Korea Aerospace University

2AI4203: Machine Learning

Lec04: Model Evaluation and Generalization

검증 일반화

Jiyoon Shin

KAU College of AI Convergence

Sep. 23, 2026

Recap:

Classification with Linear Models

- Binary
- Multiclass

Binary Classification

- For binary classification,

$$y \in \{0, 1\}$$

$wx+b$

- A linear model produces a score:

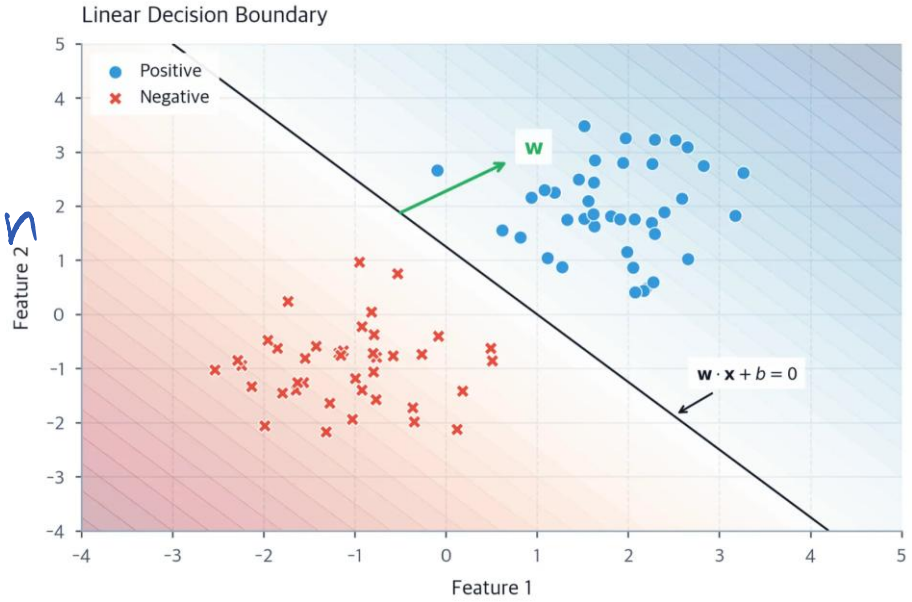
$$z = \mathbf{w}^T\mathbf{x} + b$$

lin— regression

  - $z > 0 \longrightarrow$ Class 1
  - $z < 0 \longrightarrow$ Class 0
  - $z = 0 \longrightarrow$ Decision boundary

① $z = -1$

② $z = -5$

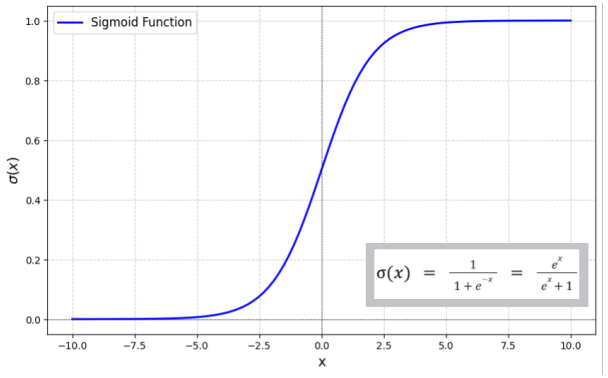


Logistic Regression

- The linear score is converted into a probability using the sigmoid function:

$$p = \sigma(z) = \frac{1}{1 + e^{-z}}$$

$$P(y = 1|\mathbf{x}) = \sigma(\mathbf{w}^T\mathbf{x} + b)$$



  - p: probability of Class 1
  - $1 - p$: probability of Class 0

- Prediction

$$\hat{y} = \begin{cases} 1, & p \geq 0.5 \\ 0, & p < 0.5 \end{cases}$$

- Pipeline

$$\mathbf{x} \rightarrow z \rightarrow \sigma(z) \rightarrow p \rightarrow \hat{y}$$

$z = \mathbf{w}^T\mathbf{x} + b$ $\quad \sigma(z) = \frac{1}{1+e^{-z}}$ $\quad P(y=1|x)$ $\quad \geq 0.5$ $\quad < 0.5$

Learning Logistic Regression

- To learn $\mathbf{w}$ and b, we minimize Binary Cross-Entropy (BCE):

