CS 598CM: ML for Compilers and Architecture

Instructor: Charith Mendis
Brief Announcements

• Only 12 students have filled out the paper selection form.
• We will have to present 2 times to cover all 20 papers.
• In that case, I am thinking of making the following grading changes
  • Mini-quizzes will be eliminated and the grade will be absorbed into the presentation grade
  • No discussion report needed after the presentation
  • Only 10 of the highest scoring reviews will be counted for the grade
• Any alternative suggestions?
Paper Reviews

• Submit a 250-750 word summary of the paper including
  • Problem definition
  • Key contributions and solution overview
  • Evaluation summary
  • Strengths and weaknesses (limitations) of the technique(s)

• Let me know how you would extend the paper in 1-2 sentences (Can be vague and creative) => May be project ideas :)

Paper Presentation

- Presentations start on September 13th; assignments would be available today after the grading rubric is finalized.

- Week before: Meet instructor in person or virtually to discuss the presentation plan (compulsory!)
  - Use this time to ask questions and discuss the outline
  - Presentation slides are due when reviews are due for that class
  - Submit the final slides using the form on website
  - You can use office hours for this or schedule a meeting with me

- During the class: Be present in class either in person or virtually (compulsory!)
  - Deliver a 20-30 min presentation on the paper
  - Answer questions for the following 15 min
  - Final 30 min for open discussion on the paper (lead by the instructor)
Paper Presentation

• **After class:** Summarize the discussion of the paper within 250 - 750 words
  • Submit the summary within 2 days at the end of the class
  • Update the same form to submit the summary (edit the same entry)

• The presentation should include
  • Problem definition
  • Motivation: Why is this an important problem?
  • Outline the high-level solution
  • Illustrate the solution
  • Evaluation: What worked and what didn’t
  • Related Work: Put the solution in context of other research
  • Strengths and weaknesses
  • How would you extend this work?
Introductions!

• Let’s get to know each other a second time!

• No specific format
  • Name
  • Department
  • Advisor
  • Research Project (if any)

• Please drop by if you want to discuss exciting class projects (Email me before to check for availability)! Potentially leading to publication!
Recap

• Domain Specific Languages and Optimizations
  • XLA - Operator Fusion, Graph Rewrites
  • Graph Processing

• ML in Architecture
  • Branch Prediction
Lecture 5: Machine Learning Techniques
Types of Learning

- Supervised Learning (labelled data)
- Unsupervised Learning
- Semi-supervised Learning
- Reinforcement Learning
Types of Learning

• Supervised Learning
• Unsupervised Learning (unlabelled data)
• Semi-supervised Learning
• Reinforcement Learning
Types of Learning

- Supervised Learning
- Unsupervised Learning
- Semi-supervised Learning
- Reinforcement Learning

Learning using a small number of labelled data and a large number of unlabelled data

Community Detection  Node classification

Karate club graph, colors denote communities obtained via modularity-based clustering (Brandes et al., 2008).
Types of Learning

- Supervised Learning
- Unsupervised Learning
- Semi-supervised Learning
- Reinforcement Learning

No labelled data; learn from experience

State → Choose a “valid” action → New State

Iterate

Reward (Win / Loss)
Examples from Systems

- Supervised Learning - Performance models, Code completion tasks, etc.
- Unsupervised Learning - Large code models (Github Co-pilot)
- Semi-supervised Learning
- Reinforcement Learning - Code Optimization, Design Space Exploration
Machine Learning Simplified!

Data → Model → Prediction

- Images
- Program Code
- Sentences
- Robot State

Possibly a Neural Network
(A non-linear function with tunable parameters)

- Label
- Performance
- Translation
- Action
Perceptrons

\[ x_i \text{ - binary input / real input} \]
\[ w_i \text{ - real weights} \]
\[ \text{Output} \text{ - binary output} \]

Where’s the non-linearity?

Can only separate linearly separable regions

Michael Nielsen, “Neural Networks and Deep Learning”, Determination Press 2015
Add more layers and perceptrons?

