Homework for The Deep Learning Class*

Artem Chernodub, Ph.D.
Ukrainian Catholic University, Faculty of Applied Sciences
Grammarly
chernodub@ucu.edu.ua

June 27, 2019 (updated)

1 Introduction

This homework is about convolutional neural networks. The goal is to study their internal structure, learning pipeline and work principles "under the hood". We are going to debug the mainstream neural model to solve the famous handwritten digits recognition problem, namely, MNIST.

The homework consists of two parts, theoretical and practical. The result of the theoretical part is a PDF document, the result of the practical part are several pieces of python source code for Pytorch. In the theoretical part, you are asked to write down mathematical equations for neural network's forward and backward flows. In the practical part, you are asked to code your equations on matrix level "from scratch" and achieve the same outputs as compared to standard Pytorch implementations. At the same time, the theoretical and practical parts are closely linked. You have to follow the source code prepared for the practical part when you are doing the theoretical part. For the first homework 60% of score are given, 40% for the second one. Also, in every homework there are 10% bonus points, but you can't get more than maximum score for this homework.

Deadlines for the first and the second parts are 17/06/2019, 9:00 AM CET and 15/07/2019, 9:00 AM CET, respectively. The penalty for missing the deadline: up to one week it is 50%, more than one week it is 100% of scores.

*No procrastination! Start today.

2 SimpleConvNet description

Consider the SimpleConvNet network shown on Fig. 1. This is a slightly modified basic MNIST example from Pytorch. Images are 28×28 grayscale images containing digits from 0 to 9.

The convolutional layer *conv_layer* receives input images packed to batches of size N_{batch}. It has 20 filters, $kernel\_size = 5$, $stride = 1$, $padding = 0$. The convolved features come to max-pooling layer $max\_pool\_2d\_layer$, here $kernel\_size = 2$, $stride = 2$, $padding = 0$. Then the tensor with data is flattened to a matrix by $reshape\_layer$ and forward propagated through the first fully connected layer $fc1\_layer$, rectified linear units $relu\_layer$ and the second fully connected layer $fc2\_layer$. Fully connected layers $fc1\_layer$ have 500 output units, $fc2\_layer$ 10 output units, respectively. Finally, the $softmax\_layer$, which implements the standard softmax function produces the network's output.

SimpleConvNet is trained by Stochastic Gradient Descent with momentum during 10-20 epochs. It achieves more than 98.5% accuracy on *MNIST test*. Batch size $N_{batch} = 64$, learning rate $\alpha = 0.01$, momentum $\mu = 0.5$. Please, check the source codes the for details.

2.1 Convolutional layer

Input: Tensor $\mathbf{a}$ of size $[N_{batch} \times C_{in} \times S_{in} \times S_{in}]$, where N_{batch} is the batch size, C_{in} is the number of input channels, S_{in} is the input image's size (here we consider a simplified case, squared images only).

Output: Tensor $\mathbf{z}^{(conv)}$ of size$[N_{batch} \times C_{out} \times S_{out} \times S_{out}]$, where N_{batch} is the batch size, $C_{out} = B$ is the number of output channels.

Convolutional layer's weights tensor $\mathbf{w}^{(conv)}$ has size $[B \times C_{in} \times K \times K]$ where B is number of filters, C_{in} is the number of input channels, K is the kernel size. Bias vector $\mathbf{b}^{(conv)}$ has size $[B]$. Output values $z^{(conv)}_{n,c_{out},m,l}$ are produced as follows:

$$z^{(conv)}_{n,c_{out},m,l} = \sum_{c_{in}=1}^{C_{in}} \sum_{i=1}^{K} \sum_{j=1}^{K} a_{n,c_{in},m+i-1,l+j-1} w^{(conv)}_{c_{out},c_{in},i,j} + b^{(conv)}_{c_{out}}, \tag{1}$$

where $1 \le n \le N_{batch}$, $1 \le m \le S_{out}$, $1 \le l \le S_{out}$, $1 \le i \le K$, $1 \le j \le K$ is output's image size.