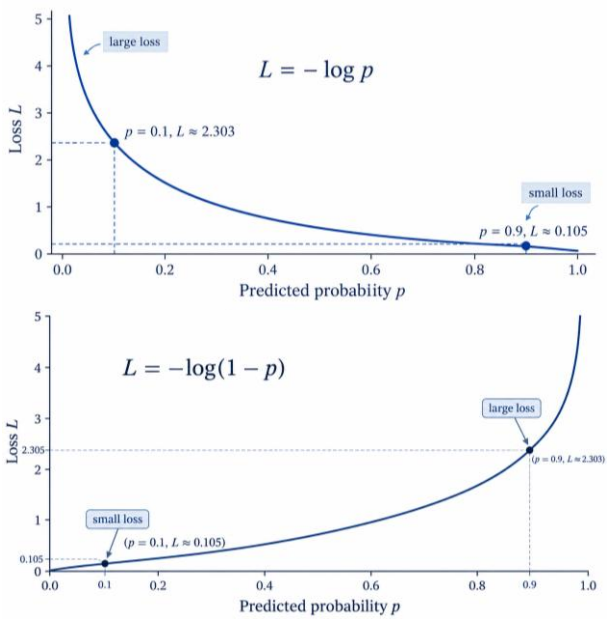
$$L = -[y \log p + (1 - y) \log(1 - p)]$$

  - When $y = 1$:

$$L = -\log p$$

  - When $y = 0$:

$$L = -\log(1 - p)$$

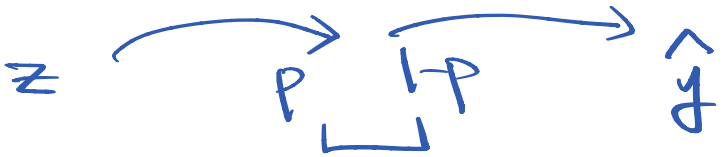


- Then update the parameters using gradient descent

$$\mathbf{w} \leftarrow \mathbf{w} - \eta \nabla_{\mathbf{w}} L$$

$$b \leftarrow b - \eta \frac{\partial L}{\partial b}$$

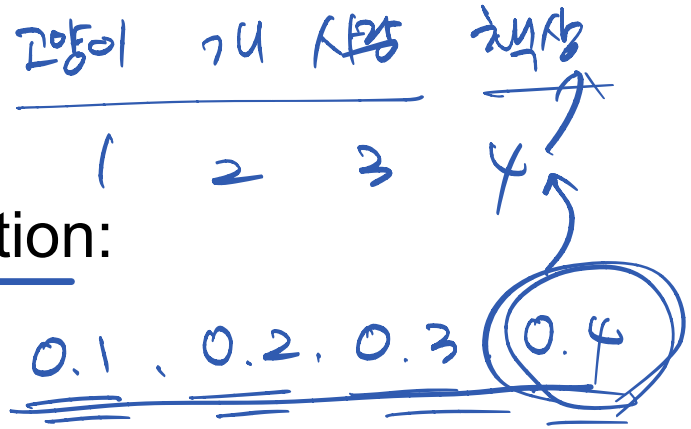
Multiclass Classification



- For K classes, compute one score for each class:

$$z_k = \mathbf{w}_k^T \mathbf{x} + b_k$$

$$\mathbf{z} = [z_1, z_2, \ldots, z_k]$$

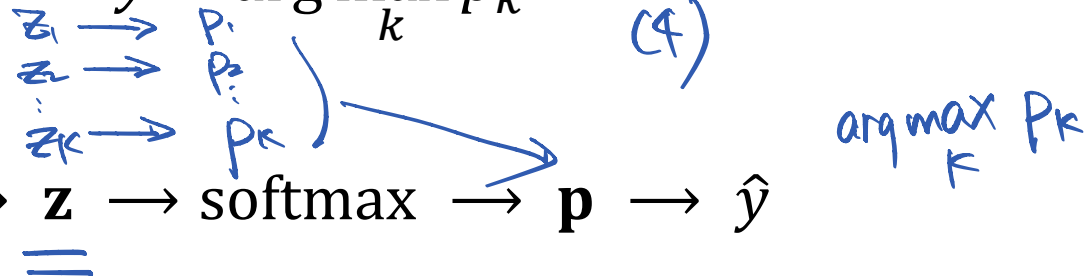


- Convert the scores into probabilities using the softmax function:

$$p_k = \frac{e^{z_k}}{\sum_{j=1}^{K} e^{z_j}}$$

- Prediction

$$\hat{y} = \arg\max_k p_k$$



- Pipeline

$$\mathbf{x} \rightarrow \mathbf{z} \rightarrow \text{softmax} \rightarrow \mathbf{p} \rightarrow \hat{y}$$

