ECE 661: Homework #1

Linear Model, Back Propagation and Building a CNN

Hai "Helen" Li

ECE Department, Duke University — Spring 2025

# Objectives

Homework #1 covers the contents of Lectures 02 ∼ 04. This assignment includes conceptual questions on the linear model, back propagation and convolutions, as well as lab questions involving trying out LMS algorithm and building and observing a CNN model.

The lab questions in this assignment don’t involve the training of deep learning models. So you are welcome to use your own computer to finish the labs. Please refer to the **NumPy/PyTorch tutorial** slides on Canvas for the environment setup on your computer.

**!**

**Warning: You are asked to complete the assignment independently.**

This lab has a total of **100** points plus 10 bonus points, yet your final score cannot exceed 100 points. You must submit your report in PDF format and your original codes for the lab questions through **Gradescope** before **11:55:00 pm, February 3**. You need to submit **three individual files** including:

1. *a self-contained report in PDF format* that provides answers to all the conceptual questions and clearly demonstrates all your lab codes, results and observations (figures and explanations);
2. *a single code file* used to produce all the results for Lab: LMS algorithms;
3. *a Jupyter notebook file* for Lab: Simple NN;

**Grading will be based solely on the pdf file. Other files will be used for plagiarism checking.**

# True/False Questions (10 pts)

For each question, please provide a short explanation to support your judgment.

**Problem 1.1 (2 pts)** On image recognition tasks, the convolution layers, compared to fully-connected layers, usually lead to better performance by exploiting shift invariant image features and typically have fewer parameters.

**Ans:** T, Convolutional layers are better because they exploit shift-invariant features in images, meaning they can detect the same pattern regardless of its location in the image. Convolutional layers have fewer parameters than fully-connected layers thanks to parameter sharing, where the same weights are reused across different positions in the image.

**Problem 1.2 (2 pts)** According to the “convolution shape rule,” for a convolution operation with a fixed input feature map, increasing the height and width of kernel size can not lead to the output feature maps in the same size.

**Ans:** T/F, Convolution layers have *· · ·* , fully-connected layers *· · ·* , so the answer should be *· · ·*

**Problem 1.3 (2 pts)** Given a learning task that can be perfectly learned by a Madaline model, this model is suitable for different weight initialization.

**Ans:** T/F, Convolution layers have *· · ·* , fully-connected layers *· · ·* , so the answer should be *· · ·*

**Problem 1.4 (2 pts)** The latency of a neural network measured on a specific processor is not always positively related to its theoretical FLOPS.

**Ans:** T/F, Convolution layers have *· · ·* , fully-connected layers *· · ·* , so the answer should be *· · ·*

**Problem 1.5 (2 pts)** The overfitting models can perfectly fit the training data. Theoretically, we should increase the diversity and variability in the training data or prune some of the nodes to improve NN’s generalization ability.

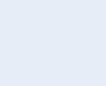
**Ans:** T/F, Convolution layers have *· · ·* , fully-connected layers *· · ·* , so the answer should be *· · ·*

# Adalines (15 pts)

In the following problems, you will be asked to derive the output of a given Adaline, or propose proper weight values for the Adaline to mimic the functionality of some simple logic functions. For all problems, please consider +1 as **True** and −1 as **False** in the inputs and outputs. **The answer of the proposed weight** *w***’s values should be within this set {-1, 0, 1}**.

**Problem 2.1 (3 pts)** Observe the Adaline shown in Figure [1](#_bookmark0), fill in the feature *s* and output *y* for each pair of inputs given in the truth table. What logic function is this Adaline performing?

Inputs +1



𝑤1 = 2

Σ

𝑤2 = 1

𝑠

Feature

**Sign()**

𝑤0 = 2

𝑥1

𝑤0 = −2

𝑥2

Output

𝑦

4

|  |  |  |  |
| --- | --- | --- | --- |
| 𝑥1 | 𝑥2 | 𝑠 | 𝑦 |
| -1 | -1 | | |
| -1 | +1 | | |
| +1 | -1 | | |
| +1 | +1 | | |

Figure 1: Problem 2.1.

|  |  |  |  |
| --- | --- | --- | --- |
| *x1* | *x2* | *s* | *y* |
| -1 | -1 | w0\*1 + w1x1 + w2x2 = 2\*1 + 2\*(-1) + 1\*(-1) = -1 | -1 |
| -1 | +1 | w0\*1 + w1x1 + w2x2 = 2\*1 + 2\*(-1) + 1\*(+1) = 1 | +1 |
| +1 | -1 | w0\*1 + w1x1 + w2x2 = 2\*1 + 2\*(+1) + 1\*(-1) = 3 | +1 |
| +1 | +1 | w0\*1 + w1x1 + w2x2 = 2\*1 + 2\*(+1) + 1\*(+1) = 5 | +1 |

The Adaline is performing the OR logic function, if we consider -1 as F and +1 as True.

