HW6 r08921a07 曾梓豪

Problem 1:

1.

I do two preprocessing to input data:

1. Standardize a dataset along the axis 0.
2. I used a second-order polynomial transformation to expand the input dimension.

following is the second-order polynomial transformation:

$$\Phi_2(X) = (1, x_1, x_2, \ldots, x_d, x_1^2, x_1x_2, \ldots x_1x_d, x_2^2, x_2x_3, \ldots x_2x_d, \ldots, x_d^2)$$

2.

I do the three experiment with the above preprocessing technique:

First, I adjusted the dimension of the hidden layer from 18 to 20, the loss in 100 epoch improved from 0.038 to 0.0032.

Second, I add an NN layer and an activation function in the middle, whose input dimension and output dimension are 18, the loss in 100 epoch is 0.0296988.

Third, I combine the two methods above, the loss in 100 epoch is 0.029600453.

I thought, the size of the dataset is too small to train a complex model, so I just adjust the hidden layer from 18 to 20 to get my best model.

3.

learning rates = 0.2
epochs = 100
optimizer = SGD
weight decay = 0
momentum = 0
all random seed = 892107

4.

Following is the structure of my best model:

```
linearRegression(
  (linear): Sequential(
    (0): Linear(in_features=15, out_features=20, bias=True)
    (1): ReLU()
    (2): Linear(in_features=20, out_features=2, bias=True)
  )
)
```

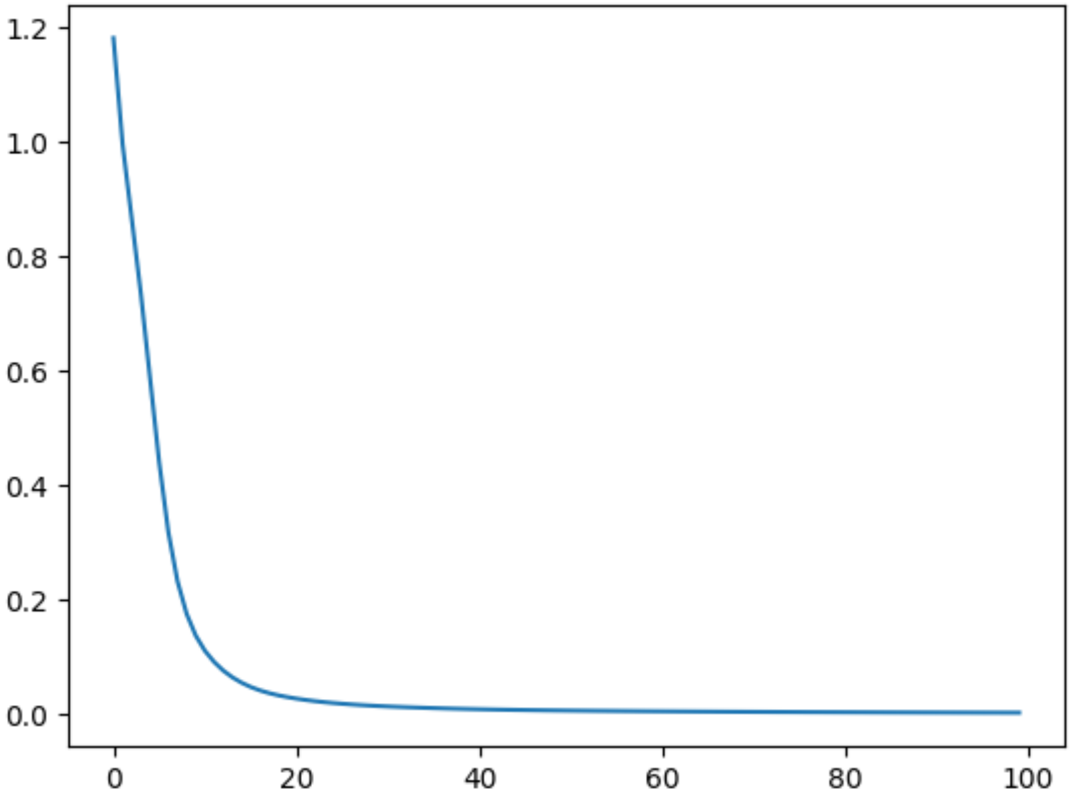
5.