Learning Multiclass Classifiers

- For multiclass classification, use cross-entropy loss. If the true class is c,

① $$L = -\log p_c$$

- Equivalently, using a one-hot label:

② $$\mathbf{y} = [y_1, \ldots, y_K]$$

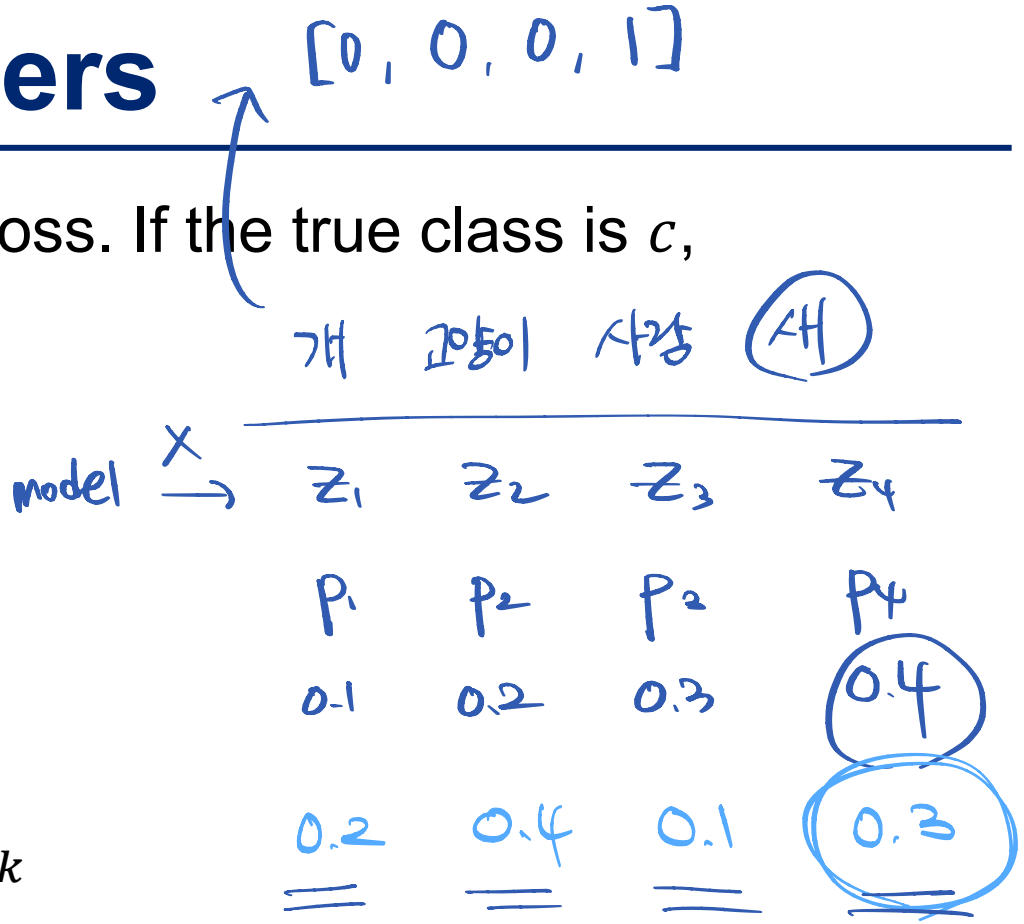
$$L = -\sum_{k=1}^{K} y_k \log p_k$$

$$= -\log p_4$$

- Then update the parameters using gradient descent