**Problem 2.2 (4 pts)** Propose proper values for weight *w*0*, w*1 and *w*2 in the Adaline shown in Figure [2](#_bookmark1) to perform the functionality of a logic **NAND** function. Fill in the feature *s* for each pair of inputs given in the truth table to prove the functionality is correct. [**Hint:** The truth table of NAND function can be found here. <https://en.wikipedia.org/wiki/NAND_logic>]

A diagram of a function

Description automatically generated

The selected values for the weights are

w0 = 1

w1 = -1

w2 = -1

These values ensure that the Adaline correctly performs the NAND logic function, as demonstrated by the calculations in the table.

|  |  |  |  |
| --- | --- | --- | --- |
| *x1* | *x2* | *s* | *y* |
| -1 | -1 | w0\*1 + w1x1 + w2x2 = 1\*1 + (-1)\*(-1) + (-1)\*(-1) = 3 | +1 |
| -1 | +1 | w0\*1 + w1x1 + w2x2 = 1\*1 + (-1)\*(-1) + (-1)\*(+1) = 1 | +1 |
| +1 | -1 | w0\*1 + w1x1 + w2x2 = 1\*1 + (-1)\*(+1) + (-1)\*(-1) = 1 | +1 |
| +1 | +1 | w0\*1 + w1x1 + w2x2 = 1\*1 + (-1)\*(+1) + (-1)\*(+1) = -1 | -1 |

**Problem 2.3 (4 pts)** Propose proper values for weight *w*0*, w*1*, w*2 and *w*3 in the Adaline shown in Figure [3](#_bookmark2) to perform the functionality of a **Majority Vote** function. Fill in the feature *s* for each triplet of inputs given in the truth table to prove the functionality is correct. [**Hint:** The truth table of Majority Vote function can be found here. <https://en.wikichip.org/wiki/boolean_algebra/majority_function>]

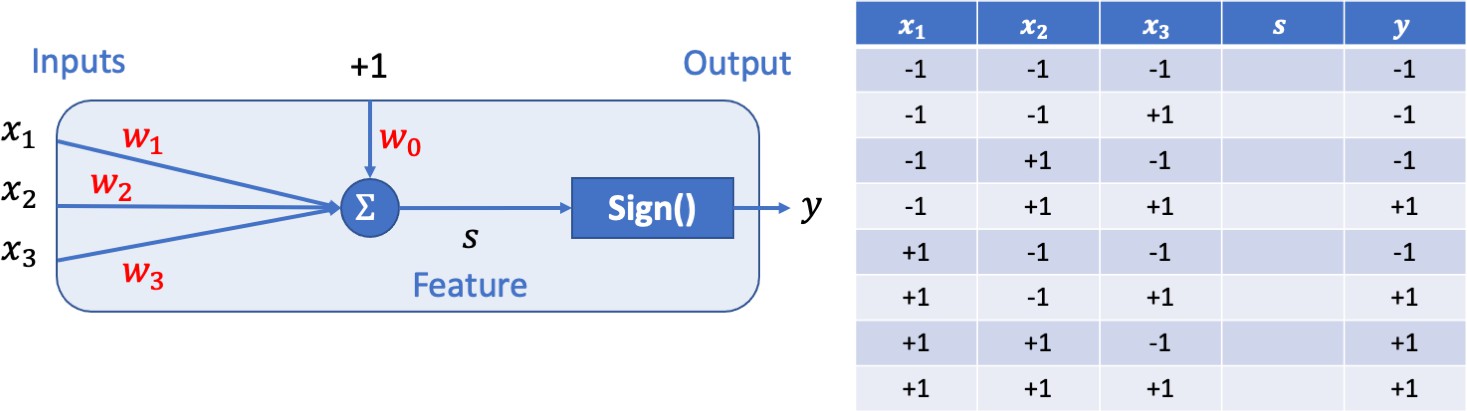


Figure 3: Problem 2.3.

