NNDL2024: Learning tasks and architectures
Session 3 (3 Apr)

Mathematical exercises

1. (20 pts) Convolutional layer as a special case of a fully connected layer.

   (a) Consider a problem where inputs $\mathbf{x} \in \mathbb{R}^D$ are processed by a single layer of M *one-dimensional convolutions* of width W, with no padding, no dilation (the convolution works on a contiguous set of inputs) and stride of one (that is, the inputs for the convolutions are shifted by only one element). **How many outputs and weights does this layer have?**

   (b) Characterize a *fully connected layer* that can perform the same computation. Do not think about the values of the weights, but just describe the structure of the layer. **How many neurons and weights does this layer have?**

   (c) **Invent an *explicit regularizer* for the fully connected layer that encourages solutions that behave like the convolutional layer would.** The regularizer should favor solutions that have the same sparsity structure and parameter sharing and penalize for all other solutions, by an amount that can be controlled by some regularization parameter(s).

2. (20 pts) Residual connections. Denote by $f_i(x) = w_i x + b_i$ a linear transformation of the input, so that $f_1(x)$ and $f_2(x)$ are both linear transformations but with different weights. Consider three alternative *residual blocks* (to be used a building blocks for a deeper network) with scalar inputs and outputs:

   1: $y = x + \mathrm{ReLU}\left(f_1(x)\right)$

   2: $y = f_1(x) + f_2\left(\mathrm{ReLU}(f_1(x))\right)$

   3: $y = x + f_2\left(\mathrm{ReLU}(f_1\left(\mathrm{ReLU}(x)\right))\right)$

   For each of these:

   (a) Draw a diagram that shows how the information is processed: A figure that starts from x, connects it with arrows to the next processing step etc.

   (b) Characterize the output space: What values the output can take for each of these, conditional on the input?

   (c) Write down the derivative $\frac{dy}{dx}$.

   Finally, comment on the alternatives. In which respects do they differ from each other? Would you prefer one over another? If yes, why?

Computer Assignments

1. (20 pts) Hyperparameter optimization. The goal of the exercise is to practice how tuning the optimizers and networks is done in practice.

   (a) Implement a neural network with the following structure for the MNIST data. Apart from the hyperparameters explicitly stated here, you can use the PyTorch's defaults. Note that part of the network structure, the convolutional kernel width, is left as a hyperparameter and you need to account for that in the code.

   - a convolutional layer with 16 output channels and the **kernel size given as a hyperparameter**,
   - ReLU activation,

- a max pooling layer with kernel size 2 and stride 2 (for reducing spatial dimensionality by two),
- a fully connected layer with 32 output features,
- ReLU activation,
- a fully connected layer with 10 output (for class probabilities),
- log softmax activation.

(b) Use the `Optuna` library for simultaneous tuning of

- Learning rate between $1e^{-6}$ and $1e^{-2}$
- Momentum between 0.0 and 1.0
- The kernel size amongst the set $3, 5, 7$

so that you only consider in total 75 candidate solutions. Start by reading some documentation of the library to understand how it is used, and then check the notebook that already implements most of the required parts.

(c) Implement grid search for the same three parameters, again covering in total 75 alternatives. Think about how to best allocate those to cover the three hyperparameters.

(d) **Report:** For both methods report the final optimal parameters, and also the progress of hyperparameter optimization as a function of round (the notebook already has plotting code for this). Brief analysis of the observations. How many hyperparameters you think you could tune simultaneously with these methods?

2. (20 pts) Transfer learning. The notebook provides a pre-trained digit classification model, trained on the MNIST data. Your task is to adapt this model to work well in a new classification problem where the inputs are images of same size, but the classes are now object categories (Fashion-MNIST data). You have only 128 training examples for this target task. However, we assume a large test set to understand the differences between the methods. Implement three different training protocols:

   (a) Training on the target data only: Ignore the pre-trained weights and train the model on the small target data data. Plot the training and test losses as a function of epoch.

   (b) Fine-tune the pre-trained source model on the target data so that you fix the weights of all other layers and only train the last layer. Plot the losses as before.

   (c) Fine-tune all weights of the pre-trained model on the new data. Again plot the losses.

   (d) **Report**: The final (based on your choice of convergence) training and test losses and classification accuracies for all three variants, and all three convergence curves drawn into a same plot. Discuss the results, both from the perspective of accuracy and computational time.

3. (20 pts) Few-shot learning. Use the same model and data as in the previous exercise. Read the `SimpleShot` few-shot learning approach from paper `https://arxiv.org/pdf/1911.04623.pdf`. Use the pre-trained MNIST model as the feature extractor and use few-shot learning to classify Fashion-MNIST images.

   (a) Write down the algorithm on a level of mathematical expressions (probably easiest with pen and paper). What is being computed, how is the classification decision made etc.

   (b) Implement the algorithm.

   (c) Evaluate the classification accuracy for $K = 1$ and $K = 7$ (K is the number of examples per class), using one of the proposed feature normalization strategies. Justify your choice of normalization based on the original paper.

   (d) **Report**: The classification accuracies for the two choices of K. Analysis of the results.