3️⃣

Neural net foundations

Learning steps

- [x] ~~yt video~~
- [x] ~~notebook/own implementation~~
- [x] ~~book chapter~~

Resources

- website 💻
    - lesson 3
- notebooks 🗒️
    - HuggingFace Spaces Pets repository
    - Which image models are best?
    - How does a neural net really work?
- book 📘
    - book chapter
    - solutions to exercises

Quick notes

YouTube video (→ link)

- introduction
    - there is a lesson 0 full of useful info
    - how to learn a lesson: watch video → run notebook → reproduce → run with different dataset
        - running the notebook in the clean version (no output, no text) is a great way to be sure to understand what is happening

        - 💡 do your own implementation of the image classifier using photos from Maïté, Snowy, and Comète
- available platforms
    - paperspace is a platform comparable to colab and jupyter notebooks
    - jupyter lab allows to have a terminal and notebook open
- making our pet classifier better
    - there are multiple image architectures available on pytorch (under the timm module)
        - they are all math functions ⇒ matrixes etc
    - important things are: how fast, how much resource needed, how accurate
        - we can see the comparison between models regarding speed and accuracy on the blog post on Kaggle
        - convnext models look nice
- what is inside the models
    - when you use the .predict() function, you get the different categories with the probabilities associated with each of them
    - the .pkg contains the dataloaders (or datablocks etc), and the architecture
        - when calling a variable containing the model, you can see the layers etc
        - ℹ️ you can use latex in python by using `'$ string $'`
    - what is the loss function i.e how do we improve the model
        - to define a quadratic function, we define a func(a, b, c, x) that returns the general formula of a quadratic
            - `partial` module allows to define values for a function → we fix the value of a, b, and c using the partial, then run the model by defining only x
            - we add some noise and plot some data that defines a quadratic on the graph
            - ❗ the seed fixes the random numbers to the same random values each time to keep the results the same
            - you can play around with the coefficients to find improvements for the fit between the graph and the points
            - we use a loss function to do the improvement of coefficients p.e. the mse
                - you can add the MSE to the plot to be able to understand the impact of the change in coefficients
                - lastly, we automate it by computing the derivative of the function (the gradient) → pytorch can do it automatically!
                - ℹ️ in pytorch, everything is a tensor (lists or arrays (1d-tensors, also called *rank 1 tensor*) of numbers , rectangles or tables (2d-tensors) of numbers, layers of tables (3d-tensors) of numbers, or more)
                - `requires_grad_()` allows to compute the gradients fur numbers used

        - you can then use this on a tensor, apply the quadratic function and compute the `loss.backward()`. then, you can print the gradients for each direction to know how each coefficient impacts the loss
    - to optimize more complex functions, we can use a *rectified linear unit* (see ReLu)
        - we can add more and more relu to have more squidly functions that can adapt to match any functions
- questions/answers
    - 💡 its better to use a fast and less efficient model at the start to make tries and retries as fast as possible (tweak parameters, datasets, ...) → use resnet34 for ex
    - to know if you have enough data, **train the model** and take a look at its accuracy
    - if accuracy is getting worse and worse, probably the learning rate is too high
- use data from kaggle competitions
    - normalize the data
        - use log for data like money that can be huge for some but really small for most
    - use 1 as const
    - 🔥 try the Titanic Kaggle competition

Exercises

- [x] ~~use "Getting started with NLP for absolute beginners" and take a look at the data~~
- [x] ~~apply notebook to another NLP competition (colab link)~~
- [ ] take part in Titanic competition
- [ ] follow "Iterate like a grandmaster!" notebook