Following is the prediction of testing data:

	A	B	C	D	E	F	G
1	No.	x1	x2	x3	x4	y1	y2
2	281	10.8	0.7	0.7	4.8	0.6868	-0.4395
3	282	5.4	12	-1.6	12.6	-0.5428	0.6969
4	283	-5	-12.1	21.4	-9.1	0.3374	-0.4075
5	284	-10.2	11.5	-4.3	0.8	-0.4482	0.2632
6	285	-0.9	2.1	-4.8	-2.7	0.6284	-0.5994
7	286	-1	-14.1	-19.2	0.5	-1.2243	1.1324
8	287	-10.3	-1.1	-5.2	6.8	0.2833	-0.5338
9	288	2.4	5.7	8.1	-6	0.6831	-0.6428
10	289	-25.9	-10.5	-1.5	-6.4	0.0554	-0.4293
11	290	-2.9	12.2	11.1	9.6	-0.2629	0.368
12	291	11.7	13.3	-8.1	9.6	-0.8615	1.1455
13	292	17.2	-1.1	8.6	-0.8	0.9023	-0.5463
14	293	-8.4	-15.3	4.6	-4.2	-0.8924	0.7669
15	294	-12.1	2	11.7	-0.3	0.7203	-0.9249
16	295	1.3	-14.1	-4.3	7.8	-0.9917	0.8946
17	296	-13.3	-1.7	11.7	-5.7	0.7673	-1.0612
18	297	1.2	-7	-16.6	-1.2	-0.0534	0.0217
19	298	-8.1	4.6	7.9	12.9	0.4256	-0.4432
20	299	-7.3	-1	0.8	11.8	0.3374	-0.4765
21	300	10.3	-16	3.2	2.6	-1.0446	1.3274

6.

Following is the MSE learning curve:



Problem 2:

1.

learning rates = 0.05
epochs = 100
batch_size = 64
optimizer = SGD
weight decay = 0
momentum = 0
all_random_seed = 892107

2.

Following is the structure of my best model:

```
Model(
  (cnn): Sequential(
    (0): Conv2d(3, 16, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
    (1): BatchNorm2d(16, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
    (2): ReLU6()
    (3): Conv2d(16, 16, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), groups=16)
    (4): BatchNorm2d(16, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
    (5): ReLU6()
    (6): Conv2d(16, 16, kernel_size=(1, 1), stride=(1, 1))
    (7): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1, ceil_mode=False)
    (8): Conv2d(16, 32, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), groups=16)
    (9): BatchNorm2d(32, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
    (10): ReLU6()
    (11): Conv2d(32, 32, kernel_size=(1, 1), stride=(1, 1))
    (12): Conv2d(32, 32, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), groups=32)
    (13): BatchNorm2d(32, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
    (14): ReLU6()
    (15): Conv2d(32, 32, kernel_size=(1, 1), stride=(1, 1))
    (16): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1, ceil_mode=False)
    (17): Conv2d(32, 64, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), groups=32)
    (18): BatchNorm2d(64, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
    (19): ReLU6()
    (20): Conv2d(64, 64, kernel_size=(1, 1), stride=(1, 1))
    (21): Dropout2d(p=0.2, inplace=False)
    (22): Conv2d(64, 64, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), groups=64)
    (23): BatchNorm2d(64, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
    (24): ReLU6()
    (25): Conv2d(64, 64, kernel_size=(1, 1), stride=(1, 1))
    (26): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1, ceil_mode=False)
    (27): Dropout2d(p=0.2, inplace=False)
    (28): Conv2d(64, 64, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), groups=64)
    (29): BatchNorm2d(64, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
    (30): ReLU6()
    (31): Conv2d(64, 64, kernel_size=(1, 1), stride=(1, 1))
    (32): Conv2d(64, 64, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), groups=64)
    (33): BatchNorm2d(64, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
    (34): ReLU6()
    (35): Conv2d(64, 64, kernel_size=(1, 1), stride=(1, 1))
  )
  (fc): Sequential(
    (0): Linear(in_features=12544, out_features=512, bias=True)
    (1): Dropout(p=0.5, inplace=False)
    (2): LeakyReLU(negative_slope=0.05)
    (3): Linear(in_features=512, out_features=64, bias=True)
    (4): Dropout(p=0.2, inplace=False)
    (5): LeakyReLU(negative_slope=0.05)
    (6): Linear(in_features=64, out_features=2, bias=True)
    (7): ReLU()
  )_
)
```

3.

train accuracy: 0.95049
valid accuracy: 0.94952

4.

Following is the MSE learning curve: