Introduction to Machine Learning for Physicists

Problem Set 2
Lecturer: Fabian Ruehle
Github: https://github.com/ruehlef/ML_Oxford_Hilary2020/Problem_set_2/

2.1 Creating better training sets

Return to your network from Exercise 1.1 (e). We will modify the input to give a better training set

(a) First, normalize the input to have zero mean and unit variance. Then train your NN for different numbers of episodes. Has this helped training? Why/why not?

(b) Next, create a more balanced training set. Note that the class "neutral" occurs much less frequent than the others. Use any of the data augmentation techniques discussed in the lecture (oversampling, undersampling, cloning, SMOTE) to create a more balanced training set. How does this influence the performance?

2.2 Classify galaxies

In this exercise, we will design a CNN that classifies galaxies according to whether they are spiral, elliptical, or uncertain. We will be using data from the Galaxy Zoo project, in which volunteers have created labels for photos of galaxies from the SDSS. So we need to download the labeled data from Galaxy Zoo, and then the corresponding images from SDSS. Since I am not the owner of either (and both are rather large), you will have to do that yourself.

- Galaxy Zoo is part of the Zooniverse project. The Galaxy Zoo website can be found at https://data.galaxyzoo.org/. A csv file with the full catalog of the Galaxy Zoo 1 data release can be downloaded there. To get an idea of the data you just downloaded look at the first few lines of the csv file.
- As explained on the webpage, the data refers to the SDSS Data Release 7. We will use the RA (right ascension) and the DEC information to find the galaxy in the SDSS database (to download the images, we need to convert the numbers given here to degrees). Go to the SDSS webpage http://skyserver.sdss.org/dr7/en/tools/chart/list.asp of data release 7 to get an idea of what the images look like.
- I am not sure what the best way to get the images actually is. One way (I doubt it is the best) is to use the python script I provide (get_images.py), which uses the aspx page with ra and dec as get arguments to download the picture. If you want to use the script, you can run it with

  ```
  python get_images.py --dir /path/to/csv/file --max 5000
  ```

Here, `/path/to/csv/file` is the path to the csv file that you downloaded from Galaxy Zoo, and `5000` means that it will download only the first 5000 images (feel free to download more or less, depending on your patience). The more data the better, but the longer you will have to wait for the download and for training the CNN. The program will download the images into the same directory as the csv file, and create another csv file that will allow to match the images with the labels (spiral, elliptical, uncertain)

With these preparations, we are now ready to classify some galaxies. I provide again a template file and a solution file for the classifier (this time in PyTorch only). Feel free to start from scratch yourself, or to use any of the files (you might have to adapt the paths to fit the location of your files). Implement the following steps:

(a) Read in the file that was generated by get_images.py.

(b) Change the input features to have zero mean and unit variance.

(c) Use some data augmentation technique to create a balanced training set. I used symmetries (rotations) to upsample the under-represented classes.

(d) Perform a train : test split of 90 : 10.

(e) Set up some CNN architecture for classification. Use the architecture discussed in the lecture, i.e. convolutional layers with ReLU activation function, followed by max pooling. Put a few of these layer into a feed-forward NN, followed by one or two fully connected layers with some activation function and finally a softmax at the output. I used the following architecture:

- A convolutional layer with 6 filters, kernel size 8 and stride 2, followed by a (leaky) ReLU activation and a max pooling layer with kernel size 2 and stride 2
- A convolutional layer with 16 filters, kernel size 8 and stride 2, followed by a (leaky) ReLU activation and a max pooling layer with kernel size 2 and stride 2
- A fully connected layer with 120 nodes and a (leaky) ReLU activation
- A fully connected layer with 60 nodes and a (leaky) ReLU activation
- The output layer with 3 nodes and softmax activation

(f) Train the network for 10 epochs and evaluate its performance.