CSE 676-B: Deep Learning, Spring 2024

Instructor: Alina Vereshchaka

Assignment 0

From Data to ML and NN Models

Checkpoint: Feb 8, Thu, 11:59 pm

Due date: Feb 15, Thu 11:59pm

Our initial assignment is focused on reviewing data analysis techniques, basic machine learning methods and refreshing the basics of deep learning framework. This assignment also focuses on theoretical and practical skills relating to neural networks. It consists of three parts where you derivate and analyze the setup and practice dealing with various datasets and implement, train, and adjust neural network.

Part I: Data analysis, ML models & PyTorch [40 points]

Step 1: Data analysis & Pre-processing [10 points]

1. Select a real-world dataset from the source listed below. Requirements for the dataset:
    - Represent the real-world data
    - Contain at least 20k entries
    - Should be different from the one used in class

    The dataset should come from one of these resources:
    - Open Data Buffalo: https://data.buffalony.gov/
    - US Government's Data: https://www.data.gov/
    - Yahoo Finance: https://finance.yahoo.com/
    - Yahoo Webscope: https://webscope.sandbox.yahoo.com/

    Note: any datasets from Kaggle won't be considered for evaluation.
2. Provide main statistic about the dataset (e.g. number of entries, features)
3. Handle missing entries, if any. Possible solutions:
    - Drop rows with missing entries. If you have a large dataset and only a few missing features, it may be acceptable to drop the rows containing missing values.
    - Impute missing data. Replace the missing entries with the mean/median/mode of the feature. You can use K-Nearest Neighbor algorithm to find the matching sample.
4. Handle mismatched string formats, if any.

For example, in the penguins dataset "Species" feature might appear as "Adelie" or "adelie," both of which refer to the same penguin species. These variations should be standardized to a consistent format such as "Adelie" or "adelie" to ensure consistency.

5. Handle outliers, if any. Detect and manage outliers within the dataset.

   For example, in the penguins dataset, while flipper lengths typically fall within the range of [180 – 210], certain entries might exhibit values like [10, 30]. These can be considered outliers. Possible solutions:

   - Remove outliers. If there are just a few outliers is limited, you may eliminate the rows containing these outliers.
   - Impute outliers. Replace the outliers with the mean/median/mode of the feature.

6. Using any data visualization library (e.g. matplotlib, seaborn, plotly), provide at least 5 visualization graphs related to your dataset. You can utilize any columns or a combination of columns in your dataset to generate graphs. E.g. correlation matrix, features vs. the target, counts of categorical features vs. the target.

7. Identify uncorrelated or unrelated features.

   Unrelated or uncorrelated features can introduce confusion to your model and negatively impact its performance. You can compute the correlation matrix between the features and the target variable. Features with a low correlation coefficient should be identified and subsequently dropped from the dataset to enhance model performance.

8. Convert features with string datatype to categorical**.** Possible ways:

   - One-hot encoding, creating binary columns for each category, denoting their presence or absence. E.g., in the "Species" feature, "Adelie," "Chinstrap," and "Gentoo" become binary columns with "1" for presence and "0" for absence.
   - Label encoding assigns unique integers to distinct feature values, useful for ordinal relationships among categories. E.g., "Small" as 0, "Medium" as 1, and "Large" as 2 can represent a "Size" feature. However, it may introduce unintended patterns.
   - You can use OneHotEncoder from scikit-learn

9. Normalize non-categorical features (scale numerical variables to have zero mean and unit variance)

   a. Find the min and max values for each column.
   b. Rescale dataset columns to the range from 0 to 1
   c. You can use StandardScaler from scikit-learn or Normalize from PyTorch

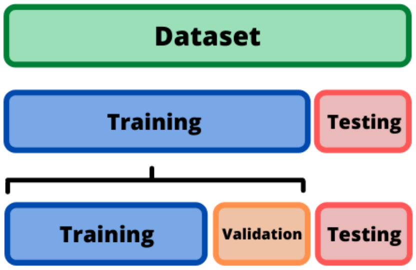
      Why do we do this? Normalization is to transform features to be on a similar scale. This improves the performance and training stability of the model

10. Choose your target Y and features ***X***

11. Split the dataset into training, testing and validation sets.

    You can use train_test_split from scikit-learn

Hint: first you can split the dataset into 'training' and 'testing' batches. Then take the 'training' batch and split it again for 'training' and 'validation'



Why do we need to split into training, testing and validation?

