Pytorch Basics (Score 10)

Homework 2 for Deep Learning, Autumn 2025

Deadline: 2025.11.3 12:00

Attention

- You need to submit all codes and a report (**in PDF format**).
- **Plagiarism is not permitted**.

1 Objective and Subjective Questions (Score 3)

These questions are all multiple choice questions. One or more options may be correct.

1.1 (score 0.5)

If we use softmax function as the classifier, can we use the MSE function as the loss function?

A. Yes

B. No

1.2 (score 0.5)

After layer decomposition, suppose the l-th layer is a sigmoid layer

$\boldsymbol{y}^{(l)} = \boldsymbol{f}(\boldsymbol{y}^{(l-1)})$

Then

$\boldsymbol{\delta}^{(l-1)} = \boldsymbol{\delta}^{(l)} \odot \boldsymbol{f}'(\boldsymbol{y}^{(l-1)})$

where $\boldsymbol{f}$ denotes a set of sigmoid functions. Is this claim correct?

A. Yes

B. No

1.3 (score 0.5)

After layer decomposition, suppose the output layer is L. In the Euclidean loss layer

$E^{(n)} = \frac{1}{2}||\boldsymbol{y}^{(L)} - \boldsymbol{t}||^2$

, where $\boldsymbol{y}^{(L)}$ denotes the output of the network and $\boldsymbol{t}$ denotes the label, we calculate ($\boldsymbol{f}$ denotes the activation function):

A. $\boldsymbol{\delta}^{(L)} = \boldsymbol{y}^{(L)} - \boldsymbol{t}$

B. $\boldsymbol{\delta}^{(L-1)} = \boldsymbol{y}^{(L)} - \boldsymbol{t}$

C. $\boldsymbol{\delta}^{(L-1)} = (\boldsymbol{y}^{(L)} - \boldsymbol{t}) \odot \boldsymbol{f}'(\boldsymbol{y}^{(L-1)})$

1.4 (score 0.5)

After layer decomposition, suppose the output layer is L. In the softmax-CE loss layer

$$E^{(n)} = -\sum_{k=1}^{K} t_k \ln f(y_k^{(L)})$$

, where $y_k^{(L)}$ denotes the output of the network and t_k denotes the label, we calculate (f denotes the softmax function):

A. $\boldsymbol{\delta}^{(L)} = \boldsymbol{y}^{(L)} - \boldsymbol{t}$

B. $\boldsymbol{\delta}^{(L-1)} = \boldsymbol{y}^{(L)} - \boldsymbol{t}$

C. $\boldsymbol{\delta}^{(L-1)} = (\boldsymbol{y}^{(L)} - \boldsymbol{t}) \odot \boldsymbol{f}'(\boldsymbol{y}^{(L-1)})$

1.5 (Score 0.5)

Compared to a fully connected layer, what are the two major properties of a convolutional layer?

A. Faster speed

B. Local connection

C. It introduces feature maps

D. Weight sharing

1.6 (Score 0.5)

Give at least two shortcomings for the MLP to do image classification.

2 Programming Practice (Score 7)

You need to use **Pytorch** framework to complete this practice. The report will take 1 score.

2.1 Tensor Operations-1 (Score 1)

Create two tensors (named A, B) of shape (5, 5) that are filled with random numbers, and complete the following operations:

- Create a zero tensor (named C) of the same shape and dtype as the tensors.
- Perform and print the result of the following operations of A and B: addition, subtraction, element-wise multiplication, and matrix multiplication.

2.2 Tensor Operations-2 (Score 1)

Create a tensor of shape (6, 4) filled with random numbers, and complete the following operations:

- Reshape it to be of shape (24,), and print the result.
- Extract and print a (4, 2) sub-tensor from the middle of the original tensor.

2.3 Autograd (Score 1)

Create a tensor of size (3, 3) with requires_grad=True. Perform following operations:

Multiple with the tensor [[1, 0, 2], [1, 1, 0], [0, 3, 1]] and sum up the elements of the resulting tensor.

Then, calculate and print the gradient of the operations with respect to the original tensor.

2.4 Observe Gradients (Score 1)

Learn the basic usage of torchvision.datasets.MNIST and torchvision.transforms using grad_cmp.ipynb , visualize the gradient comparison results across different network layers, and observe changes by modifying model parameters.

2.5 Simple Linear Regression Model (Score 2)

Implement a simple linear regression model with one input and one output. For this, you can define a model class that inherits from torch.nn.Module, and within this class define your model's architecture in the **__init__** method and how it processes input tensors in the forward method. The model should consist of **one linear layer (torch.nn.Linear)**. Then, generate some random data and labels, perform a forward pass of the data through the model, calculate **the mean squared error loss**, perform backpropagation using the backward method, and finally update the model weights with the help of **the SGD optimizer**.

After this training epoch, save its weights. Then create a new model of the same architecture, load the saved weights into it, and apply it to some data.