

DIDACTIC TRANSFORMATION AND CLASSROOM MODULE FOR THE PERCEPTRON: WITH A FOCUS ON THE XOR PROBLEM

SEJUN OH^{a,*}

ABSTRACT. We develop a classroom module that introduces the perceptron through the XOR gate, a compact example that links secondary-school mathematics (coordinate geometry and systems of inequalities) to the idea that composing simple linear separators can create non-linear decision boundaries. The module is organized as three lessons and includes activity outlines, student-facing proof sketches for the limitation of single-layer perceptrons on XOR, and a minimal two-hidden-unit construction that realizes XOR. An optional computational lab visualizes a learned decision boundary of a small multilayer perceptron trained on XOR.

1. Introduction

Artificial intelligence (AI) has rapidly moved from an advanced topic to a literacy concern in school education, motivating the search for mathematically meaningful and age-appropriate entry points. Internationally, widely circulated K–12 AI guidelines emphasize core ideas such as learning from data and representation [1, 2]. In South Korea, the 2022 revised mathematics curriculum explicitly mentions elements of simple neural models such as perceptron weights and activation functions [3]. At the policy level, national strategies in multiple countries have also highlighted AI as a key educational and economic priority [4, 5, 6].

Presentations that foreground gradient-based learning can be misaligned with students' mathematical prerequisites, depending on elective-track choices. This paper therefore aims to transform the classical perceptron topic into a teachable module

Received by the editors October 23, 2025. Revised January 26, 2026. Accepted March 24, 2026.
2020 *Mathematics Subject Classification*. 97D40, 68T07.

Key words and phrases. Perceptron, didactic transposition, XOR problem, linear separability, multilayer perceptron, AI mathematics education.

*Corresponding author.

© 2026 Korean Soc. Math. Edu.



that is defensible within the curriculum and usable by teachers with heterogeneous student backgrounds.

We center the module on the XOR function. XOR is easy to state (a truth table on $\{0, 1\}^2$) yet it sharply exposes the expressive limitation of a single linear separator, a point emphasized in early analyses of perceptrons [9]. Because the impossibility proof uses only coordinate geometry and elementary inequalities, XOR offers a concrete link between familiar reasoning (lines and half-planes) and a foundational machine-learning idea: depth as composition. In classroom settings, the diagonal arrangement also helps surface a common intuition that “more training” can solve any classification task, motivating the need for a new representational structure.

The manuscript is organized as follows. Section 2 situates the module in a didactic framework and curriculum constraints. Section 3 presents the design method and a three-lesson sequence with optional computational extensions. Section 4 gives the mathematical validation of the module in a classroom-friendly form. Section 5 discusses adaptations, misconceptions, and instructional moves.

2. Background: didactic framing and curriculum constraints

2.1. Didactic transposition and design principles

Following Chevallard’s didactic transposition framework [11], we treat perceptron theory as *savoir savant*. Grounded in this perspective, we use the term *didactic transformation* to emphasize the design of a classroom module. We design a transformation into *savoir enseigné* by (i) selecting a minimal motivating problem (XOR), (ii) coordinating representations that are familiar to students (truth tables $\rightarrow$ points $\rightarrow$ lines/half-planes), and (iii) emphasizing constructive reasoning and classroom tasks over generality.

2.2. Educational validity grounded in prior research

The main outcome of this study is a teaching module, so the quality argument concerns the design of a usable instructional artifact. In educational design research, designed interventions are commonly discussed in terms of relevance/validity (fit to goals and theoretical rationale), consistency (internal coherence), practicality (feasible use in context), and effectiveness (learning impact), with the last category

requiring empirical classroom testing [16, 17, 18]. This article focuses on validity/consistency and practicality by aligning the module with the 2022 curriculum constraints, selecting a simple but mathematically decisive test case (XOR), and providing teacher-facing activity outlines with an optional computational visualization (Section 3.4). Classroom implementation data are left for future studies.

The module design is grounded in prior research on how students make sense of formal objects through coordinated representations and productive tensions. First, the representation ladder (truth table $\rightarrow$ points $\rightarrow$ lines/half-planes) follows the idea that mathematical understanding is supported by conversion among semiotic registers [12]. Second, the XOR “impossibility” moment is intentionally used as productive cognitive conflict: the failure of a single line creates a need for a new explanatory structure (composition/hidden layer), consistent with conceptual-change accounts [13]. Third, the optional computational component is designed to be novice-friendly by starting from a given artifact and encouraging controlled modifications before open creation, consistent with the Use–Modify–Create progression in computational thinking integration [14]. Finally, the focus on interpreting weights, boundaries, and depth supports AI literacy goals that emphasize understanding and critical interpretation of learning systems [1, 15]. Recent work in mathematics education has likewise argued for treating neural networks as meaningful mathematical learning objects when the content is didactically reduced and connected to accessible mathematics [19]. Empirical studies and development work in Korea suggest that short AI-mathematics modules that couple mathematical ideas with light-weight programming/feedback can support students’ engagement and value recognition [20, 21].

2.3. Curriculum placement and prerequisites (Korea 2022 revised curriculum)

We position the proposed module as flexible with respect to course selection in the 2022 revised curriculum [3]. The student-facing core can be taught with prerequisites limited to: (i) coordinate-plane representation of points, (ii) the equation of a line and half-planes, and (iii) basic manipulation of inequalities. These topics are commonly available by lower secondary or early upper secondary levels, although their timing can vary across elective tracks and local contexts. Section 5.3 provides brief guidance for classes with weaker algebra backgrounds and for classes that have not studied derivatives.

Table 1. Design principles, theoretical grounding, and operationalization in the proposed module.

Design principle	Grounding (examples)	Realization in this paper
Multiple representations	Semiotic representation theory [12]	A1 truth table $\rightarrow$ points; A2 line/half-plane; A3 diagonal argument
Productive conflict	Conceptual change [13]	A3 “structured failure” of a separating line for XOR
Scaffolded coding	Use–Modify–Create [14]	Section 3.4: run (use) $\rightarrow$ change hyperparameters (modify) $\rightarrow$ extend to a new Boolean task (create)
AI literacy	AI literacy competencies [1, 15]	Interpreting weights geometrically; depth as composition; interpreting a decision boundary plot
Korean implementation evidence	Korean AI-math material/intervention studies [20, 21]	three-lesson scope; prerequisite control; optional visualization for heterogeneous contexts

2.4. Two-level exposition

A recurring obstacle in teaching neural networks in secondary school is that learning rules are often presented via multivariable calculus. To keep the classroom narrative feasible, the main text avoids multivariable derivations and uses finite-difference intuition when discussing optimization. Supplementary material provides instructor notes that connect the classroom narrative to more formal statements (e.g., backpropagation as a structured application of the chain rule) without requiring the full formalism in student work.

3. Design method: a three-lesson classroom module

3.1. Learning trajectory (overview)

The module proceeds in three phases: (1) represent Boolean functions by points on the coordinate plane, (2) interpret a perceptron as a line-based classifier and identify why XOR fails, and (3) introduce composition (a hidden layer) and learning-as-optimization to recover XOR.

Three-lesson pacing (suggested). The activities can be organized into three 50-minute class periods as follows: (Lesson 1) Activities A1–A2; (Lesson 2) Activity A3 and the constructive resolution in Activity A4; (Lesson 3) Extension A5 and the optional computational visualization in Section 3.4, with a short written reflection/assessment.

3.2. Activity set (teacher-facing outlines)

Activity A1: From truth tables to points. Students convert a two-input truth table $f : \{0, 1\}^2 \rightarrow \{0, 1\}$ into labeled points in the plane. The central question is: *Can one straight line separate the points labeled 1 from those labeled 0?*

Activity A2: Construct OR/AND as perceptrons. Given a perceptron $y = H(w_1x_1 + w_2x_2 + b)$, students search for weights implementing OR and AND. They interpret $w_1x_1 + w_2x_2 + b = 0$ as a line and the sign as a half-plane.

