CS 598CM: ML for Compilers and Architecture

Instructor: Charith Mendis
Brief Announcements

• **Paper Reviews:** hotCRP access

• **Paper Presentations:** Meet during office hours (1-2pm on Tuesdays)

• **Piazza:** (optional) on-line discussions of papers

• **COVID19:** If you feel sick, please stay at home and you can join the class virtually using Zoom
Recap

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

• Works on graph structured data

• Main goal is to find representations for nodes or edges (node or edge embeddings) that can be used for many downstream tasks

**Embeddings**

<table>
<thead>
<tr>
<th>‘cat’</th>
<th>0.5</th>
<th>0.8</th>
<th>0.7</th>
<th>0.9</th>
<th>0.9</th>
</tr>
</thead>
<tbody>
<tr>
<td>‘dog’</td>
<td>0.3</td>
<td>0.4</td>
<td>0.7</td>
<td>0.5</td>
<td>0.9</td>
</tr>
</tbody>
</table>

Word Embeddings

Node Embeddings
Graph Neural Networks

Find node embeddings for
- Protein folding
- Node clustering
- Variable inference

Protein Interface Prediction using Graph Convolutional Networks

Alex Fout1
Department of Computer Science
Colorado State University
Fort Collins, CO 80523
fout@colostate.edu

Jonathon Byrd1
Department of Computer Science
Colorado State University
Fort Collins, CO 80523
jbyrd@colostate.edu

Babir Shariat2
Department of Computer Science
Colorado State University
Fort Collins, CO 80523
babir@cs.colostate.edu

Asa Ben-Hur3
Department of Computer Science
Colorado State University
Fort Collins, CO 80523
asa@cs.colostate.edu

Spectral Clustering with Graph Neural Networks for Graph Pooling

Filippo Maria Bianchi1, Daniele Grattarola2, Cesare Alippi2,3

Predicting drug–target binding affinity with graph neural networks

Thin Nguyen, Hang Le, Thomas P. Quinn, Thuc Le, Svetla Venkatesh
doi: https://doi.org/10.1101/684662
Computational Model

Bulk Synchronous Parallel Style

All nodes are updated at layer (L) from layer (L-1) values in parallel

Layer (L-1) → Layer (L)
Computational Model

Bulk Synchronous Parallel Style

All nodes are updated at layer (L) from layer (L-1) values in parallel

Updates from a neighborhood of nodes (message passing)
Computational Model

Bulk Synchronous Parallel Style

- All nodes are updated at layer (L) from layer (L-1) values in parallel
- Updates from a neighborhood of nodes (message passing)
- Barrier until all nodes are updated
Graph Convolutional Network (GCN)

**Neighborhood** = All single-hop

**Message**  
Fixed importance  
Learnable Weights  
\[ d = \frac{1}{\sqrt{d_{ii}d_{jj}}} \]

**Aggregation**  
Sum (fixed)

**Update**  
\[ h_v^{(l+1)} = \sigma \left( \sum_{u \in ne(v)} \frac{1}{\sqrt{d_{ii}d_{jj}}} h_u^{(l)} \cdot W(l) \right) \]
GraphSAGE

Generalization of GCN

Suitable for Inductive Tasks

**Neighborhood** = All single-hop

**Message**  
Fixed importance  \( d = 1 \)  
Learnable Weights  \( W(l) \)  
Subsample nodes

**Aggregation**  
+, pool, LSTM

**Update** (concatenate current node embedding)

\[
h_v^{(l+1)} = \sigma \left( \left( h_v^{(l)} || \text{AGG}(h_u^{(l)} | u \in N(v)) \right) . W^{(l)} \right)
\]

Inductive Representation Learning on Large Graphs (NeurIPS 2017)
Transfomers Primer

Easily parallelizable and can distribute work

Attention Is All You Need (NeurIPS 2017)
Function Approximators

• Neural networks are non-linear function approximators

• We usually use gradient based techniques to learn the parameters (weights)
  - Other techniques can be used as well

• Can be used to approximate many parts of a system
  • Policy of a moving robot
  • Image classification system
  • Machine Translation

• Novelty comes from designing new topologies and adapting your system to use NNs.
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

Population i

Evolutions

Change weights

Loss function

Compute Fitness

Keep the Fittest

New weights

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 $P$

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

$t$ - 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 $P$

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

$t$ - temperature (hyperparameter)

$\Delta C$ - Cost difference
Acceptance Probability

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

<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

\[ t \uparrow \]

\[ \Delta C \downarrow \]

How should be change \( t \) across iterations?

Start high (exploration) and decrease (exploitation)
Sequential Decision Making

Choose a “valid” action

Policy

State  \rightarrow  New State

Iterate

Reward (Win / Loss)

Markov Decision Process (MDP)
Vectorization as a MDP

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

Choose a “valid” action
\{
  \{a[3], a[2]\}, \\
  \{a[1], a[2], a[3]\}, a[4]\}

Iterate
\{
  \{a[3], a[4]\}, \\
  \{a[1], a[2]\}, a[3:4]\}

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

New State

Reward (Speed of execution)
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 → Action (A) (Not the optimal action - OA)

State → Optimal State

State → 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
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

Transformation Space  Optimization Strategy  Cost Model

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_x32(func input) {
    func blur_x, blur_y;
    var x, y, x1, y1;

    // The algorithm - no storage or order
    blur_x(x, y) = (input(x-1, y) + input(x, y) + input(x+1, y))/3;
    blur_y(x, y) = (blur_x(x, y-1) + blur_x(x, y) + blur_x(x, y+1))/3;

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

    return blur_y;
}
```

```
blur_x.compute_at(blur_y, x).
    .vectorize(x, 8);
blur_y.tile(x, y, x1, y1, 8, 8)
    .parallel(y)
    .vectorize(y, 4);
blur_x.split(x, x1, 8)
    .vectorize(x1, 4).parallel(x);
blur_y.split(x, x1, 8)
    .vectorize(x1, 4).parallel(x);
blur_x.store_root()
    .compute_at(blur_y, y)
    .split(x, x1, 8)
    .vectorize(x1, 4);
blur_y.split(y, y1, 8)
    .vectorize(x, 4)
    .parallel(y);
```
Agenda for Auto-tuning

- **Empirical Autotuning:** FFTW
- **Exposing Choices:** Petabricks
- **Autotuning Techniques:** Bliss (Bayesian Optimization)
- **Frameworks:** OpenTuner
- **Scaling Up:** GPTune
- **Diverse Workloads:** GSwitch (graphs)
- **Auto-tuning in industry:** 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?