Is it more powerful than a single perceptron?
Now it is non-linear; yes
Each layer makes decisions about high-level concepts

How do we set the weights?
Let’s devise an algorithm to learn them

Michael Nielsen, “Neural Networks and Deep Learning”, Determination Press 2015
Smooth Neurons

Instead of step function

Sigmoid Activations

Rectified Linear Unit Activations
Learning

• The process of learning weights of each neuron connection

• Use gradient descent since NNs are differentiable

\[ W_{i+1} = W_i - \eta \nabla F(W_i) \]

\( \eta \) – Learning Rate

\( F(.) \) – Neural Network Function

• Use better variants with better convergence properties (e.g. Stochastic Gradient Descent, ADAM)

Multilayer perceptrons

- Same as Feedforward fully connected neural networks
- Uses smooth activations to build a multilayer connection of neurons
How powerful are NNs?

- Neural Networks are function approximations (differentiable)

- Do you need plenty of hidden layers to achieve more capacity?
  - Theoretically No: Universal Approximation Theorem

  - Informally, it says one hidden layer is sufficient to approximate any continuous function

  - It does not say about learnability (how to set the weights)

- In practice, different network topologies with deeper networks are needed to learn better approximations for problems at hand
Quick Overview of Different NN topologies
Convolutional Neural Networks

• Used in the image domain and mimics convolution filters on parts of the image

• Learnable parameters are weights of these convolution filters

• Usually have multiple convolutional layers and max pooling in between
Convolutional Neural Networks
Convolutional Neural Networks

Input → Conv → Activation (ReLU) → Conv → Activation (ReLU) → Pool → Conv → Activation (ReLU) → Conv → Activation (ReLU) → Pool → FC → Output

Repeated
AlexNet (2012)

Figure 2: An illustration of the architecture of our CNN, explicitly showing the delineation of responsibilities between the two GPUs. One GPU runs the layer-parts at the top of the figure while the other runs the layer-parts at the bottom. The GPUs communicate only at certain layers. The network’s input is 150,528-dimensional, and the number of neurons in the network’s remaining layers is given by 253,440–186,624–64,896–64,896–43,264–4096–4096–1000.

Krizhevsky et. al “ImageNet Classification with Deep Convolutional Neural Networks”
What is a convolution?
What is a convolution?

Input

Weighted sum of input values \( \sum W_i x_i \)

Output
What is a convolution?

Weighted sum of input values \( \sum W_i x_i \)

- 3D convolutions
- Depth-wise convolutions (grouped)
- Dilated Convolutions
- Padded Convolutions
Recurrent Neural Networks

- Neural network topology with history

\[ h_t = RNN(Vh_{t-1}, Ux_t) \]
Recurrent Neural Networks

• Main use case: when you need to remember across time steps

• Two main problems with training vanilla RNNs
  • Handling long term dependencies can be tricky
  • Vanishing or exploding gradients during training

• Two types of popular RNN cells that alleviate these problems
  • Long Short Term Memory cells and Gated Recurrent Units (https://colah.github.io/posts/2015-08-Understanding-LSTMs/)
Use case: Machine Translation

Sequence to Sequence Learning with Neural Networks
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 Front
Department of Computer Science
Colorado State University
Fort Collins, CO 80525
front@colostate.edu

Jonathan Byrd
Department of Computer Science
Colorado State University
Fort Collins, CO 80525
jbyrd@colostate.edu

Barek Sharati
Department of Computer Science
Colorado State University
Fort Collins, CO 80525
bshar@cs.colostate.edu

Asa Ben-Hur
Department of Computer Science
Colorado State University
Fort Collins, CO 80525
asa@cs.colostate.edu

Spectral Clustering with Graph Neural Networks for Graph Pooling

Filippo Maria Bianchi 1, Daniele Grattarola 1, Cesare Alippi 1,2,3

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

Diagram:
- Neighborhood
- Message
- Aggregation
- Update
Graph Convolutional Network (GCN)

Neighborhood = All single-hop

Message
- Fixed importance
  \[ d = \frac{1}{\sqrt{d_{ii}d_{jj}}} \]
- Learnable Weights
  \[ W(l) \]

Message

Aggregation
- Sum (fixed)

Update

Semi-Supervised Classification with Graph Convolutional Networks (ICLR 2017)
GraphSAGE

Generalization of GCN

**Neighborhood** = All single-hop

**Message**  
- Fixed importance  
- Learnable Weights $W(l)$
- Subsample nodes

**Aggregation**  
- $+, \text{pool, LSTM}$

**Update** (concatenate current node embedding)

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

Suitable for Inductive Tasks

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

Easily parallelizable and can distribute work

Attention Is All You Need (NeurIPS 2017)
Next Lecture

- Genetic Algorithms
- Reinforcement Learning basics
- Auto-tuning and Design Space Exploration
Any Questions?