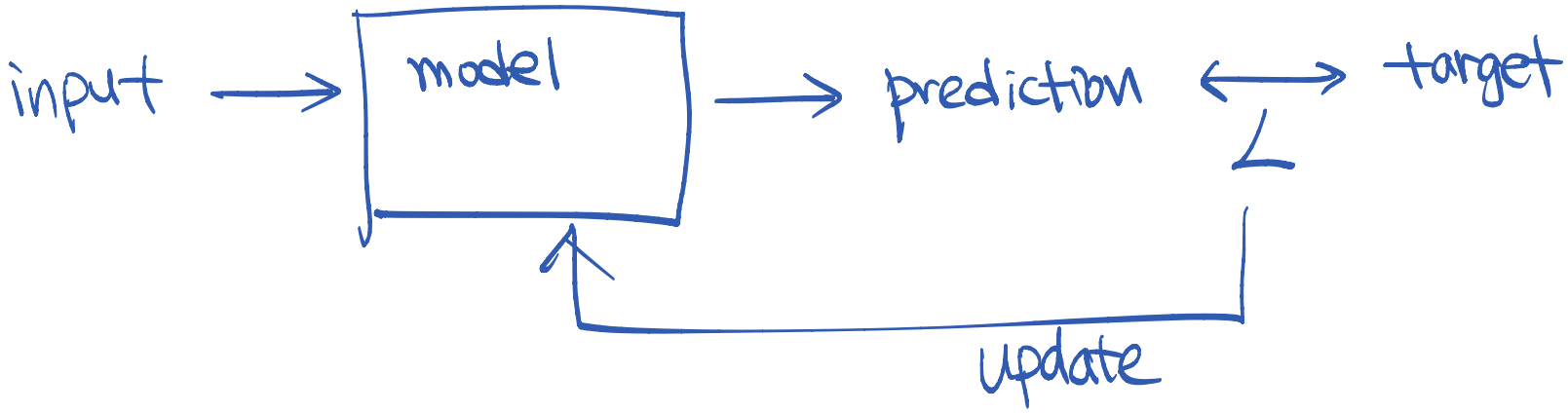
$$w \leftarrow w - \eta \nabla_w L$$

$$b \leftarrow b - \eta \frac{\partial L}{\partial b}$$



Korea Aerospace University

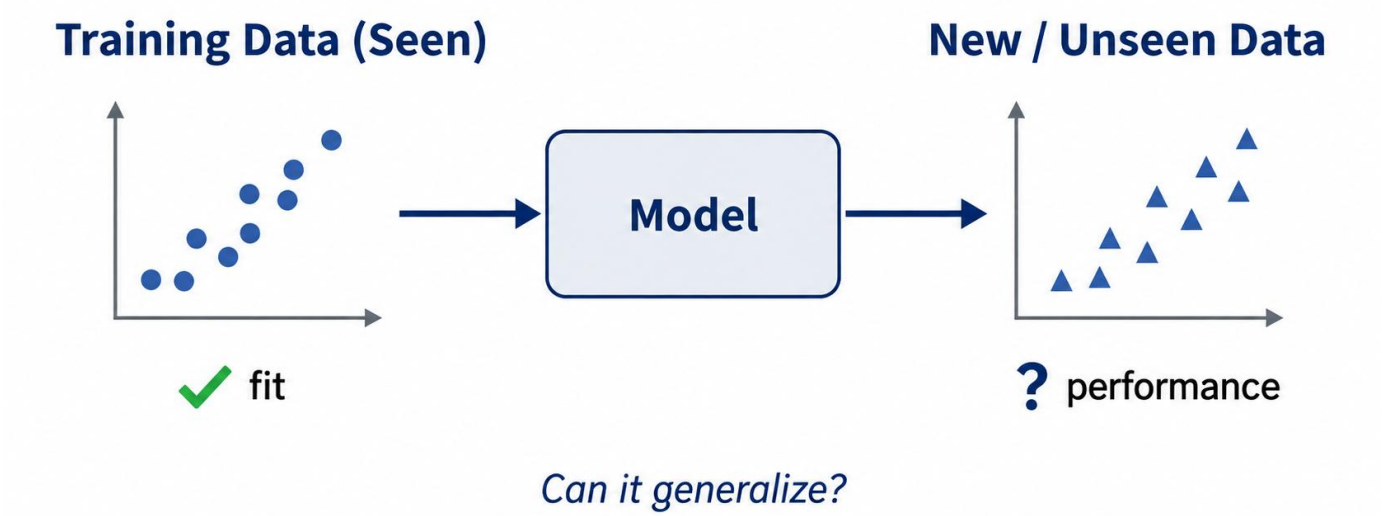
Part 1. Model Evaluation



Evaluation

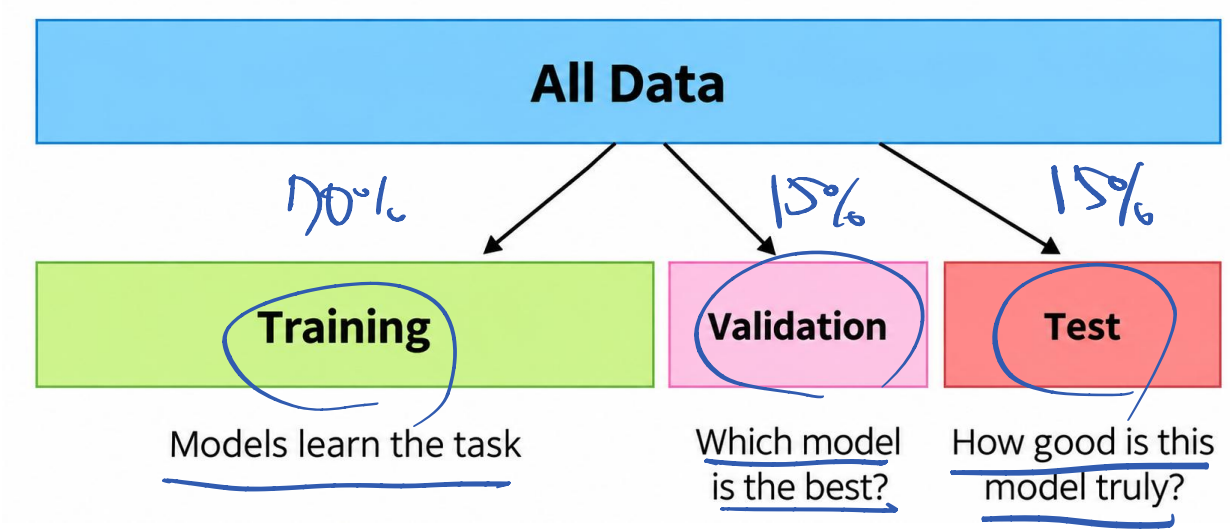
- A model should perform well on unseen data, not only on training data