The selected values for the weights are

w0 = 0

w1 = 1

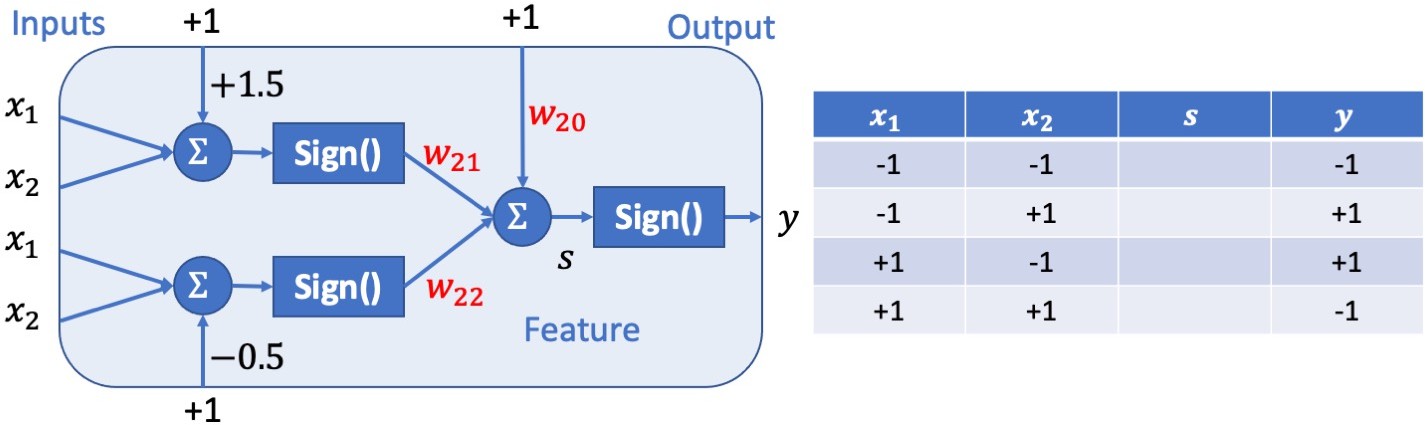
w2 = 1

w3 = 1

These values ensure that the Adaline correctly implements the Majority Vote logic function, as demonstrated by the calculations in the table.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| *x1* | *x2* | *x3* | *s* | *y* |
| -1 | -1 | -1 | w0 + w1x1 + w2x2 + w3x3 = (0)\*1 + (1)\*(-1) + (1)\*(-1) + (1)\*(-1) = -3 | -1 |
| -1 | -1 | +1 | w0 + w1x1 + w2x2 + w3x3 = (0)\*1 + (1)\*(-1) + (1)\*(-1) + (1)\*(+1) = -1 | -1 |
| -1 | +1 | -1 | w0 + w1x1 + w2x2 + w3x3 = (0)\*1 + (1)\*(-1) + (1)\*(+1) + (1)\*(-1) = -1 | -1 |
| -1 | +1 | +1 | w0 + w1x1 + w2x2 + w3x3 = (0)\*1 + (1)\*(-1) + (1)\*(+1) + (1)\*(+1) = +1 | +1 |
| +1 | -1 | -1 | w0 + w1x1 + w2x2 + w3x3 = (0)\*1 + (1)\*(+1) + (1)\*(-1) + (1)\*(-1) = -1 | -1 |
| +1 | -1 | +1 | w0 + w1x1 + w2x2 + w3x3 = (0)\*1 + (1)\*(+1) + (1)\*(-1) + (1)\*(+1) = +1 | +1 |
| +1 | +1 | -1 | w0 + w1x1 + w2x2 + w3x3 = (0)\*1 + (1)\*(+1) + (1)\*(+1) + (1)\*(-1) = +1 | +1 |
| +1 | +1 | +1 | w0 + w1x1 + w2x2 + w3x3 = (0)\*1 + (1)\*(+1) + (1)\*(+1) + (1)\*(+1) = +3 | +1 |

**Problem 2.4 (4 pts)** As discussed in Lecture 2, the XOR function cannot be represented with a single Adaline, but can be represented with a 2-layer Madaline. Propose proper values for second-layer weight *w*20*, w*21 and *w*22 in the Madaline shown in Figure [4](#_bookmark3) to perform the functionality of a **XOR** function. Fill in the feature *s* for each pair of inputs given in the truth table to prove the functionality is correct.



−1

+0.5

1

1

−1

+0.5

Figure 4: Problem 2.4.

