Report on Weekly Coding Exercises and Learnings

Anuj Yadav (24M2012)

February 15, 2025

1 Introduction

This report summarizes the coding exercises and key learnings from my weekly exploration of fundamental and advanced machine learning concepts. The journey covered topics ranging from gradient computation and backpropagation to neural network language models and transformer architectures. Each week's work provided valuable insights into numerical computation, optimization, and deep learning methodologies.

2 Week 1: Derivatives and Gradient Computation

The focus was on computing analytical and numerical gradients and implementing backpropagation using a custom computational graph.

2.1 Key Implementations

- **Analytical Gradients:** Derived and implemented gradients for a mathematical function using differentiation rules.
- **Numerical Gradient Approximation:** Used finite difference methods to verify analytical gradients, noting that the symmetric derivative provided better accuracy.
- **Backpropagation with a Computational Graph:** Implemented a `Value` class to support automatic differentiation.
- **Softmax and Negative Log-Likelihood (NLL) Loss:** Developed a classification loss function and verified correctness using PyTorch.

2.2 Learnings

- Numerical gradients serve as a useful verification tool for analytical solutions.
- Implementing backpropagation manually enhances understanding of deep learning frameworks.
- Softmax and NLL loss are fundamental in probabilistic classification.

3 Week 2: Trigram Language Model Implementation

The primary goal was to implement a trigram language model using a single-layer perceptron trained with softmax regression.

3.1 Key Implementations

- **Data Preprocessing:** Tokenized and structured textual data into bigram and trigram datasets.
- **Model Training:** Implemented a trigram-based softmax classifier with cross-entropy loss and gradient descent optimization.
- **Regularization Tuning:** Applied L2 regularization to mitigate overfitting.
- **Bigram vs. Trigram Model:** Compared predictive accuracy and loss trends.

3.2 Learnings

- Higher-order n-gram models capture better contextual dependencies in text.
- Regularization is crucial for preventing overfitting in language models.
- Softmax regression effectively models probabilistic word prediction.

4 Week 3: Neural Network Language Modeling

This week involved training a neural network to predict the next character in a sequence.

4.1 Key Implementations

- **Character-Level Model:** Mapped characters to embeddings, followed by a tanh-activated hidden layer and softmax output.
- **Mini-Batch Training & Learning Rate Decay:** Improved stability and convergence.
- **Experiments:** Investigated weight initialization effects, trigram integration, and batch normalization folding.

4.2 Learnings

- Proper weight initialization is essential to avoid vanishing gradients.
- Mini-batch training and learning rate decay improve optimization efficiency.
- Batch normalization stabilizes training but can be folded into inference layers.

5 Week 5: Advanced Transformer Implementation

Explored optimizations and modifications to transformer architectures for efficiency and improved language modeling.

5.1 Key Implementations

- **Vectorized Multi-Head Attention:** Parallelized self-attention computations to leverage GPU efficiency.
- **Training on Custom Datasets:** Adapted GPT training to the CNN/DailyMail dataset.
- **Pretraining & Fine-Tuning:** Demonstrated transfer learning by fine-tuning a pre-trained transformer on Shakespearean text.
- **LazyFormer Architecture:** Implemented a research-inspired optimization to reduce computational complexity.

5.2 Learnings

- Vectorized tensor operations significantly improve transformer efficiency.
- Pretraining on large datasets enhances fine-tuning performance on domain-specific tasks.
- Research-driven modifications can optimize computation without compromising model expressiveness.

6 Conclusion

The weekly exercises provided an in-depth exploration of key machine learning principles. Implementing gradient computations, language models, and transformers from scratch deepened my understanding of fundamental algorithms and deep learning architectures. Moving forward, I plan to explore hybrid transformer architectures and resource-efficient deployment strategies.