- Training set: used to train the model to learn the patterns or features in the data. It is important to have a large and representative training set so that the model can learn well.
- Validation set: used to tune the model hyperparameters and to prevent overfitting. Overfitting is when the model learns the training data too well and cannot generalize to new data. Hyperparameters are parameters that are not learned from the data, but are set by the user. By tuning the hyperparameters, we can improve the model's performance on the validation set.
- Test set: used to evaluate the final model's performance on unseen data. This gives us a good idea of how well the model will perform in the real world.

The commonly used splits of 70:15:15 or 80:10:10 are good starting points, but the optimal split ratio will vary depending on the size and characteristics of the dataset.

12. Print the shape of your X_train, y_train, X_test, y_test, X_validation, y_validation

Step 2: ML Models [10 points]

1. Apply ML algorithms (min 3 different algorithms) to model the target variable. This can be either a classification or regression task. Note:
   - You can use any of the libraries with inbuilt ML functions
   - The accuracy for any models used should be 65% or higher.
2. Provide a comparison of the results of different ML models you have used. This should be done in the form of graph representation and your reasoning about the results.

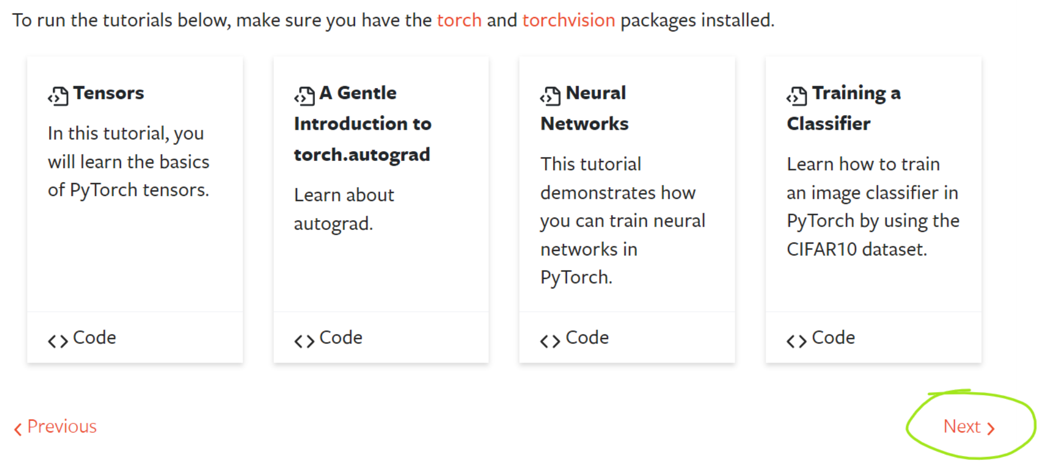
Step 3: Intro to PyTorch and Building a NN [15 points]

1. Complete Pytorch tutorial listed below. Use your own example values and hyperparameters, where applicable, while following the tutorial examples.

Deep Learning With Pytorch: A 60 Minute Blitz (https://pytorch.org/tutorials/beginner/deep_learning_60min_blitz.html)

Tips:

1. To get started -- click on the Next button on the bottom right



To run the tutorials below, make sure you have the torch and torchvision packages installed.

Tensors

In this tutorial, you will learn the basics of PyTorch tensors.

<> Code

A Gentle Introduction to torch.autograd

Learn about autograd.

<> Code

Neural Networks

This tutorial demonstrates how you can train neural networks in PyTorch.

<> Code

Training a Classifier

Learn how to train an image classifier in PyTorch by using the CIFAR10 dataset.

<> Code

2. Complete all 4 parts:
    - Tensors
    - A Gentle Introduction to torch.autograd
    - Neural Networks
    - Training a Classifier

   You can combine the results into one Jupyter notebook file. There is no need to include theoretical materials. Code with your values and clear section naming is sufficient.

3. Hyperparameters tuning.

   Provide the results of the neural network setup using your own THREE different sets of parameters (consider changing the number of layers, number of nodes, activation function, optimizer, etc.). While evaluation, accuracy will not be considered as a priority. The primary motivation is for you to explore how different NN setup influences the accuracy of the model.

4. Build a shallow NN and apply that to solve the problem you defined in Part I. An accuracy of 65% or higher will be considered for evaluation.
5. Save the model that you designed in Step 3 (e.g. https://pytorch.org/tutorials/beginner/saving_loading_models.html)
6. Compare and analyze the results.

In your report for Part I:

1. Provide brief details about the nature of your dataset. What is it about? What type of data are we encountering? Provide the main statistics about the entries of the dataset (mean, std, number of missing values, etc.)

2. What kind of preprocessing techniques have you applied to this dataset?
3. Provide at least 5 visualization graphs with a brief description for each graph, e.g. discuss if there are any interesting patterns or correlations.
4. Provide brief details and mathematical representation of the ML methods you have used. What are the key features? What are the advantages/disadvantages?
5. Provide your loss value and accuracy for all 3 methods.
6. Show the plot comparing the predictions vs the actual test data for all methods used. Analyze the results. You can consider accuracy/time/loss as some of the metrics to compare the methods.
7. Provide the neural network structure you have built to solve the problem defined in Part I. Show the plot. Analyze the results.

Note: You can partially reuse related implementations that you have completed for other courses with a proper citation, e.g. “Part I is based on the CSE 574 Machine Learning Assignment 1 submission by My Full Name”

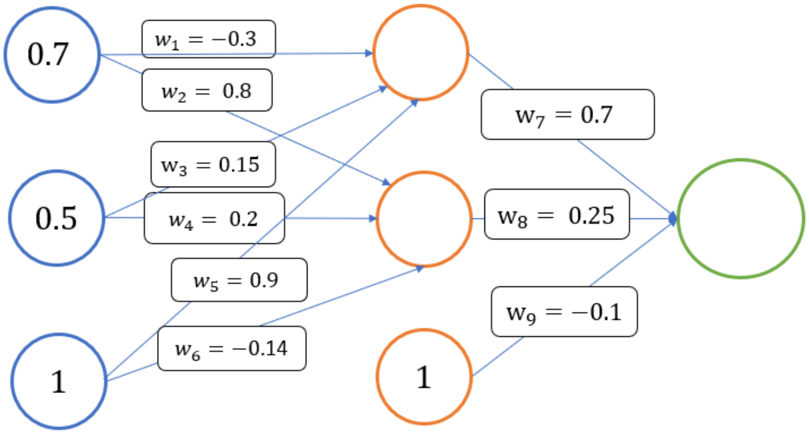
Part II: Deep Learning Theoretical Part [20 points]

1. Forward-backward Pass [15 points]

Consider the following neural network with initialized weights.

Given the following setup:

- Hidden layer activation function: ReLU
- Output layer activation function: Linear
- Learning rate: 0.03
- Target (y): 0.5



TASK:

1. Perform a forward pass and estimate the predicted output ($\hat{y}$)
2. Estimate the MSE
3. Find the gradient using back-propagation
4. Update the weights and the bias
5. Draw a computation graph for the forward and backward pass
6. Perform a forward pass to estimate the predicted output using the updated weights.
7. Estimate the MSE and compare the results with Step 2.

2. Derivative of Tanh [5 points]

The hyperbolic tangent (tanh) activation function has the following form:

$$f(x) = \tanh(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}}$$

TASK:

Prove that the derivative of Tanh has the following form:

$$f'(x) = 1 - f(x)^2$$

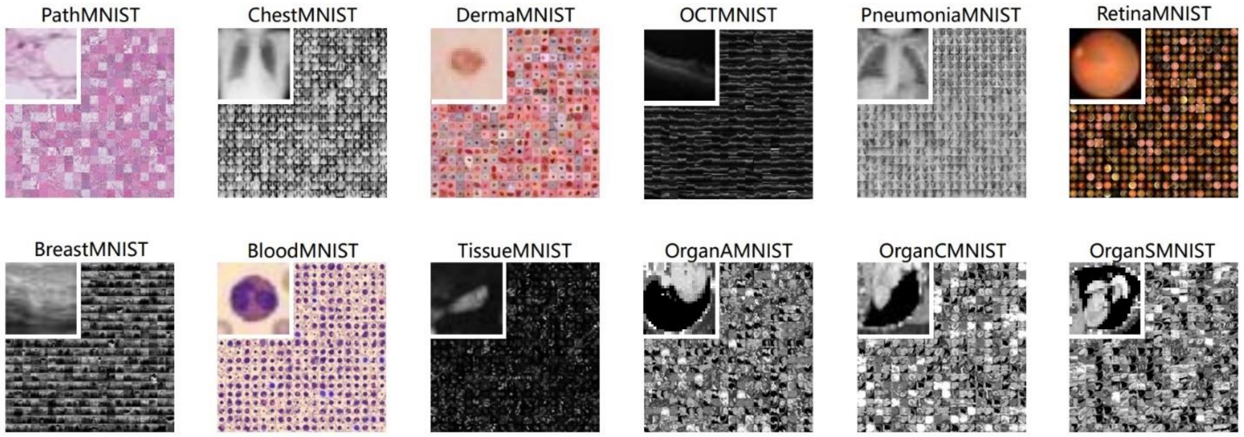
In your report for Part II:

1. For this part you need to submit only the report with your solutions. You can use docx, LaTex, or hand-written and scanned format. Convert the result to .pdf
2. Include the solution for both tasks as part of the report.

Part III: OCTMNIST Classification [40 points]

For this part, we will be working with a real-world dataset - OCTMNIST.

OCTMNIST



The OCTMNIST is based on a prior dataset of 109,309 valid optical coherence tomography (OCT) images for retinal diseases. Each example is a 28x28 image, associated with a label from 4 classes.

Getting the data:

- MedMNIST is a collection of multiple datasets, for this assignment we will be working with one dataset from the collection – OCTMNIST

- pip install medmnist

References:

- https://medmnist.com/
- https://github.com/MedMNIST/MedMNIST
- Direct download - https://zenodo.org/record/6496656

Steps:

1. Download the OCTMNIST 2D dataset and prepare it for training:
    a. Preprocess the dataset by normalizing the pixel values to a standardized range, typically between 0 and 1.
    b. Split the dataset into three parts: training set, validation set, and test set. The recommended split ratio is 70% for training, 15% for validation, and 15% for testing. Adjust the split ratio based on the size of the dataset and your exploration strategies.
2. Build a neural network:
    a. Design a model architecture that consists of at least three layers, including convolutional layers and fully connected layers. You can build an FC NN or CNN model.
    b. Ensure appropriate activation functions are applied after each layer to introduce non-linearity.
3. Apply following techniques to prevent overfitting and discuss each of them and how they impact the performance:
    a. Regularization: Apply regularization techniques, such as L1 or L2 regularization, to the model's parameters.
    b. Dropout: Introduce dropout layers between the fully connected layers.
    c. Early stopping: Monitor the performance of the model on a validation set and stop training when the validation loss stops improving.
    d. Expected min accuracy: 70%
4. Save the weights of the trained neural network that provides the best results. Check saving and loading models (PyTorch)
5. Discuss the results and provide relevant graphs:
    a. Report training accuracy, training loss, validation accuracy, validation loss, testing accuracy, and testing loss.
    b. Plot the training and validation accuracy over time (epochs).
    c. Plot the training and validation loss over time (epochs).
    d. Generate a confusion matrix using the model's predictions on the test set.
    e. Calculate and report other evaluation metrics such as precision, recall (scikit-learn), and F1 score (scikit-learn) to further analyze the model's performance on the test set.

Note:

- You can use any inbuilt functions or define some functions from scratch.
- All libraries are allowed, except those with pre-trained models or predefined architectures. Submissions with pre-trained models or predefined architectures will not be considered.

Bonus points [5 points]

Deploy the model

Deploy a model you defined in Part 1 or 3 on the server using any tools/methods.

Hint: You can refer to Jupyter Demo Time recording for Deploying deep learning model by Shubham Pandey

Bonus part submission:

- Create a separate folder named as Your_UBIT _assignment0_bonus
  e.g., avereshc_ assignment0_bonus
- You can duplicate code from your Part 1 or 3 if needed
- Include all the files needed in the folder
- You can deploy locally or directly on the server
- Make a 3-4 mins video recording showing how it works
- Share a link, that is publicly available
- Report is not required, you can include all the analysis as part of your .ipynb file

ASSIGNMENT STEPS

1. Submit checkpoint (February 8)

- Complete Part I & II of the assignment
- Include all the references at the end of the report. There is no minimum requirement for the report size, just make sure it includes all of the information required.
- Add .txt file “dataset_assignment0_part1.txt” and include a link that points to the dataset you have selected. Do not submit the dataset itself.
- Add all your assignment files in a zip folder including .ipynb files, the report, .h5 file and .txt at a zip folder. Name zip folder with all the files as
  Your_UBIT _assignment0_checkpoint.zip
  e.g., avereshc_ assignment0_checkpoint.zip
- Submit to UBLearns > Assignments

