Foundations of Generative AI and Reasoning Systems

Course Description

Generative AI is transforming how we learn, create, and build AI models and software—and this course provides a hands-on introduction to the science and algorithms behind today's Large Language Models (LLMs). You'll learn how transformer-based models are architectured, trained and deployed, then venture into the techniques that power real-world genAI applications: prompt engineering (including algorithmic approaches), and post-training with reinforcement learning methods such as RLHF and direct preference optimization (DPO). The course also covers large reasoning models, in-context learning, and efficient adaptation via fine-tuning (LoRA), with a look at emerging foundation models beyond text (e.g., time-series). The course assignments require use of code assistants to develop imlementations for a variety of use cases and topics. Practical implementations by using modern software techniques and understanding of the science of LLMs are two main goals of the course.

Knowledge of predictive analytics at the level of IEMS 304 is required, in particular, classification algorithms such as naïve Bayes, SVM, logistic regression, and random forests.

This course will NOT teach about

- Classification algorithms
- Big data (too complex and messy)
- Fundamentals of Python (students are expected to know or learn by themselves the basics of Python)
    - Are they really needed?

Implementation for Homework Assignment

You are not allowed to use open source code that does not have standard python packages. In other words, you can only use modules that can be installed through pip/conda, poetry, or uv in your virtual environment.

You are not allowed to use or copy any code that is not part of a standard python package (read the previous paragraph).

You are encouraged and free to use code assistants (Claude Code, Codex, Cursor, Warp, etc) – duh (search what it means "eat your own dog food"). The code that you deliver is your product and you cannot blame code assistants for bugs, subpar performance, etc. The delivered code must also be 'production ready.' For example, there should not be repeated code, no hard coded parameters, no data in github, no tokens and other API keys in github, modular structure, logging with timestamps. We will deduct points for such deficiencies.

Every homework assignment must be delivered on github. You have to make the TA the collaborator on your repository.

Course Objectives

By the end of this course, you should be able to:

1. Finetune LLMs
2. Efficiently and algorithmically form prompts for LLMs
3. Claim that you are at least an intermediate (perhaps expert) software engineer with code assistants
4. Use reinforcement learning for LLM alignment

Course Outline

Weeks 1-2: Preliminaries

- Machine learning training
- Deep neural network basics
- RNN/LSTM concepts

Weeks 3: Transformers

- Attention
- Transformers
- Positional encoding

Week 4: Transformers for LLMs

- Training
- Long context
- ViT
- MoE
- The 3 phases
- Ethical considerations and challenges

Weeks 5-6: Prompt Engineering

- Basics of prompt engineering
- Techniques and best practices
- When to use prompt engineering
- Algorithmic prompt engineering
- Multi-modal aspects
- Coding with LLMs aka code assistants
    - Vibe coding
    - Ralph Loop

Week 7: Fine-Tuning and Low-Rank Adaptation (LoRA)

- Introduction to fine-tuning
- Zero and few shot learning with LLMs
- Zero shot learning by in-context learning
- Fine-tuning by LoRA
- Applications and benefits of LoRA

Week 8: Reinforcement Learning

- Modeling
- Policy gradient
- Proximal policy gradient

Week 9: Large Reasoning Models

- Reinforcement learning with human feedback (RLHF)
- Concept of LRM
- Training of LRM
- Direct preference optimization (DPO)
- GRPO

Week 10: Agents and Foundation Models

- Understanding intelligent agents
- Designing and implementing agents
- Real-world applications

Week 11: Foundation Models

- Foundation models (use cases and availability)
- Time series foundation models