- Evaluation tells us how well the model performs on new data
- The ultimate goal is generalization (일반화)

- Good training performance does not guarantee performance on unseen data



Data Splits

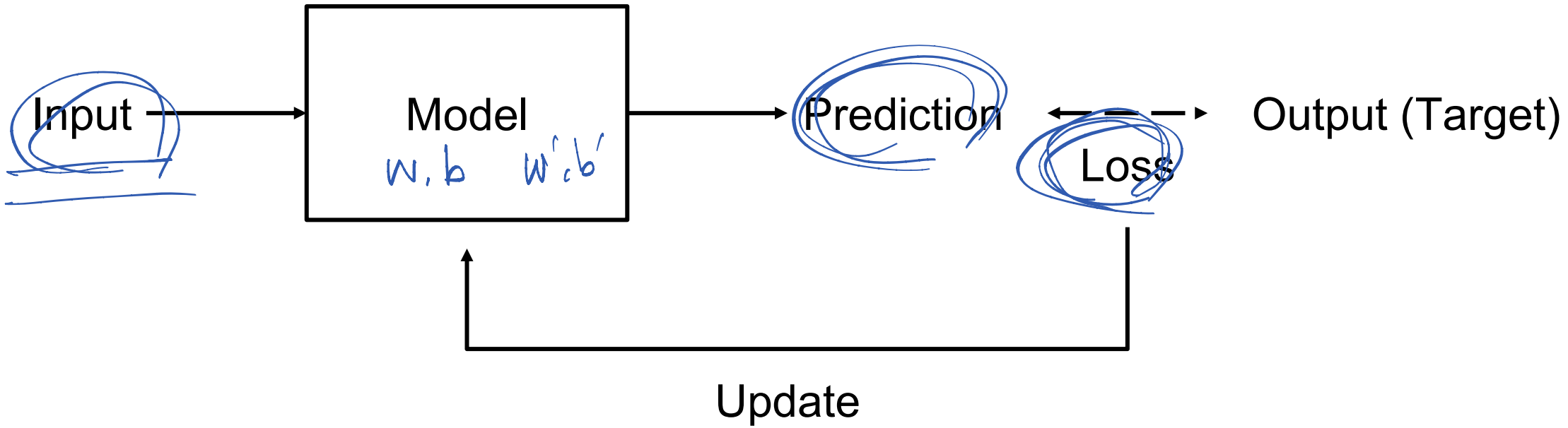
- Training set: used to learn the model
- Validation set: used to make choices about the model
- Test set: used for final evaluation



Training Set

- The training set is used to learn the model parameters

- The model sees the training data repeatedly during training



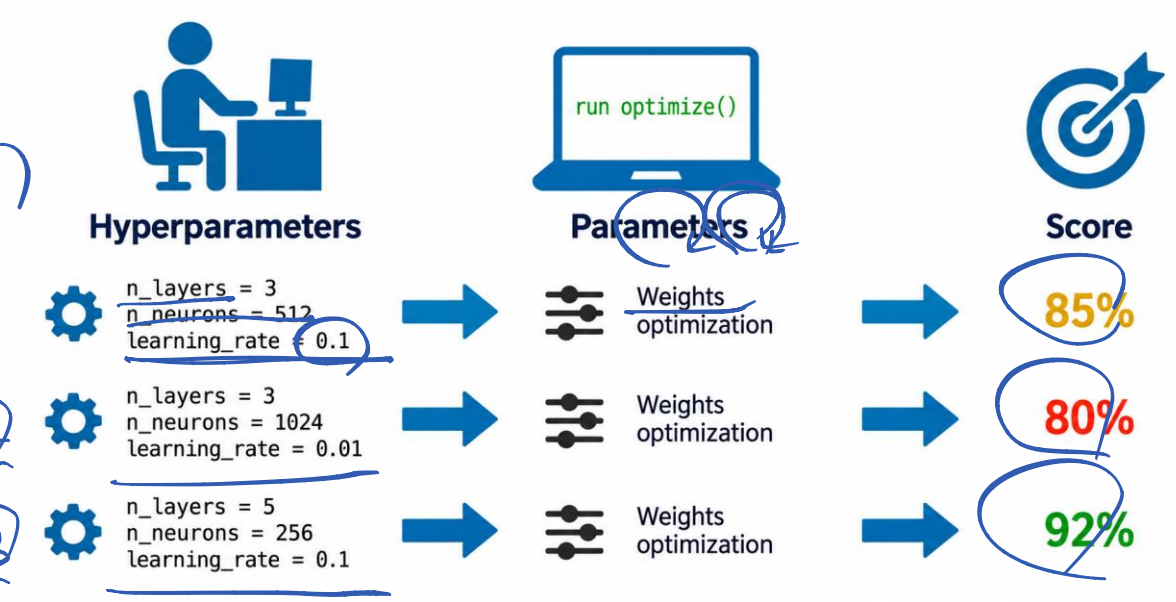
Validation Set

- The validation set is used to make decisions about the model
  - Which model should we use?
  - What learning rate should we use?
  - How complex should the model be?
  - How strong should regularization be?

- The validation set is not used to directly update model parameters

Parameters vs. Hyperparameters

- Parameters: learned from training data
  - w, b in linear/logistic regression
  - Weights in a neural network
- Hyperparameters: settings chosen before training and tuned using validation data
  - Learning rate
  - Regularization strength λ
  - Number of training epochs

