heuristic

• Similar to the general genetic algorithm structure

• Basic idea => Propose a solution (initial weights) and propose other solutions (new weights) and accept them based on a calculated probability
Simulated Annealing

- Select an initial solution
Simulated Annealing

- Select an initial solution
- Select solution within a small neighborhood
Simulated Annealing

- Select an initial solution
- Select another solution within a small neighborhood
- Accept the new solution with probability $P$

$$P = \begin{cases} 
1 & \text{if} \Delta c \leq 0 \\
b^{\frac{-\Delta c}{t}} & \text{if} \Delta c > 0 
\end{cases}$$

$t$ - temperature (hyperparameter)

$\Delta c$ - Cost difference

Should we accept

Option 1? Yes
Option 2? Maybe
Simulated Annealing

- Select an initial solution
- Until Convergence
  - Change temperature
  - Select another solution within a small neighborhood
  - Accept the new solution with probability $P$

$$P = \begin{cases} 
1 & \text{if } \Delta c \leq 0 \\
\frac{1}{e^{-\Delta c/t}} & \text{if } \Delta c > 0 
\end{cases}$$

$t$ - temperature (hyperparameter)

$\Delta C$ - Cost difference
Acceptance Probability

\[ P = \begin{cases} 
1 & \text{if } \Delta c \leq 0 \\
 e^{-\Delta c / t} & \text{if } \Delta c > 0 
\end{cases} \]

<table>
<thead>
<tr>
<th>Change</th>
<th>Temperature</th>
<th>Acceptance Probability</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.2</td>
<td>0.9</td>
<td>0.8007</td>
</tr>
<tr>
<td>0.4</td>
<td>0.9</td>
<td>0.6412</td>
</tr>
<tr>
<td>0.6</td>
<td>0.9</td>
<td>0.5134</td>
</tr>
<tr>
<td>0.8</td>
<td>0.9</td>
<td>0.4111</td>
</tr>
</tbody>
</table>

Sample acceptance probabilities given the a temperature of 0.9

<table>
<thead>
<tr>
<th>Change</th>
<th>Temperature</th>
<th>Acceptance Probability</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.2</td>
<td>0.1</td>
<td>0.1353</td>
</tr>
<tr>
<td>0.4</td>
<td>0.1</td>
<td>0.0183</td>
</tr>
<tr>
<td>0.6</td>
<td>0.1</td>
<td>0.0025</td>
</tr>
<tr>
<td>0.8</td>
<td>0.1</td>
<td>0.0003</td>
</tr>
</tbody>
</table>

Sample acceptance probabilities given the a temperature of 0.1

Acceptance probability increases with decreasing \(|\Delta C|\) and increasing \(t\).

How should be change \(t\) across iterations?
Start high (exploration) and decrease (exploitation)
Sequential Decision Making

Markov Decision Process (MDP)

```
State       \rightarrow\choose\text{a "valid" action}\rightarrow\text{Policy}\rightarrow\text{New State}\downarrow\text{Reward (Win / Loss)}\rightarrow\text{Iterate}
```
Vectorization as a MDP

Choose a “valid” action

State

Iterate

New State

Reward (Speed of execution)

\[
\begin{align*}
\end{align*}
\]

\[
\begin{align*}
\end{align*}
\]
Imitation Learning

• Collect a dataset of (state, action) pairs from an oracle

• Essentially supervised learning on those collected experiences
What if the model makes a mistake?

State \rightarrow \text{Action (A)} \ (Not \ the \ optimal \ action \ - \ OA) \rightarrow \text{Optimal State}

State \rightarrow A \rightarrow \text{Unseen State}

The model may make even more erroneous decisions!

DAgger algorithm gives a solution (Ross et. al)!
Reinforcement Learning

- No oracle, but we have access to the reward
- Devise a mechanism to take the best action that would maximize the total reward
- Not greedy; but tries to maximize the total reward for the entire episode

![Diagram](image-url)
Different RL algorithms

• Directly learn the policy function (policy gradient techniques)
• Learn the value function (Q-learning)
• Hybrid (Actor-critic models)

http://rail.eecs.berkeley.edu/deeprlcourse/

https://nanjiang.cs.illinois.edu/cs498/
Auto-tuning

- Automatically find the best program or program configuration in an optimization space according to some metric

Exhaustive Search
Genetic Algorithms
Simulated Annealing
Reinforcement Learning
Examples of auto-tuning

- Normally we just do **parameter tuning**
- Loop unrolling (unroll factor)
- Compiler Flags (binary vector)
- Operator fusion, layout selection, tile-size selection in DL stacks (how?)

Fusion?
Halide

**Halide auto-tuner:** Generate Program Constructs as well; not just parameter tuning

```c
func blur_3x3(func input) {
  func blur_x, blur_y;
  Var x, y, x2, y2;

  // The algorithm - no storage or order
  blur_x(x, y) = 1.0 * input(x, y) + input(x-1, y) + input(x-1, y-1) / 8;
  blur_y(x, y) = 1.0 * input(x, y) + input(x, y-1) + blur_x(x, y-1) / 8;

  // The schedule - defines order, locality; implies storage
  blur_y.tile(x, y, x1, y1, 256, 8);
  .vectorize(x1, 0); parallel(y);
  blur_x.compute_at(blur_y, x); vectorize(x, 8);

  return blur_y;
}
```

Agenda for Auto-tuning

- **Empirical Autotuning**: ATLAS
- **Exposing Choices**: Petabricks
- **Autotuning Techniques**: Bliss (Bayesian Optimization)
- **Frameworks**: OpenTuner
- **Scaling Up**: GPTune
- **Diverse Workloads**: GSwitch (graphs)
- **ML in compilers**: Presentation from a Google Research Scientist
Paper Reading

• Try to read evaluation, contributions and motivating examples first

• Then go into details of the paper

• Focus on high-level ideas and not on the implementation in your summary
Any Questions?