Deep Learning & Machine Intelligence Task 3

Hesam Asadollahzadeh, Nesa Abbasi, Hadi Babalou Deadline: **17th Shahrivar 1402**

1. Introduction

Neural networks have become one of the most powerful machine learning structures in recent years. With the increase in processors' computing power, they have found many applications in various scientific and industrial fields. One advantage of these algorithms is the ability to construct any differentiable function using only two hidden layers of neurons. Therefore, it is possible to implement and train nonlinear functions and classifiers with their help. In this project, we will implement forward-feed neural networks for image classification.

2. Dataset Description

Fashion-MNIST is a dataset of Zalando's article images consisting of a training set of 60,000 examples and a test set of 10,000.

3. Phase One - Data preprocessing and visualization

The first step in any machine learning project is to observe, explore, and examine the data and the relationship between the features. For this purpose, do the following steps:

3.1. A Jupyter Notebook file has been provided for loading the dataset. Check out the first sample of the training data and see the image shape.

3.2. How many classes does the dataset have and what are they?

3.3. Select and display an image from each class as desired in the training dataset. For each image, display its label along with the image. (use matplotlib library)

4. Phase Two - Prepare DataLoader

4.1. torch.utils.data.DataLoader helps load data into a model. It turns a large Dataset into a Python iterable of smaller chunks. These smaller chunks are called batches or mini-batches and can be set by the *batch_size* parameter. What is the reason for using batches in the training process? Explain the advantages and disadvantages of very small and very large batch sizes.

4.2. What's the *shuffle* parameter of DataLoader? When should we use this parameter?

4.3. Create a DataLoader for our training and test sets with proper batch size. What's the effect of different batch sizes on model's performance and accuracy?

5. Phase Three - Data Classification

5.1. The *nn.Flatten()* layer took our shape from [color_channels, height, width] to [color_channels, height*width]. Explain why we have to do this.

5.2. To create your first neural network model, start by using *nn.Flatten()* as the input layer. You can then add hidden layers using *nn.Linear*, with non-linear activation functions like *relu* in between the layers. Experiment with different layer sizes, activation functions, and other hyperparameters to try to improve the model's performance.

5.3. To train our neural network model, we need to define a loss function and optimizer. As a starting point, you can use CrossEntropyLoss as the loss function and Stochastic Gradient Descent as the optimizer. However, we recommend exploring other options as well, such as different loss functions like MSELoss or KLDivLoss, and advanced optimizers like Adam or RMSprop. Experimenting with different losses and optimizers is important to find the best configuration for optimizing your model's performance on the given task. The goal is to continuously improve the model by tweaking the training process.

5.4. To train the model, create a training loop that goes through the following steps:

1. Iterate over a number of epochs.
2. In each epoch, loop through the training data in batches. For each training batch:

- Perform a forward pass through the model to make predictions
- Calculate the loss (e.g. cross entropy loss)
- Backpropagate to get gradients
- Update model weights using the optimizer

3. After each epoch, loop through the test data batches and:

- Perform a forward pass to get predictions
- Calculate the loss on the test batch
- Track overall test loss per epoch

This process of forwarding, calculating loss, and updating weights should be repeated for multiple epochs until the model converges. The test loss indicates how well the model generalizes.

5.5. Define an evaluation function that takes in a trained model, a DataLoader for the evaluation data, a loss function, and an accuracy function. The evaluation function should:

1. Put the model in evaluation mode.
2. Iterate through the DataLoader and pass each data batch through the model to make predictions
3. Compare the predictions to the ground truth labels and calculate the loss for each batch using the provided loss function
4. Calculate the accuracy between predictions and labels using the accuracy function
5. Return the average loss and accuracy across all batches in the DataLoader

Notes:

- It's recommended to use Jupyter Notebooks for these exercises, mostly because it displays DataFrames clearly.
- Feel free to ask any questions on Telegram/Skype!
- Please upload your responses in this Google form.

Good Luck!