Activity A3: XOR as a structured failure. Students attempt a separating line for XOR and discover the “diagonal” obstruction. This failure motivates a new mathematical tool: composition of two separators (see Figure 1). Possible teacher prompts include: (i) “Try to draw a separating line—what goes wrong for the two diagonal pairs?” and (ii) “If you move the line, can you ever keep both positive points on one side without capturing a negative point? Explain.”

Activity A4: Solve XOR with two hidden neurons (composition). Students build XOR from two linearly separable gates (e.g., NAND and OR) and verify a concrete 2-2-1 network that realizes XOR (Section 4.4).

Extension Activity A5: Linearizing XOR by feature augmentation. This optional extension reframes the XOR difficulty as a representation issue. Students construct an additional feature

$$x_3 := (x_1 - x_2)^2,$$

and observe that XOR becomes a linear separation problem in the augmented space (x_1, x_2, x_3) .

Proposition 3.1 (Feature linearization of XOR). *For $(x_1, x_2) \in \{0, 1\}^2$, let $x_3 = (x_1 - x_2)^2$. Then $\text{XOR}(x_1, x_2) = x_3$. Hence the plane $x_3 = 0.5$ separates the two XOR classes in (x_1, x_2, x_3) , equivalently $\hat{y} = H(x_3 - 0.5)$.*

Proof. The four cases are listed in Table 2; in each row the computed value of x_3 matches the XOR label. $\square$

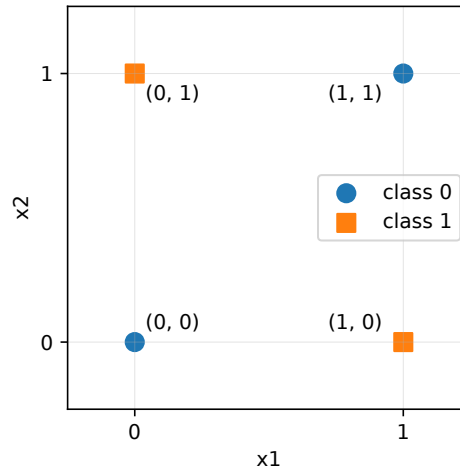


Figure 1. XOR data on $\{0, 1\}^2$: the positive points $(0, 1)$ and $(1, 0)$ lie on one diagonal and the negative points $(0, 0)$ and $(1, 1)$ lie on the other. Different markers indicate the two classes. No single line separates them.

Table 2. A feature augmentation that linearizes XOR.

x_1	x_2	$y = \text{XOR}(x_1, x_2)$	$x_3 = (x_1 - x_2)^2$
0	0	0	0
0	1	1	1
1	0	1	1
1	1	0	0

3.3. Assessment ideas (quick checks)

- Explain in one sentence what “linearly separable” means using the coordinate-plane representation.
- Given a truth table, predict whether a single-layer perceptron can implement it and justify with a sketch.
- Describe why adding a hidden layer can create a non-linear boundary even though each neuron uses a line.

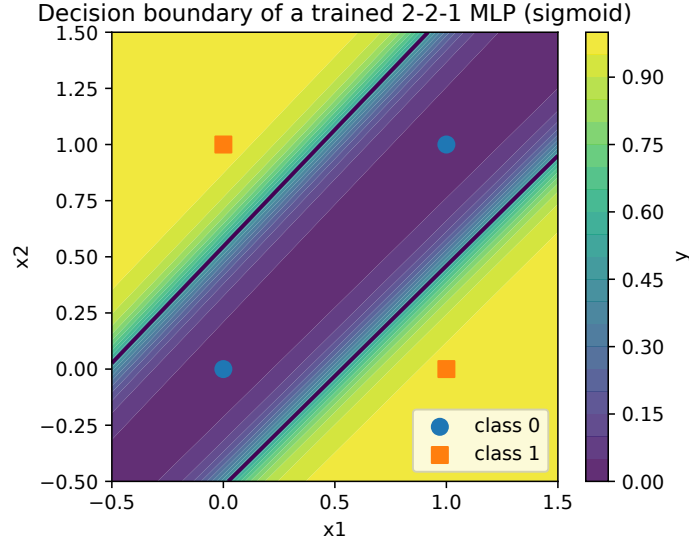


Figure 2. Decision boundary of a trained 2-2-1 multilayer perceptron with sigmoid activations on XOR. The contour indicates the 0.5 decision threshold; the model separates the diagonal classes by a non-linear boundary created through composition of two hidden units.

3.4. Optional computational visualization

When computers are available, students can run a short script to train a small multilayer perceptron (MLP) on XOR and visualize the learned decision boundary. Figure 2 shows a typical learned boundary; Figure 3 shows the corresponding training-loss curve (log scale). The reference script used to generate these plots (fixed seed, initialization scale 1.0, learning rate 0.5, and 10000 epochs) is provided in Supplementary Material C. For classroom comparison across groups, fixing the seed and keeping the learning rate/epoch count consistent helps keep outputs comparable.

4. Mathematical validation in classroom-friendly form

4.1. Perceptron as a linear separator and linear separability

A (two-input) perceptron computes

$$(1) \quad y = H(w_1x_1 + w_2x_2 + b), \quad (x_1, x_2) \in \mathbb{R}^2,$$

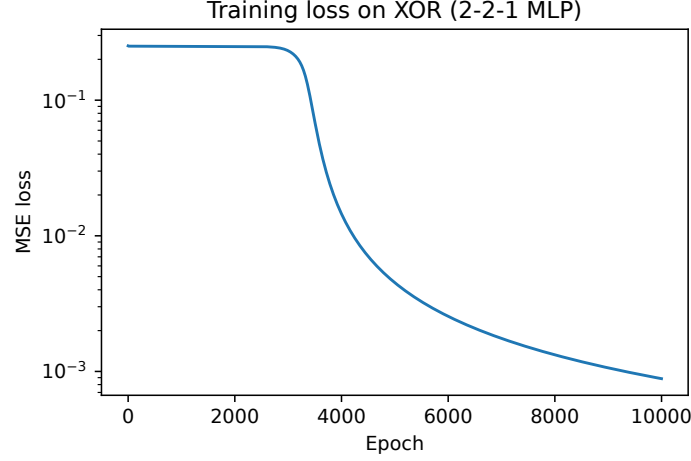


Figure 3. Training loss (mean squared error) during gradient descent on XOR for the 2-2-1 sigmoid MLP used to produce Figure 2 (log scale).

where $H(z) = 1$ if $z \geq 0$ and $H(z) = 0$ otherwise.

Definition 4.1 (Linearly separable). A labeled dataset $\{(x_i, y_i)\}$ with $x_i \in \mathbb{R}^2$ and $y_i \in \{0, 1\}$ is *linearly separable* if there exist (w_1, w_2, b) such that $y_i = H(w_1 x_{i,1} + w_2 x_{i,2} + b)$ for all i .

4.2. A minimal non-linearity proposition

Proposition 4.2 (Two-input Boolean functions). *Among the 16 Boolean functions $f : \{0, 1\}^2 \rightarrow \{0, 1\}$, exactly two are not linearly separable: XOR and XNOR.*

Proof idea. Plot the four inputs as the vertices of a square. Any linearly separable function must label one side of some line as 1 and the other side as 0. Exhausting the possible labelings shows that all cases except the diagonal patterns (XOR/XNOR) admit such a line. $\square$

4.3. The XOR impossibility theorem (central result)

Table 3 gives the XOR truth table, and Figure 1 shows the corresponding labeled points in the plane.

Theorem 4.3 (XOR impossibility). *No single-layer perceptron of the form (1) can implement the XOR function in Table 3.*

Table 3. XOR truth table.

x_1	x_2	$\text{XOR}(x_1, x_2)$
0	0	0
0	1	1
1	0	1
1	1	0

Proof sketch (inequalities). Assume $y = H(w_1x_1 + w_2x_2 + b)$ matches XOR. From $\text{XOR}(0,0) = 0$ we get $b < 0$. From $\text{XOR}(1,0) = 1$ and $\text{XOR}(0,1) = 1$ we get $w_1 + b \geq 0$ and $w_2 + b \geq 0$, hence $w_1 \geq -b$ and $w_2 \geq -b$. Adding gives $w_1 + w_2 \geq -2b$. But $\text{XOR}(1,1) = 0$ implies $w_1 + w_2 + b < 0$, i.e., $w_1 + w_2 < -b$. Since $b < 0$, we have $-2b > -b$, so $w_1 + w_2 \geq -2b$ contradicts $w_1 + w_2 < -b$. $\square$

4.4. A minimal two-hidden-neuron realization of XOR

A convenient constructive solution uses two hidden units that compute linearly separable gates, then combines them at the output unit. One concrete choice is:

$$h_1 = H(x_1 + x_2 - \tfrac{1}{2}) \quad (\text{OR gate}), \quad h_2 = H(-x_1 - x_2 + \tfrac{3}{2}) \quad (\text{NAND gate}),$$

$$\text{XOR}(x_1, x_2) = H(h_1 + h_2 - \tfrac{3}{2}).$$

This realizes XOR with a 2-2-1 network using only threshold units and elementary inequalities. The use of two hidden threshold units is minimal in this setting: with only one hidden threshold unit followed by a threshold output, the resulting classifier is either constant or equivalent to a single half-plane classification (possibly complemented), and Theorem 4.3 rules out such a representation for XOR.

4.5. Learning as optimization

For students, learning can be framed as choosing parameters to reduce a loss (misclassifications or squared error). Without derivatives, students can approximate “which way to move” a parameter by finite differences:

$$\frac{\partial L}{\partial w} \approx \frac{L(w + \Delta w) - L(w)}{\Delta w}.$$

Instructor notes that connect this intuition to a more formal gradient viewpoint are provided in Supplementary Material A.

5. Discussion and instructional implications

5.1. Mathematical connections

The module builds a coherent storyline from secondary mathematics: lines and half-planes (decision boundaries), systems of inequalities (proof of impossibility), and function composition (hidden layers creating non-linear boundaries).

5.2. Anticipated student difficulties and instructional responses

Several student difficulties may arise from the mathematical structure of the module. First, students may think that “more training” should eventually allow one neuron to solve XOR. In this case, teachers can return to Theorem 4.3 and the diagonal representation in Figure 1, emphasizing that the obstruction is representational rather than computational. Second, students may interpret a multilayer network as merely a larger linear classifier. Teachers can address this by highlighting that the hidden layer composes half-plane decisions and by using the non-linear decision boundary in Figure 2. Third, students may regard backpropagation as a mysterious algorithmic procedure. A useful response is to trace how one weight affects the final loss along a computation graph, as in Supplementary Material B.

5.3. Limitations and adaptation

The calculus-dependent treatment of learning should be adjusted to course selection. For classes with weaker algebra backgrounds, teachers may reduce the symbolic manipulation in the proof sketches and rely more on truth tables, plotted points, and guided line-drawing tasks before introducing the inequality notation. If students have not studied derivatives, teachers can keep the finite-difference intuition and qualitative “move-the-line” activities. If derivatives are available, teachers can enrich the module by connecting finite-difference slopes to gradients. Finally, this manuscript foregrounds the design and mathematical framing of a classroom module rather than reporting classroom implementation data; systematic evaluation of learning impact is best addressed through subsequent implementation studies.

6. Conclusion

We presented a didactic transformation of the perceptron with a focus on XOR. The contribution is a designed module and its mathematical validation, not a report

of classroom implementation data. By using XOR as a minimal non-linearity case and coordinating truth tables, points, inequalities, and composition, the module provides a practical pathway from secondary mathematics to foundational ideas in machine learning. Optional computational visualization complements the classroom narrative by making decision boundaries tangible.

A. Instructor notes and technical background

These notes provide background for instructors who wish to connect the classroom activities to standard formal results.

A.1. Linear separability and perceptron solvability

A dataset is linearly separable if and only if there exist weights (w, b) such that $w \cdot x + b \geq 0$ for every positive point x and $w \cdot x + b < 0$ for every negative point x . With the convention $H(z) = 1$ for $z \geq 0$, the positive class lies on the nonnegative side of the separating hyperplane and the negative class lies on the negative side.

A.2. Perceptron convergence theorem

If a dataset is linearly separable, the perceptron learning algorithm converges in finitely many updates to a separating hyperplane [7, 8].

A.3. Backpropagation in an L -layer network

For a feedforward network with differentiable activations, the gradient of the loss with respect to each weight can be computed by propagating error signals backward through the layers (backpropagation) [10].

B. Computation-graph worksheet for explaining backpropagation

This supplementary material provides a one-page worksheet that explains backpropagation as repeated multiplication of local derivatives along a computation graph, without requiring students to expand multivariable partial derivatives.

Worksheet (handout)

Consider a minimal sigmoid network with one input and one hidden unit:

$$z_1 = w_1x + b_1, \quad a_1 = \sigma(z_1), \quad z_2 = w_2a_1 + b_2, \quad a_2 = \sigma(z_2),$$

and the squared-error loss $L = \frac{1}{2}(a_2 - y)^2$. Use the identity $\sigma'(z) = \sigma(z)(1 - \sigma(z))$.

Snapshot values. Let $x = 1$, $y = 0$, and suppose a forward pass produced $z_1 = 1.0$, $a_1 = \sigma(z_1) = 0.731$, $\sigma'(z_1) = 0.197$; and $z_2 = 0.731$, $a_2 = \sigma(z_2) = 0.675$, $\sigma'(z_2) = 0.219$.

Tasks. Fill in the blanks.

(1) Output error signal:

$$\delta_2 := \frac{\partial L}{\partial z_2} = (a_2 - y)\sigma'(z_2) = \underline{\hspace{2cm}}$$

(2) Output-layer gradients:

$$\frac{\partial L}{\partial w_2} = \delta_2 a_1 = \underline{\hspace{2cm}}, \quad \frac{\partial L}{\partial b_2} = \delta_2 = \underline{\hspace{2cm}}$$

(3) Hidden error signal:

$$\delta_1 := \frac{\partial L}{\partial z_1} = \delta_2 w_2 \sigma'(z_1).$$

(Use $w_2 = 1$ for this worksheet.) Compute $\delta_1 = \underline{\hspace{2cm}}$.

(4) Input-layer gradients:

$$\frac{\partial L}{\partial w_1} = \delta_1 x = \underline{\hspace{2cm}}, \quad \frac{\partial L}{\partial b_1} = \delta_1 = \underline{\hspace{2cm}}$$

(5) Interpretation (one sentence): explain why each gradient is a product of “(error at the end)×(local slopes along the path)”.

Instructor answer key

Using the snapshot values in the worksheet,

$$\delta_2 = (0.675 - 0) \cdot 0.219 \approx 0.148,$$

$$\frac{\partial L}{\partial w_2} \approx 0.148 \cdot 0.731 \approx 0.108, \quad \frac{\partial L}{\partial b_2} \approx 0.148,$$

$$\delta_1 \approx 0.148 \cdot 1 \cdot 0.197 \approx 0.029,$$

$$\frac{\partial L}{\partial w_1} \approx 0.029, \quad \frac{\partial L}{\partial b_1} \approx 0.029.$$

The small discrepancies that can occur from rounding are acceptable for the worksheet.

C. Reference script for the computational visualization

The figures in Section 3.4 were produced by training a 2–2–1 sigmoid MLP on XOR with simple gradient descent and then evaluating the model on a grid to plot the contour $\hat{y} = 0.5$. For classroom use, it can be helpful to fix a random seed so that groups obtain comparable plots. With the settings below, the final predictions at the four XOR inputs are close to (0,1,1,0), and the script writes `fig2_decision_boundary.pdf` and `fig3_loss_curve.pdf` in the current directory.

```
# xor_mlp_reference.py
import numpy as np
import matplotlib.pyplot as plt

def sigmoid(z):
    return 1.0 / (1.0 + np.exp(-z))

def sigmoid_prime(z):
    s = sigmoid(z)
    return s * (1.0 - s)

# XOR data
X = np.array([[0, 0],
              [0, 1],
              [1, 0],
              [1, 1]], dtype=float)
y = np.array([[0],
              [1],
              [1],
              [0]], dtype=float)

# 2-2-1 network parameters (fixed seed for reproducibility)
rng = np.random.default_rng(0)
```

```
W1 = rng.normal(scale=1.0, size=(2, 2))
b1 = np.zeros((1, 2))
W2 = rng.normal(scale=1.0, size=(2, 1))
b2 = np.zeros((1, 1))

lr = 0.5
epochs = 10000
loss = []

for _ in range(epochs):
    # forward
    z1 = X @ W1 + b1
    a1 = sigmoid(z1)
    z2 = a1 @ W2 + b2
    a2 = sigmoid(z2)

    # mean squared error (MSE)
    L = np.mean((a2 - y) ** 2)
    loss.append(L)

    # backward
    dL_da2 = 2.0 * (a2 - y) / len(X)
    dL_dz2 = dL_da2 * sigmoid_prime(z2)
    dW2 = a1.T @ dL_dz2
    db2 = np.sum(dL_dz2, axis=0, keepdims=True)

    dL_da1 = dL_dz2 @ W2.T
    dL_dz1 = dL_da1 * sigmoid_prime(z1)
    dW1 = X.T @ dL_dz1
    db1 = np.sum(dL_dz1, axis=0, keepdims=True)

    # gradient descent update
    W2 -= lr * dW2
    b2 -= lr * db2
    W1 -= lr * dW1
```

```

    b1 -= lr * db1

# decision boundary on a grid
grid = np.linspace(-0.5, 1.5, 201)
xx, yy = np.meshgrid(grid, grid)
G = np.c_[xx.ravel(), yy.ravel()]

z1 = G @ W1 + b1
a1 = sigmoid(z1)
z2 = a1 @ W2 + b2
p = sigmoid(z2).reshape(xx.shape)

plt.figure()
plt.contourf(xx, yy, p, levels=np.linspace(0, 1, 21), alpha=0.85)
plt.colorbar(label="y")
plt.contour(xx, yy, p, levels=[0.5], linewidths=2)
plt.scatter(X[y[:, 0] == 0, 0], X[y[:, 0] == 0, 1], marker="o",
            label="class 0")
plt.scatter(X[y[:, 0] == 1, 0], X[y[:, 0] == 1, 1], marker="s",
            label="class 1")
plt.xlabel("x1")
plt.ylabel("x2")
plt.title("Decision boundary of a trained 2-2-1 MLP (sigmoid)")
plt.legend()
plt.tight_layout()
plt.savefig("fig2_decision_boundary.pdf")

plt.figure()
plt.plot(loss)
plt.yscale("log")
plt.xlabel("Epoch")
plt.ylabel("MSE loss")
plt.title("Training loss on XOR (2-2-1 MLP)")
plt.tight_layout()
plt.savefig("fig3_loss_curve.pdf")

```

References

1. D. Touretzky, C. Gardner-McCune, F. Martin, and D. Seehorn, Envisioning AI for K–12: What should every child know about AI?, *Proceedings of the AAAI Conference on Artificial Intelligence*, **33**(01) (2019), 9795–9799. doi:10.1609/aaai.v33i01.33019795.
2. AI4K12 Initiative, *AI for K–12 Guidelines and the Five Big Ideas in AI*, available online at <https://ai4k12.org/> (accessed 2026-05-12).
3. Ministry of Education (Korea), *2022 Revised Mathematics Curriculum*, Ministry of Education Notice No. 2022–33 [Book 8], 2022.
4. State Council of the People’s Republic of China, *A New Generation Artificial Intelligence Development Plan*, 2017. (English translation available via Stanford DigiChina: <https://digichina.stanford.edu/work/full-translation-chinas-new-generation-artificial-intelligence-development-plan-2017/>, accessed 2026-05-12.)
5. UK Government, *National AI Strategy*, 2021, available online at <https://www.gov.uk/government/publications/national-ai-strategy> (accessed 2026-05-12).
6. Singapore Ministry of Education, *Artificial intelligence in education (EdTech Masterplan 2030)*, last updated 11 May 2026, available online at <https://www.moe.gov.sg/education-in-sg/educational-technology-journey/edtech-masterplan/artificial-intelligence-in-education> (accessed 2026-05-12).
7. F. Rosenblatt, The perceptron: A probabilistic model for information storage and organization in the brain, *Psychological Review*, **65**(6) (1958), 386–408. doi:10.1037/h0042519.
8. A. B. J. Novikoff, On convergence proofs on perceptrons, in *Proceedings of the Symposium on the Mathematical Theory of Automata*, Vol. XII, Polytechnic Institute of Brooklyn, 1962, 615–622.
9. M. L. Minsky and S. A. Papert, *Perceptrons: An Introduction to Computational Geometry*, MIT Press, 1969.
10. D. E. Rumelhart, G. E. Hinton, and R. J. Williams, Learning representations by back-propagating errors, *Nature*, **323**(6088) (1986), 533–536. doi:10.1038/323533a0.
11. Y. Chevallard, *La transposition didactique: Du savoir savant au savoir enseigné*, La Pensée Sauvage, 1985.
12. R. Duval, A cognitive analysis of problems of comprehension in a learning of mathematics, *Educational Studies in Mathematics*, **61**(1–2) (2006), 103–131. doi:10.1007/s10649-006-0400-z.
13. G. J. Posner, K. A. Strike, P. W. Hewson, and W. A. Gertzog, *Accommodation of a scientific conception: Toward a theory of conceptual change*, *Science Education* **66**(2) (1982), 211–227. doi:10.1002/sce.3730660207.

14. N. Lytle, V. Cateté, D. Boulden, Y. Dong, J. Houchins, A. Milliken, A. Isvik, D. Bounajim, E. Wiebe, and T. Barnes, *Use, Modify, Create: Comparing computational thinking lesson progressions for STEM classes*, in *Proceedings of the 2019 ACM Conference on Innovation and Technology in Computer Science Education (ITiCSE '19)*, Association for Computing Machinery, New York, NY, 2019, 395–401. doi:10.1145/3304221.3319786.
15. D. Long and B. Magerko, What is AI literacy? Competencies and design considerations, in *Proceedings of the 2020 CHI Conference on Human Factors in Computing Systems (CHI '20)*, Association for Computing Machinery, New York, NY, 2020, 1–16. doi:10.1145/3313831.3376727.
16. S. Prediger, K. Gravemeijer, and J. Confrey, Design research with a focus on learning processes: An overview on achievements and challenges, *ZDM Mathematics Education*, **47**(6) (2015), 877–891. doi:10.1007/s11858-015-0722-3.
17. T. Plomp and N. Nieveen (eds.), *Educational Design Research, Part A: An Introduction*, SLO (Netherlands Institute for Curriculum Development), Enschede, 2013.
18. J. van den Akker, K. Gravemeijer, S. McKenney, and N. Nieveen (eds.), *Educational Design Research*, Routledge, 2006. doi:10.4324/9780203088364.
19. S. Kindler, S. Schönbrodt, and M. Frank, Analysing artificial neural networks in high school mathematics education, in S. Podworny and S. Schönbrodt (eds.), *Towards Fostering AI and Data Science Literacy in Schools Across Disciplines: Proceedings of the 1st Symposium on Integrating AI and Data Science into School Education Across Disciplines (AIDEA1 2025)*, Salzburg, Austria, 2025.
20. S. Lee and H. K. Ko, Development of optimization teaching and learning materials for artificial intelligence mathematics using ChatGPT and Python, *Communications of Mathematical Education*, **38**(3) (2024), 459–486. doi:10.7468/jksmee.2024.38.3.459.
21. K.-H. Lee, D. Shin, J.-H. Park, and S. Oh, Educational effects of an AI-based system for perceptron learning in artificial intelligence mathematics: Analyzing changes in high school students' affective domain, *The Journal of Research in Education*, **38**(4) (2025), 87–116. doi:10.24299/kier.2025.384.87.

^aSEJUN OH, ASSISTANT PROFESSOR, HONGIK UNIVERSITY, SEOUL, KOREA:
 Email address: soh@hongik.ac.kr