Parnal Sinha
L03: AWS MLU Reflection
ITAI 2376: Data Science in AI

AWS MLU Reflective Journal

Lab 1: Getting Started with PyTorch

Summary of Key Learnings

In Lab 1, I explored the fundamentals of PyTorch, including how to create and manipulate tensors, perform automatic differentiation, and execute basic linear algebra operations. I gained hands-on experience with tensors — understanding their indexing, slicing, and the importance of shape manipulation for feeding data into neural networks. Additionally, I practiced implementing automatic differentiation, a critical tool for training neural networks efficiently.

Insights & Understanding

I came to appreciate how PyTorch's dynamic computation graphs and native GPU acceleration facilitate rapid model development and experimentation. I was particularly intrigued by how tensor operations mimic NumPy but are optimized for GPU operations, making them highly scalable for deep learning tasks.

Challenges Encountered

Initially, I struggled with understanding tensor reshaping and ensuring data dimensionality matched across operations. However, running the provided activities and examining the output of each operation helped solidify my understanding.

Application & Relevance

The skills acquired in this lab — data manipulation and automatic differentiation — are directly applicable to all future AI projects. Whether developing custom models or modifying existing ones, understanding tensor operations and efficient computation is essential.

Code and Experimentation

I modified tensor shapes, experimented with indexing and slicing strategies, and validated the results of automatic differentiation manually to ensure comprehension.

Crucial Aspects Summary (Lab 1)

1. Explored tensor creation, reshaping, and indexing
2. Practiced element-wise and matrix operations
3. Implemented and verified automatic differentiation
4. Set computation device dynamically (CPU/GPU)

Parnal Sinha
L03: AWS MLU Reflection
ITAI 2376: Data Science in AI

Lab 2: Creating a Multilayer Perceptron and Dropout Layers

Summary of Key Learnings

Lab 2 advanced into building a minimal viable neural network — starting with logistic regression and progressing to creating a simple Multilayer Perceptron (MLP). The lab emphasized building the neural network architecture, initializing weights using Xavier initialization, selecting an appropriate loss function, and training the network with stochastic gradient descent (SGD).

Insights & Understanding

I gained an understanding of how neural network architecture choices (like activation functions and dropout) significantly influence model performance and generalization. The visualization of training loss over epochs clarified the relationship between hyperparameters and training stability.

Challenges Encountered

A primary challenge was managing data flow between the CPU and GPU. Early errors occurred due to tensor device mismatches (similar to the issue I encountered later in Lab 3), which required careful device management in code. Weight initialization and tuning learning rates also required several iterations to achieve stable training.

Application & Relevance

This lab provided me with critical foundational experience in defining, initializing, and training neural networks — skills applicable to both academic research and industry projects involving AI model development.

Code and Experimentation

I implemented an MLP using `nn.Sequential`, experimented with varying the number of hidden neurons and dropout rates, and monitored the effects on training convergence.

Crucial Aspects Summary (Lab 2)

1. Created logistic regression and progressed to an MLP
2. Implemented dropout layers to mitigate overfitting
3. Applied Xavier initialization for stable training
4. Addressed data device compatibility (CPU vs. GPU)
5. Visualized training loss progression

Parnal Sinha
L03: AWS MLU Reflection
ITAI 2376: Data Science in AI

Lab 3: Building an End-to-End Neural Network Solution

Summary of Key Learnings

Lab 3 involved developing a full machine learning pipeline using the Austin Animal Center pet adoption dataset. The lab covered data preprocessing (numerical, categorical, and textual features), data cleaning, feature engineering, neural network design, training, validation, and testing.

Insights & Understanding

This lab emphasized the real-world complexity of ML workflows — from cleaning and transforming heterogeneous data types to designing a neural network that could handle such inputs. I also gained deeper insights into the importance of:

1. Preprocessing pipelines (using `ColumnTransformer`)
2. Feature scaling and encoding
3. Text data vectorization

Additionally, I experienced firsthand the challenges of maintaining device consistency between data and models — running into and resolving device mismatch errors that could otherwise derail training.

Challenges Encountered

Aside from the CPU vs. GPU device mismatch issue, other challenges included balancing neural network complexity with training time and avoiding overfitting despite the relatively small dataset size.

Application & Relevance

Building an end-to-end solution closely mirrored industry workflows. The lab's combination of data preprocessing, neural network construction, training, validation, and performance evaluation reflects the full ML development lifecycle.

Code and Experimentation

I designed an MLP with two hidden layers of 64 neurons each, added dropout layers, applied Xavier initialization, and experimented with learning rates and optimizers. I validated the model on a hold-out validation set and evaluated final performance on a test set using classification metrics.

Crucial Aspects Summary (Lab 3)

1. Performed EDA and data cleaning (numerical, categorical, text features)

Parnal Sinha
L03: AWS MLU Reflection
ITAI 2376: Data Science in AI

2. Developed data pipelines for preprocessing using `Pipeline` and `ColumnTransformer`
3. Built a custom neural network architecture with dropout
4. Addressed and resolved GPU/CPU device mismatch during training
5. Evaluated model performance using standard classification metrics

References

1. AWS Machine Learning University (MLU) Documentation
2. PyTorch Official Documentation
3. Course Lecture Materials
4. Scikit-learn Documentation