Machine learning lab 2: around neural networks

November 25, 2020

Objectives: train a neural network, visualize and understand learning curves, hyperparameter tuning, convolutional neural networks

1 First approach

Open the notebook keras_mnist.ipynb. The first cells should load the dataset, create a simple neural network and train it. Visualize the learning curves. Using that information, try to tune the main hyperparameters (learning rate, batch size), as well as the architecture of the network, to reach an accuracy of at least 90% on the test set.

2 A more challenging problem: CIFAR-100

Load the dataset with: `tf.keras.datasets.cifar100.load_data()` (even if you don't use keras afterwards, pytorch and tensorflow are also welcome).

This dataset contains images of 32x32 pixels and 3 color channels (so of shape (32, 32, 3)), representing instances of 100 classes such as plane, deers, kangaroos or clocks. One can of course measure the accuracy of a model in finding the good class, but since this is very challenging (and confusing a dog with a wolf can be forgivable, even for a human), we usually also look at the top-5 accuracy, which measures for each image if the correct class is among the 5 classes having the largest outputs according to the model. This metric is implemented in keras under the name `top_k_categorical_accuracy`.

Here the objective for you will be to train a model achieving at least 30% normal accuracy and at least 70% top-5-accuracy on the CIFAR-100 dataset (on validation data of course !). For this, a convolutional neural network will probably be needed, as well as some strategies to avoid overfitting (like dropout or data augmentation). Many resources are available for this, including chapters 3 and 4 of the datacamp course "Introduction to deep

learning with keras". I will ask that in the notebook you keep at least 2 non optimal networks (for example, one which is not convolutional, and one which overfits), explaining what was going wrong with them, as well as the final best scoring network. Given the heavy computational burden, I can strongly recommend using google colab.

3 Other project ideas

- Deep reinforcement learning's "hello world": cart-pole balancing (heavy in math, resources include this and this). Project would require to test different architectures of deep-Q learners on the task.
- Language generation with recurrent neural networks (see for a starting point this post). Project would require to choose a corpus of text, and train a model to generate similar text, as well as knowledge about recurrent neural networks. The course Recurrent Neural Netowrks for Language Modeling in Python on datacamp covers those subjects.
- Not necessarily deep learning-related: sentiment analysis on text data. A good dataset is the Yelp reviews dataset, and a starting point for inspiration is this notebook. Ideally, the project would involve comparing different models on the prediction task on this kind of datasets, and if possible add a component of unsupervised learning (like a low-dimensional visualisation of the data). It needs some familiarity with natural language processing tools as well. Chapter 4 of Introduction to Natural Language Processing in Python gives an introduction to how to build these kinds of models.