Exercise for MA-INF 2213 Computer Vision SS20
05.06.2020
Submission deadline: 17.06.2020
Neural Networks

1. **Convolutional Neural Networks**

   Install pytorch library for convolutional neural networks. The library provides a python interface. You may also need to install some dependencies such as `numpy` and `scipy`, if needed. In this task, you are requested to train a small CNN on the MNIST dataset for digit recognition. The dataset is very small, so even on a CPU the CNN can be trained quickly.

   (a) Implement CNN named 'Shallow Model' with the following architecture.
   2D CONVOLUTION LAYER ($outChannel = 10, kernelSize = 3, stride = 1$)
   2D CONVOLUTION LAYER ($outChannel = 20, kernelSize = 3, stride = 1$)
   MAXPOOL ($kernelSize = 2, stride = 2$)
   LINEAR LAYER ($NumClasses = 10$)
   (2 Points)

   (b) Implement CNN named 'Deeper Model' with the following architecture.
   2D CONVOLUTION LAYER ($outChannel = 10, kernelSize = 3, stride = 1$)
   2D CONVOLUTION LAYER ($outChannel = 20, kernelSize = 3, stride = 1$)
   MAXPOOL ($kernelSize = 2, stride = 2$)
   2D CONVOLUTION LAYER ($outChannel = 40, kernelSize = 3, stride = 1$)
   2D CONVOLUTION LAYER ($outChannel = 80, kernelSize = 3, stride = 1$)
   MAXPOOL ($kernelSize = 2, stride = 2$)
   LINEAR LAYER (200)
   LINEAR LAYER ($NumClasses = 10$)
   (2 Points)

   (c) Implement the training and testing procedure in the `main` function using cross entropy loss. Use Adam optimizer with a learning rate of 0.0001
   (3 Points)

   (d) Introduce Batch Normalization (only for 'Deeper Model') after each convolution layer. *(1 Point)*

   (e) Train the following networks for 10 epochs i) Shallow Model ii) Deeper Model and iii) Deeper Model with Batch Normalization. Share all the 3 model weights after training (ie. after 10 epochs) in your solution.
   (**NOTE**: No points will be awarded for this sub-question if the trained model weights are not uploaded with the solution) *(3 Points)*

   (f) Plot two graphs comparing the above three networks in terms of i) Loss on test data at various epochs (Testing Loss v No of Epochs) and ii) Accuracy on test data at various epoch (Testing Accuracy v No of Epochs). Upload the two graphs with the solution.
   (1 Point)

   Implement your solutions in python '*Sheet04.py*'. `TODO`s indicate where code has to be added. You can take help from Pytorch MNIST example on github.

2. **Cross entropy criterion and softmax output**

Neural networks are often trained according to the *cross-entropy* criterion rather than according to the squared error criterion. Consider a neural network with L layers $l = 1, \dots, L$. Let $\mathbf{y}^{(l)}$ denote the output of layer l and, for terms of simplicity, $\mathbf{y}^{(0)}$ the input to the network. Each layer has D_l units. Then, we have for the linear transformation in layer l

$$\mathbf{a}^{(l)} = {\mathbf{W}^{(l)}}^T \mathbf{y}^{(l-1)} + \mathbf{b}^{(l)}, \tag{1}$$

where $\mathbf{W}^{(l)} \in \mathbb{R}^{D_{l-1} \times D_l}$ are the weights and $\mathbf{b}^{(l)} \in \mathbb{R}^{D_l}$ is the bias of layer l. The output of layer l is then given by

$$\mathbf{y}^{(l)} = \sigma^{(l)}(\mathbf{a}^{(l)}) \tag{2}$$

with $\sigma^{(l)} : \mathbb{R}^{D_l} \mapsto \mathbb{R}^{D_l}$ being the non-linear function in layer l.

In the hidden layers, $\sigma^{(l)}$ is usually the element-wise sigmoid function

$$\sigma_i^{(l)}(\mathbf{a}^{(l)}) = \frac{1}{1 + \exp\left(-a_i^{(l)}\right)}, \quad l \in \{1, \dots, L-1\}, \tag{3}$$

but in the output layer, the softmax function is used:

$$\sigma_i^{(L)}(\mathbf{a}^{(L)}) = \frac{\exp\left(a_i^{(L)}\right)}{\sum_{d=1}^{D_L} \exp\left(a_d^{(L)}\right)}. \tag{4}$$

The natural training criterion for these kind of neural networks is the *cross-entropy* criterion, which aims at the minimization of

$$F(\Lambda) = \frac{1}{N} \sum_{n=1}^{N} F_n(\Lambda), \quad F_n(\Lambda) = -\log\big(p(c_n|\mathbf{x}_n)\big), \tag{5}$$

where $(\mathbf{x}_1, c_1), \dots, (\mathbf{x}_N, c_N)$ denotes the training data. The dimension of the output layer D_L equals the number of classes for the classification task and if $\mathbf{x}$ is the input to the network, $y_c^{(L)} = p(c|\mathbf{x})$.

(a) Assume a neural network with the above structure, i.e. a network with L layers, the sigmoid function in the hidden layers and softmax in the output layer. The training criterion is cross-entropy. Calculate the *error signals* of the output layer L,

$$\frac{\partial F_n}{\partial a_i^{(L)}} \quad (1 \leq i \leq D_L). \tag{6}$$

(2 Points)

(b) Now, calculate the error signals of each hidden layer,

$$\frac{\partial F_n}{\partial a_i^{(l)}} \quad (1 \leq l \leq L-1, \quad 1 \leq i \leq D_l). \tag{7}$$

(2 Points)

(c) Finally, give formulas for the derivatives with respect to the weights and biases,

$$\frac{\partial F}{\partial w_{ij}^{(l)}} \quad \text{and} \quad \frac{\partial F}{\partial b_j^{(l)}} \quad (1 \leq l \leq L) \tag{8}$$

using the error signals.
(2 Points)

3. **Meaning of the cross-entropy criterion**

   The squared error criterion aims to minimize (as the name says) the squared error of the network output compared to the ideal output defined by the training data. Which quantity is minimized/maximized by the cross-entropy criterion?

   (2 Points)

If you write neatly you may upload a scan of your solutions for the theoretical exercises. Of course, you may as well use LaTeX if you like.