The selected values for the weights are

w20 = +1

w21 = -1

w22 = -1

As:

|  |  |  |  |
| --- | --- | --- | --- |
| *x1* | *x2* | *s1* | *s2* |
| -1 | -1 | (0.5)\*1 + (-1)\*x1 + (+1)\*x2 | (0.5)\*1 + (+1)\*x1 + (-1)\*x2 |
| -1 | +1 | (0.5)\*1 + (-1)\*x1 + (+1)\*x2 | (0.5)\*1 + (+1)\*x1 + (-1)\*x2 |
| +1 | -1 | (0.5)\*1 + (-1)\*x1 + (+1)\*x2 | (0.5)\*1 + (+1)\*x1 + (-1)\*x2 |
| +1 | +1 | (0.5)\*1 + (-1)\*x1 + (+1)\*x2 | (0.5)\*1 + (+1)\*x1 + (-1)\*x2 |

|  |  |  |  |
| --- | --- | --- | --- |
| *x1* | *x2* | *s1* | *s2* |
| -1 | -1 | (0.5)\*1 + (-1)\*(-1)+ (+1)\*(-1) = 0.5 | (0.5)\*1 + (+1)\*(-1)+ (-1)\*(-1) = 0.5 |
| -1 | +1 | (0.5)\*1 + (-1)\*(-1)+ (+1)\*(+1) = 2.5 | (0.5)\*1 + (+1)\*(-1)+ (-1)\*(+1) = -1.5 |
| +1 | -1 | (0.5)\*1 + (-1)\*(+1)+ (+1)\*(-1) = -1.5 | (0.5)\*1 + (+1)\*(+1)+ (-1)\*(-1) = 2.5 |
| +1 | +1 | (0.5)\*1 + (-1)\*(+1)+ (+1)\*(+1) = 0.5 | (0.5)\*1 + (+1)\*(+1)+ (-1)\*(+1) = 0.5 |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| *x1* | *x2* | *Sign(s1 )* | *Sign(s2 )* | *s* |
| -1 | -1 | Sign(0.5) = +1 | Sign(0.5) = +1 | W20\*1 + w21\*(Sign(s1 ))+ w22\*(Sign(s1 )) |
| -1 | +1 | Sign(2.5) = +1 | Sign(-1.5) = -1 | W20\*1 + w21\*(Sign(s1 ))+ w22\*(Sign(s1 )) |
| +1 | -1 | Sign(-1.5) = -1 | Sign(2.5) = +1 | W20\*1 + w21\*(Sign(s1 ))+ w22\*(Sign(s1 )) |
| +1 | +1 | Sign(0.5) = +1 | Sign(0.5) = +1 | W20\*1 + w21\*(Sign(s1 ))+ w22\*(Sign(s1 )) |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| *x1* | *x2* | *Sign(s1 )* | *Sign(s2 )* | *s* | *y* |
| -1 | -1 | Sign(0.5) = +1 | Sign(0.5) = +1 | (+1)\*1 + (-1)\*(+1)+ (-1)\*(+1) = -1 | -1 |
| -1 | +1 | Sign(2.5) = +1 | Sign(-1.5) = -1 | (+1)\*1 + (-1)\*(+1)+ (-1)\*(-1) = +1 | +1 |
| +1 | -1 | Sign(-1.5) = -1 | Sign(2.5) = +1 | (+1)\*1 + (-1)\*(-1)+ (-1)\*(+1) = +1 | +1 |
| +1 | +1 | Sign(0.5) = +1 | Sign(0.5) = +1 | (+1)\*1 + (-1)\*(+1)+ (-1)\*(+1) = -1 | -1 |

1. **Back Propagation (15 pts)**

# A math equations and formulas Description automatically generated with medium confidence

# First, we have that:

# Where Ik is the identity matrix of size k x k

# Calculating ∂L/∂W2

# Calculating ∂L/∂b2

# Calculating ∂L/∂W1

# We have

# Let’s define z2 as

# We know from lecture slide 2, slides 29 and 30 that:

# Calculating ∂L/∂b1

# Finally, we have:

# A black and white text with black numbers Description automatically generated

# Calculating L

# Calculating ∂L/∂W2

# Calculating ∂L/∂b2

# Calculating ∂L/∂W1

# ∂x2/∂z changes because we are now using ReLU instead of sigmoid, so we have:

# Calculating ∂L/∂b1

# Finally, we have:

# A black and white text Description automatically generated

We have:

Strides = S = 1

K = 3

Padding = P = = 1

With padding of 1, we add zeros around the matrix, which results in the following padded input matrix to be used in the convolution operation:

# Kernel

# Example top left corner

# Convolution result of the top-left corner:

# (0)\*(0) + (-1/2)\*(0) + (0)\*(0) + (-1/2)\*(0) + (1)\*(0) + (-1/2)\*(0) + (0)\*(0) + (-1/2)\*(0) + (0)\*(-1) =0

# Final Output ([you can see the math here:)](https://docs.google.com/spreadsheets/d/1sH-CrE1iY1uzaZZMwrovozI65bm8eRyzhmtsrr2F3lo/edit?usp=sharing)

# 