Machine Learning: Homework #3

Professor Shuang Liang

2252709
Xuanhe Yang

June, 2025

Problem 1

Question 1: Neural Network Theory
(1) Why do neural networks need activation functions?
Activation functions introduce non-linearity into neural networks, enabling them to express complex non-linear relationships. Without activation functions, each layer of the neural network would only perform linear transformations. No matter how many layers are stacked, the entire network would only be able to express linear models, which cannot solve complex problems.
Common activation functions include ReLU, Sigmoid, and Tanh, which help networks capture non-linear characteristics in data, thereby improving the model's expressive power.
(2) What is the impact of learning rate values on neural network training?
The learning rate determines the step size of parameter updates and has a significant impact on training effectiveness:
Learning rate too large:

- Parameter update steps are too large, potentially preventing model convergence
- May cause oscillations or divergence in the loss function

Learning rate too small:

- Parameter update steps are too small, leading to very slow convergence
- Training time increases significantly
- May get stuck in local optima

Appropriate learning rate:

- Loss function decreases gradually
- Model converges stably to global or near-optimal solutions
- Often optimized through learning rate scheduling (dynamic adjustment)

(3) What are the advantages of CNN over fully connected DNN in image classification?
Convolutional Neural Networks (CNN) have the following significant advantages over fully connected Deep Neural Networks (DNN):
Fewer parameters: CNN utilizes local connectivity and parameter sharing in convolution operations, significantly reducing the number of parameters to train. Compared to fully connected layers, CNN is more efficient and suitable for processing high-dimensional image data.
Spatial characteristics: CNN can automatically extract spatial structural features from images (such as edges and textures) without manually designing feature extractors. This makes CNN more suitable for processing image data.
Translation invariance: Through convolution and pooling operations, CNN has robustness to image transformations such as translation and scaling, effectively capturing global patterns of objects.
Better generalization ability: CNN's architecture aligns better with the characteristics of image data, enabling more efficient training and thus improving classification accuracy and generalization ability.

Problem 2

Question 2: AlexNet CONV1 Layer Calculation
Problem Description:
Input size: $227 \times 227 \times 3$
CONV1 parameters:

Problem 2 continued on next page...

- Number of filters: 96
- Filter size: 11×11
- Stride: 4
- Padding: 0

Calculate the output size of CONV1.
Solution:
The output size formula for convolutional layers is:

$$\text{Output Size} = \left\lfloor \frac{\text{Input Size} - \text{Filter Size} + 2 \times \text{Padding}}{\text{Stride}} \right\rfloor + 1$$

1. Calculate output width and height
For CONV1:

- Input size $= 227$
- Filter size $= 11$
- Stride $= 4$
- Padding $= 0$

Substituting into the formula:

$$\text{Output Size} = \left\lfloor \frac{227 - 11 + 2 \times 0}{4} \right\rfloor + 1$$

$$\text{Output Size} = \left\lfloor \frac{216}{4} \right\rfloor + 1 = 54 + 1 = 55$$

Therefore, both output width and height are 55.
2. Calculate output depth
The output depth equals the number of filters. With 96 filters, the output depth is 96.
Final Output Size:
CONV1 output size is: $\boxed{55 \times 55 \times 96}$

Problem 3

Question 3: Convolution and Pooling Operations
(1) Convolution Operation
A 4×4 feature map is convolved with a 3×3 convolution kernel (stride $= 1$).
Feature Map:

$$\begin{bmatrix} 1 & 2 & 3 & 0 \\ 0 & 1 & 2 & 3 \\ 3 & 0 & 1 & 2 \\ 2 & 3 & 0 & 1 \end{bmatrix}$$

Kernel:

$$\begin{bmatrix} 2 & 0 & 1 \\ 0 & 1 & 2 \\ 1 & 0 & 2 \end{bmatrix}$$

a) No padding:
Output size $= (4 - 3)/1 + 1 = 2 \times 2$
Calculations:

Problem 3 continued on next page. . .

- Position (0,0): $1 \times 2 + 2 \times 0 + 3 \times 1 + 0 \times 0 + 1 \times 1 + 2 \times 2 + 3 \times 1 + 0 \times 0 + 1 \times 2 = 15$
- Position (0,1): $2 \times 2 + 3 \times 0 + 0 \times 1 + 1 \times 0 + 2 \times 1 + 3 \times 2 + 0 \times 1 + 1 \times 0 + 2 \times 2 = 16$
- Position (1,0): $0 \times 2 + 1 \times 0 + 2 \times 1 + 3 \times 0 + 0 \times 1 + 1 \times 2 + 2 \times 1 + 3 \times 0 + 0 \times 2 = 6$
- Position (1,1): $1 \times 2 + 2 \times 0 + 3 \times 1 + 0 \times 0 + 1 \times 1 + 2 \times 2 + 3 \times 1 + 0 \times 0 + 1 \times 2 = 15$

Output (no padding):

$$\begin{bmatrix} 15 & 16 \\ 6 & 15 \end{bmatrix}$$

b) Zero padding to maintain output size:
Padding = 1 is required to maintain 4×4 output.
Output (with padding):

$$\begin{bmatrix} 7 & 12 & 10 & 2 \\ 4 & 15 & 16 & 10 \\ 10 & 6 & 15 & 6 \\ 8 & 10 & 4 & 3 \end{bmatrix}$$

(2) Pooling Operations
Given feature map:

$$\begin{bmatrix} 1 & 4 & 2 & 1 \\ 5 & 8 & 3 & 4 \\ 7 & 6 & 4 & 5 \\ 1 & 3 & 1 & 2 \end{bmatrix}$$

Using 2×2 pooling layer with stride = 2.
Output size = $4/2 = 2 \times 2$
Max Pooling:

- Top-left region: $\max(1, 4, 5, 8) = 8$
- Top-right region: $\max(2, 1, 3, 4) = 4$
- Bottom-left region: $\max(7, 6, 1, 3) = 7$
- Bottom-right region: $\max(4, 5, 1, 2) = 5$

Max pooling output:

$$\begin{bmatrix} 8 & 4 \\ 7 & 5 \end{bmatrix}$$

Average Pooling:

- Top-left region: $(1 + 4 + 5 + 8)/4 = 4.5$
- Top-right region: $(2 + 1 + 3 + 4)/4 = 2.5$
- Bottom-left region: $(7 + 6 + 1 + 3)/4 = 4.25$
- Bottom-right region: $(4 + 5 + 1 + 2)/4 = 3.0$

Average pooling output:

$$\begin{bmatrix} 4.5 & 2.5 \\ 4.25 & 3.0 \end{bmatrix}$$