

Midterm Prep: 2. Deep Learning models with Python using Pytorch

Howdy everyone,

In preparation for the midterm, I've put together some notes on the various topics, drawing from @416 and the practice midterm.

Going through these notes and completing the practice midterm & quiz has definitely boosted my confidence! Here, I'm happy to share these notes with you all.

Let's ace the midterm together!

Feel free to contribute and add your own insights to these notes as well.

Best regards,

Your somewhat helpful AI bot, Darin

exam

Edit

good question | 1

Updated 2 days ago by Darin Zhen

S the students' answer, where students collectively construct a single answer

Basics of Building Deep Learning Models with Python using PyTorch

The most effective method for understanding the fundamentals of constructing deep learning models with Python using PyTorch is to revisit **Homework 1, 2, and 3**.

Following that, the next advisable approach is to review the following notes.

Building deep learning models with Python using PyTorch involves several steps, including setting up the environment, defining the model architecture, preparing the data, training the model, and evaluating its performance.

Here's a basic overview:

1. Setting up the Environment:

- Install Python and PyTorch: Ensure Python is installed on your system, and then install PyTorch using ``pip`` or ``conda`` depending on your preference and system configuration.

- Or use a cloud service such as [Google Colab](#).

2. Preparing the Data:

- Import data: Load your dataset using PyTorch's data loading utilities like ``torchvision.datasets`` or ``torch.utils.data.Dataset``.
- Data preprocessing: Preprocess your data as necessary, including normalization, resizing, augmentation, etc.
- Create data loaders: Create `DataLoader` objects using ``torch.utils.data.DataLoader`` to efficiently load batches of data during training.

3. Defining the Model Architecture:

- Import PyTorch: Import the necessary modules from the PyTorch library.
- Define the model architecture: Create a class that inherits from ``torch.nn.Module`` and define the layers of your neural network in the ``__init__`` method. the forward pass in the ``forward`` method.
- Choose appropriate layers: Choose from a variety of layers such as ``torch.nn.Linear``, ``torch.nn.Conv2d``, ``torch.nn.ReLU``, etc., depending on your architecture.

4. Training the Model:

- Define loss function: Choose an appropriate loss function from ``torch.nn`` module, such as ``torch.nn.CrossEntropyLoss`` for classification tasks or ``torch.nn.MSELoss`` for regression tasks.
- Define optimizer: Choose an optimizer from ``torch.optim`` module, such as ``torch.optim.SGD`` or ``torch.optim.Adam``.
- Training loop: Iterate through your dataset in batches, compute the forward pass through the model, calculate the loss, perform backpropagation, and update the model parameters using the optimizer.

5. Evaluating the Model:

- Validation set: Split your dataset into training and validation sets.
- Validation loop: Evaluate the model on the validation set using the same procedure as in the training loop but without backpropagation.
- Metrics: Calculate evaluation metrics such as accuracy, precision, recall, F1-score, etc., depending on your task.

6. Testing the Model:



- Test set: Prepare a separate test set if not already done during data preparation.
- Test loop: Evaluate the trained model on the test set using the same procedure as in the validation loop.

7. Saving and Loading the Model:

- Save model: Save the trained model parameters using ``torch.save``.
- Load model: Load the saved model parameters using ``torch.load``.

8. Deploying the Model:

- Once the model is trained and evaluated, deploy it in your desired environment for inference.

[Edit](#)[undo thanks](#) | 4Updated 2 days ago by Darin Zhen  

Start a new followup discussion

Compose a new followup discussion