2. Submit final results (February 15)

- In your report or Jupyter notebook include all the references that were used to complete the assignment or the task. You can include that at the end of the report.
- Your Jupyter notebook should be saved with the results. If you are submitting python scripts, after extracting the ZIP file and executing command python main.py in the first level directory, all the generated results and plots you used in your report should appear printed out in a clear manner.
- Report (as a pdf file): combine the reports for all parts into one pdf file named as

  Your_UBIT _assignment0_report.pdf
  e.g., avereshc_ assignment0_report.pdf
- Code (as ipynb file with saved outputs): separate file for Part I and Part III named as

  Your_UBIT _assignment0_part_1.ipynb
  Your_UBIT _assignment0_part_3.ipynb
  e.g., avereshc_ assignment0_ part_1.ipynb
- Pickle or .h5 files with saved models that generate the best results for your model (Part III, step 4) named as

  Your_UBIT _assignment0_part_3.pickle
- For Bonus part, create a separate folder named as Your_UBIT _assignment0_bonus
- All assignment files and bonus folder should be packed in a ZIPfile named: Your_UBIT _assignment0.zip

  e.g. avereshc _ assignment0.zip
- Submit your ZIP file at UBlearns > Assignments > Assignment 0
- Verify that the submission was successful, e.g., download the submitted files and verify they are all correct
- You can make unlimited number of submissions and only the latest will be evaluated

Notes:

- Ensure that your code follows a clear structure and contains comments for the main functions and specific attributes related to your solution. You can submit multiple files, but they all need to be labeled with a clear name.
- Recheck the submitted files, e.g. download and open them, once submitted and verify that they open correctly

Assignments Grading

Checkpoint submission:

- Graded based on the 0/1 grade.
  - "1" is assigned, if more than 80% of the work has been fully completed for the checkpoint-related parts. "0" is assigned for all other cases
- Obtaining "0" for a checkpoint submission results in a fixed 30 points loss from the final assignment evaluation.

Final submission:

- Graded based on the X out of 100 points + bonus [if applicable]
- During the final evaluation, all the parts are evaluated, so please include a final version of all the parts of the assignment in your final submission.

Notes:

- Only files submitted on UBlearns are considered for evaluation.
  - Files from local device/GitHub/Google colab/Google docs/other places are not considered for evaluation
  - We strongly recommend submitting your work in-progress on UBlearns and always recheck the submitted files, e.g. download and open them, once submitted

Academic Integrity

The standing policy of the Department is that all students involved in any academic integrity violation (e.g., plagiarism in any way, shape, or form) will receive an F grade for the course. The catalog describes plagiarism as "Copying or receiving material from any source and submitting that material as one's own, without acknowledging and citing the particular debts to the source, or in any other manner representing the work of another as one's own.". Refer to the Office of Academic Integrity for more details.

Important Information

This assignment should be done individually.

- No collaboration, cheating, and plagiarism is allowed in assignments, quizzes, the final exam or final project.
- All the submissions will be checked using SafeAssign as well as other tools. SafeAssign is based on the submitted works for the past semesters as well the current submissions. We can see all the sources, so you don't need to worry if there is a high similarity with your Checkpoint submission.
- The submissions should include all the references. Kindly note that referencing the source does not mean you can copy/paste it fully and submit as your original work. Updating the hyperparameters or modifying the existing code is a subject to plagiarism. Your work has

to be original. If you have any doubts, send a private post on piazza to confirm.

- All group members and parties involved in any suspicious cases will be officially reported using the Academic Dishonesty Report form. What does that mean?
  - In most cases, the grade for the assignment/quiz/final project/midterm will be 0 and all bonus points will be subject to removal from the final evaluation for all students involved.
  - Final grade reduction (e.g. if you obtain B, that will result in B-)
  - Those found violating academic integrity more than once throughout their program will receive an immediate F in the course.

  Please refer to the Academic Integrity Policy for more details.
- The report should be delivered as a separate pdf file. You can combine report for Part I, Part II & Part III into the same pdf file. You can follow the NIPS template as a report structure. You may include comments in the Jupyter Notebook; however, you will need to duplicate the results in a separate pdf file.
- All the references can be listed at the end of the report. There is no minimum requirement for the report size, just make sure it includes all the information required.
- For the Bonus part, no report is needed.

Late Days Policy

You can use up to 5 late days throughout the course that can be applied to any assignment-related due dates. You do not have to inform the instructor, as the late submission will be tracked in UBlearns.

Important Dates

February 8, Thu, 11:59 pm - Checkpoint Submission is Due

February 15, Thu, 11:59 pm - Final Submission is Due