Assignment: Multi-Label Classification with PyTorch Optimizer Comparison

Course: Scalable Data Mining (CS60001)

Dataset Description: Amazon-670K (TF-IDF Format)

Dataset Available here: Google Drive Link

You will be working with a large-scale multi-label text classification dataset derived from the Amazon-670K corpus. The dataset has been preprocessed into a sparse TF-IDF format and is provided in LibSVM format.

- **Number of Classes:** 670,091
- **Number of Training Samples:** 490,449
- **Number of Test Samples:** 153,025
- **Number of Features:** 135,909

Preprocessing Summary

- Raw texts are a concatenation of `title` and `content`.
- TF-IDF features are computed using `sklearn`'s `TfidfVectorizer`, with:
  - `vocabulary`: Predefined from the dataset
  - `stop_words='english'`: To remove common uninformative words
  - `strip_accents='unicode'`: For NFKD normalization
- Format: LibSVM, with label indices separated by commas and features in `index:value` format.
- All instances are normalized to unit length.

Assignment Instructions

Your task is to implement and evaluate a multi-label classifier in PyTorch using a linear model trained under different optimization strategies. Ensure that your code is optimized to run on machines with **8–16GB RAM**. Use batch-wise processing and avoid loading the entire dataset into memory.

PyTorch Implementation and Optimizer Evaluation (100 Marks)

Objective: Implement a multi-label classifier using PyTorch and evaluate three different optimizers across varying learning rates using appropriate loss.

a. Build a custom PyTorch `Dataset` class to read LibSVM-format data and load it in mini-batches.

b. Define a simple model:

- Input dimension: 135,909
- Output dimension: 670,091

c. Use appropriate loss function

d. Train your model using the following optimizer configurations:

1. **SGD (no momentum)**
   Learning rates: `0.1`, `0.01`, `0.001`
2. **SGD with Momentum and Nesterov Acceleration**
   Learning rates: `0.1`, `0.01`, `0.001`
   Momentum: `0.9`, `nesterov=True`
3. **Adadelta Optimizer**
   Learning rates: `0.1`, `0.01`, `0.001`

e. For each configuration, record:

- Training loss per epoch
- Test set evaluation using multi-label metrics:
  - Precision
  - Recall
  - F1-score

f. Visualize and compare:

- Plot training loss vs. epochs for each optimizer + learning rate.
- Report metric scores in a tabular format.

g. Analyze:

- Which optimizer performed best and why?
- How did learning rate affect convergence and final performance?

Evaluation Criteria (Total 100 Marks)

• Data loading and custom Dataset class	15
• Model architecture and loss function setup	15
• Implementation of all optimizers and learning rate configurations	30
• Metric evaluation and result visualization	20
• Analysis and clarity of findings	20

Deliverables

1. Python code (Preferably Jupyter Notebook) for the PyTorch model and training loop.
2. Plots of training loss and performance metrics for each optimizer.
3. A very brief written comparison of the optimizers and learning rate behavior in the Jupyter Notebook.

Submission

Submit your code on CSE Moodle with name "$Assgn1 < yourrollno > .zip$". Ensure that all code executes within the memory constraints of a standard 8–16GB RAM machine.