2.2 Max-pooling layer

Input: Tensor $\mathbf{a}$ of size $[N_{batch} \times C_{in} \times S_{in} \times S_{in}]$.

Output: Tensor $\mathbf{z}^{(pool)}$ of size – ?
Output values $z^{(pool)}_{n,c_{out},m,l}$ are produced as follows:

$$z^{(pool)}_{n,c_{out},m,l} = max(a_{n,c_{out},2m-1,2l-1}, a_{n,c_{out},2m-1,2l}, a_{n,c_{out},2m,2l-1}, a_{n,c_{out},2m,2m}), \tag{2}$$

where $1 \leq n \leq N_{batch}$, $1 \leq m \leq S_{out}$, $1 \leq l \leq S_{out}$, S_{out} is output image size.

2.3 Reshape layer

Input: Tensor $\mathbf{a}$ of size $[N_{batch} \times C_{in} \times S_{in} \times S_{in}]$.
Output: Matrix $\mathbf{z}^{(reshaped)}$ of size $[N_{batch} \times ?]$.
Output values $z^{(reshaped)}_{n,j}$ are produced as follows:

$$z^{(reshaped)}_{n,j} = a_{n,c_{in},m,l}, \tag{3}$$

$$j = (c_{in} - 1) * S_{in} * S_{in} + (m - 1) * S_{in} + l, \tag{4}$$

where $1 \leq n \leq N_{batch}$, $1 \leq c_{in} \leq C_{in}$, $1 \leq m \leq S_{in}$, $1 \leq l \leq S_{in}$, $1 \leq j \leq N_{out}$.

2.4 Fully-connected layer

Input: Matrix $\mathbf{a}$ of size $[N_{batch} \times D]$, where N_{batch} is the batch size, D is the number of inputs.

Output: Matrix $\mathbf{z}^{(fc)}$ of size $[N_{batch} \times P]$, where P is the number of outputs.

Fully connected layer's weights matrix $\mathbf{w}^{(fc)}$ has size $[P \times D]$, bias vector $\mathbf{b}^{(fc)}$ has size $[P]$.

$$z^{(fc)}_{n,j} = \sum_{i=1}^{D} w^{(fc)}_{j,i} * a_{n,i} + b^{(fc)}_{j}, \tag{5}$$

where $1 \leq n \leq N_{batch}$, $1 \leq i \leq D$, $1 \leq j \leq P$.

3 Homework I, theoretical part (60% of the total score)

Homework I should be presented as a PDF document. I recommend using the LaTeX on-line service https://www.overleaf.com, MS Word equations works as well. A scanned document is also accepted, but the penalty for this is 25% of the Homework I score.

The list of tasks to be accomplished is provided below.

1. Write down the neural network's forward pass in scalar form using (1-5). Show it as an evolution of the input vector that passes layer-by-layer through the entire network to the output layer. You have to show the size of each vector or matrix. Indices in summation operators must also be specified (20% of the Homework I score).

2. Write down the neural network's forward pass in vector form:

   (a) for the convolutional layer described in section 2.1 on "moving window" level. Replace "moving window" algorithm by matrix multiplication. Please, use "im2col trick" (please, see here and here). Column's length in reshaped input matrix is $K * K$ (30% of the Homework I score).

   (b) For the pooling layer described in section 2.2 (10% of the Homework I score).

   (c) For the fully-connected layer described in section 2.4 (10% of the Homework I score).

   (d) (bonus task, up to +10% score for Homework I). Write down convolution as matrix multiplication adding parallelization on channel level (column's length in reshaped input matrix is $C_{in} * K * K$).

3. Write down the backward pass in scalar form. Derive the gradients $\frac{\partial Loss}{\partial w^{(conv)}_{c_{out},c_{in},m,l}}$ and $\frac{\partial Loss}{\partial b^{(conv)}_{c_{out}}}$ in (1). You don't need to differentiate the forward pass equations, please, use the "delta-rule" instead. Local gradient for the output softmax layer is:

$$\delta_k^{(out)} \equiv \frac{\partial Loss}{\partial z_k^{(fc2)}} = y_k - t_k, \tag{6}$$

   where y_k is k-th model's output, t_k is k-th target value. Local gradients δ for all intermediate layers must be shown (30% of the Homework I score).

4 Homework II, practical part (40% of total score)

The homework II should be presented as source codes on python3 for Pytorch 1.0.0 or higher. Please, use the python sources for Homework II (files *simple_conv_net_train.py* and *simple_conv_net_func.py*). Your goal is to replace standard Pytorch implementation by your custom methods on matrix level.

List of tasks to accomplish is provided below.

1. Implement SimpleConvNet's layer's *conv2d_scalar*, *pool2d_scalar*, *relu_scalar*, *fc_layer_scalar* (forward pass) in scalar form. Ensure that all layer's intermediate outputs are exactly the same as in Pytorch framework. You may use *diff_mse* function for this purpose (20% of the Homework II score).

2. Implement SimpleConvNet's layer's (forward pass) in vector form. Please, use your derived equations from Homework I.

   (a) for the convolutional layer *conv2d_vector* described in section 2.1 in matrix form. Column's length in reshaped input matrix is $C_{in} * K * K$. (up to 60% of the Homework II score).

   (b) For the pooling layer *pool2d_vector* (10% of the Homework II score).

   (c) For the fully-connected layer described in section 2.4 *fc_layer_vector* (10% of the Homework II score).

3. Train your network for 20 epochs, report the achieved accuracy on *MNIST test*. Measure and report the time on one epoch for scalar and vector variants (10% of the Homework II score).

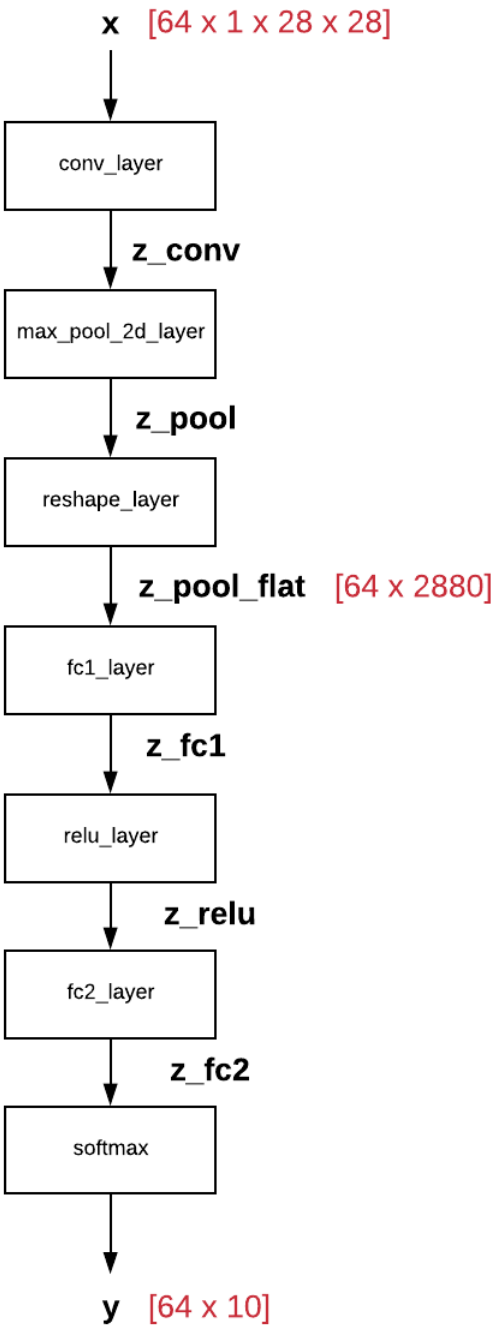

Figure 1: SimpleConvNet neural network.