Mathematics and AI (Fall 2025): Jarod Alper
Lecture 5

Today's plan:

— Survey of applications of AI/ML to mathematical research problems:

— Survey of other Math AI Courses

— Implementing ML pipelines for feedforward neural networks

— Discussion of class projects

— Up next: other neural network architectures: *recurrent,* convolutional, graph, adversarial, … Then transformers.

AI for mathematical research

Nice survey: https://seewoo5.github.io/awesome-ai-for-math/

Survey of other MathAI-related courses

- Berkeley Math 270: A Survey of Deep Learning for Mathematicians
- Brown Math 1810C: Artificial Intelligence and Euclidean Geometry
- UW CSE543: Deep Learning
- UW CSE573: Introduction to AI
- Harvard CS 2881: AI Safety

Machine learning in python

Coding environments for python: VSCode (.py scripts or .ipynb interactive python notebooks), Juniper notebooks in browser

Computing:

- numpy: basic scientific computing for arrays (NumPy array), linear algebra, …
- scipy: additional scientific computing tools, sophisticated linear algebra, optimization, statistics
- matplotlib: scientific plotting
- pandas: data wrangling and analysis using DataFrame, which is essentially a table like an excel spreadsheet

Machine-learning:

- scikit-learn
- pytorch
- tensorflow

Implementing feedforward neural networks

- *Gradient descent algorithms*: SGD with moment, Adam, AdamW
- *Learning rate*: not unitless, depends on specific problem. In practice, learning rate schedulers are used that adjust the learning rate during training
- *Splitting data*: training data/testing data (around 80/20) or sometimes as train/validation/test (80/10/10)
- *Epochs*: This is the number of full cycles passing over the training data. Can vary depending on application but usually 1-100. Very different for llms, which stratify data by quality and where training may not even see every training datapoint
- *Batch size:* In each epoch, training data is split into batches of some size. Each batch corresponds to one gradient descent update.
- *Data representation*: The choice of how data is represented in a neural network is critical!

Implementing feedforward neural networks (cont.)

- *Data representation*: The choice of how data is represented in a neural network is critical!
- *Overfitting*: When neural networks memorize data rather than learning pattern.
- *Regularization*: penalize complexity, early stopping (stop training when the validation loss levels off, even if the training loss is improving) , dropout (randomly drop a proportion of the neurons during training.)