EE-559 Deep learning – Practice 1, Students' questions

A question is denoted by **Q**; the corresponding answer is denoted by **A**. The questions-related exercises are marked by their numbers in Practice_1.pdf and Practice_1.ipynb documents.

Enviroment

Q: How to create an environment in Noto?
A: Instructions for environment creation are mentioned in the first cell of Practice_1.ipynb. To run the commands, you need to open the bash terminal in Noto, which you can do by going to Launcher $\rightarrow$ Other $\rightarrow$ Terminal.

Q: Do you provide an environment for local installation ?
A: No, we do not provide an environment for local installation. We recommend using Noto platform for week 1, and Gnoto platform for the following weeks, as the exercises have been developed and tested on these platforms.

Q: Are we going to use PyTorch for the labs?
A: Yes.

Q: I cannot read my dataset. The error says that the dataset is not found, even though my notebook is in the same repository as the dataset
A: Your working folder might not always coincide with the folder where your notebook is located. To check which folder you are currently in, you can use the `os.getcwd()` command and adjust your read path accordingly.

Neural Network

Q: Do we need activation functions between linear layers in exercise 1.4?
A: Yes, the activation function between linear layers is necessary. Otherwise, the composition of linear functions is just another linear function. For example, if $f(x) = ax + b$ – linear function and $g(x) = cx + b$ – linear function then the composition of these functions is $f(g(x)) = f(cx + d) = a * (cx + d) + b = acx + ad + b$ and is still a linear function. This linearity limits the network's ability to effectively capture complex data patterns. To introduce non-linearity and thus enable the modelling of intricate patterns, we incorporate specific non-linear functions like ReLU, Sigmoid, etc. These functions, though fixed, play a crucial role in transforming the network into a non-linear function. We will explore these functions in greater detail during week 2.

Q: How do we load the model? What is `model[0]`, `model[1]`?
A: To load the model in torch you can use `model.load_state_dict(dictionary_with_models_weights)`. `model[i]` is the way to select a specific i-th layer of the `nn.Sequential` model. For example, if the model is `nn.Sequential(nn.Linear(in_ch, out_ch), nn.Softmax())`, `model[0]` will refer to the linear layer and `model[1]` to a softmax. More details on saving and loading the weights can be found here.

Q: Does the nn.Linear layer have an activation function?

A: No, it is simply a linear transform that computes a weighted sum of the input features and adds a bias (if bias=True – default value).

Q: Do we need the (additive) bias in the linear layers?
A: Since the input data is usually shifted and not centred, the additive bias is necessary to remove the assumption that your input is centred around the origin. In nn.Linear() bias parameter is True by default, which makes the model learn an additive bias. You can exclude bias by setting nn.Linear(bias=False). nn.Linear documentation.

Q: Can we change the dimension in nn.Softmax(dim=1) to dim=0, for example?
A: Parameter dim identifies the dimension along which Softmax will be computed (so every slice along dim will sum to 1). In the current task, we need to transform logits (features of each x) into probabilities for each sample in x. Logits have size (number of samples, number of features), where the number of features is equal to the number of output classes. For each sample in x, we want to receive probabilities for each class, so we compute softmax along features in dimension 1. nn.Softmax documentation.

Q: When we call model(input) (which is equivalent to model.forward(input)) how does the system know that it has to go to the forward() definition inside the main class? If a new function is declared in the main class (say forward2) why does it not go there?
A: Every model class should be inherited from nn.Module. Let's say you want to create a class MyModelClass(nn.Module) and make and object model=MyModelClass(). The parent class nn.Module has two main functions that you have to redefine for your MyModelClass: __init__ and forward. Calling model(input) is equivalent to calling model.forward(input) or model.__call__(input). Additionally, parent nn.Module class has many functions that compute backwards pass, output parameters (.parameters()), load weights(.load_state_dict()) and so on, which means that the child class inherits those functions too. You do not have to implement any of those additional functions as they are implemented in the parent class and will automatically work correctly if you define __init__ and forward functions for your child class. The output of those functions depends on your implementation of __init__ and forward because they will be called implicitly. If you create a new function for your class that is not __init__ or forward and not defined in a parent class, let's say a function forward2(), it will be specific only for your MyModelClass. Calling model.forward2() will output the result of operations in forward2, but it will not replace the original forward() function. forward2() will be just another class function you can use. And you can create as many such functions of new names as you want. This all covers the concept of inheritance and overriding of parent functions with child functions in Python. A detailed explanation and documentation for inheritance can be found here. You can also look into the construction of models in pytorch and how classes in Python work.

Q: In exercise 1.7, for the l3 layer, why does using nn.Linear() cause an error, while nn.Sequential(nn.Linear()) does not?
A: nn.Sequential() is a container that chains multiple layers together in a sequential order. In this case, we are using it to wrap a single linear layer, making its functionality identical to using nn.Linear() alone. However, since the model loads pre-trained weights from a .pth file, it is crucial that our designed network matches the architecture expected by the saved

weights. Any mismatch in layer definitions can lead to errors when loading the weights.

Other questions

Q: What is a linear region?
A: A linear region is a region defined by a linear function. More details can be found in Section 3.1 Prince's book.