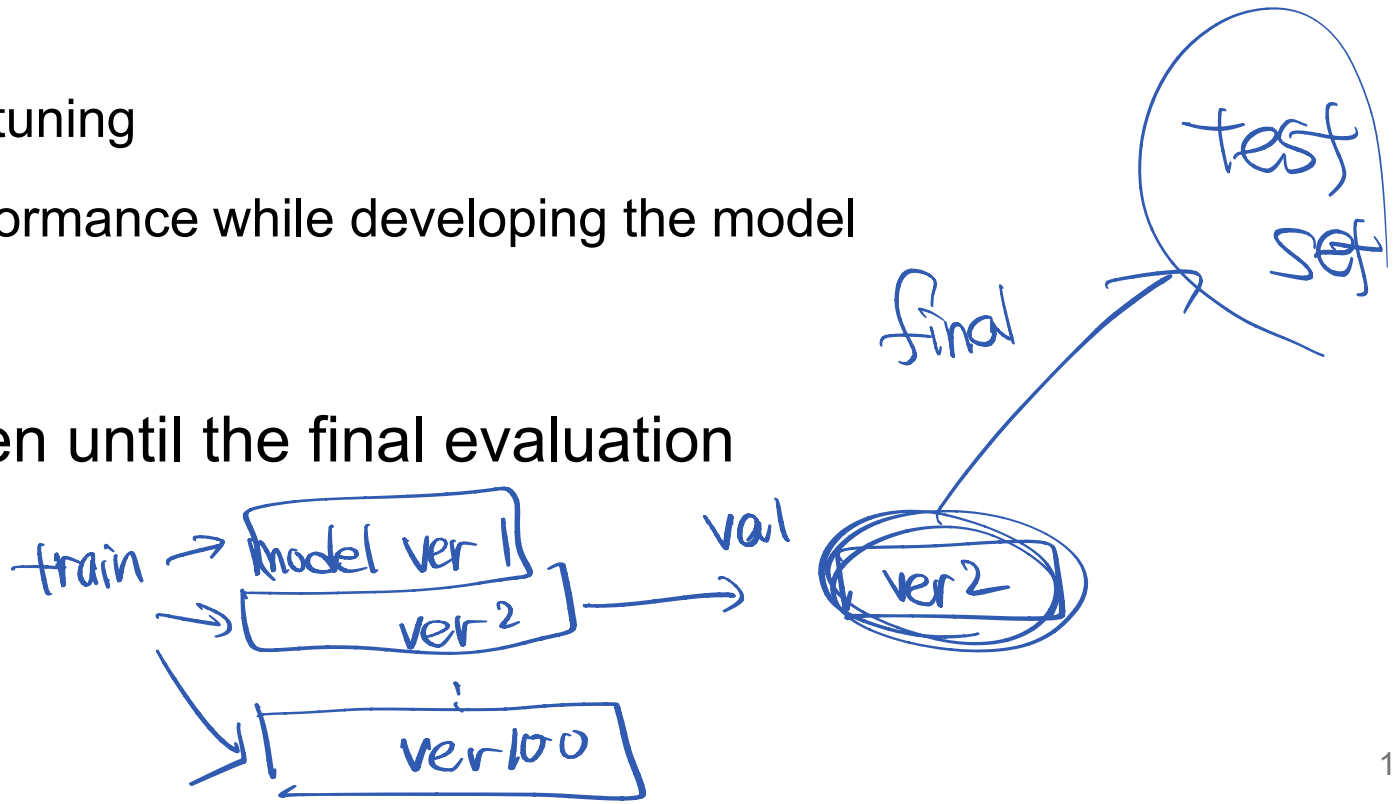


Test Set

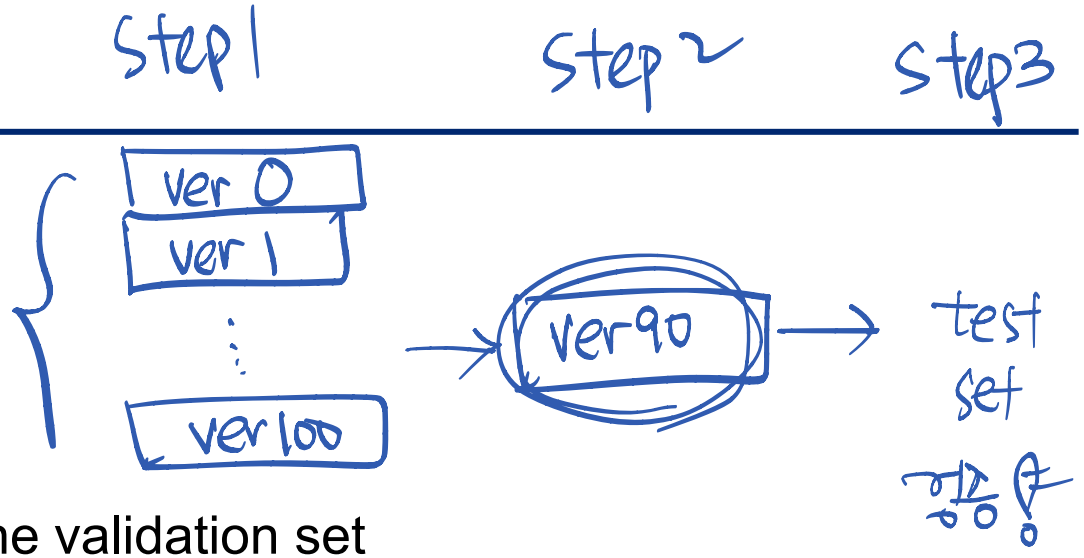
- The test set is used to evaluate the final model on unseen data

- Important rules
  - Use it after model selection
  - Do not use it for hyperparameter tuning
  - Do not repeatedly check test performance while developing the model

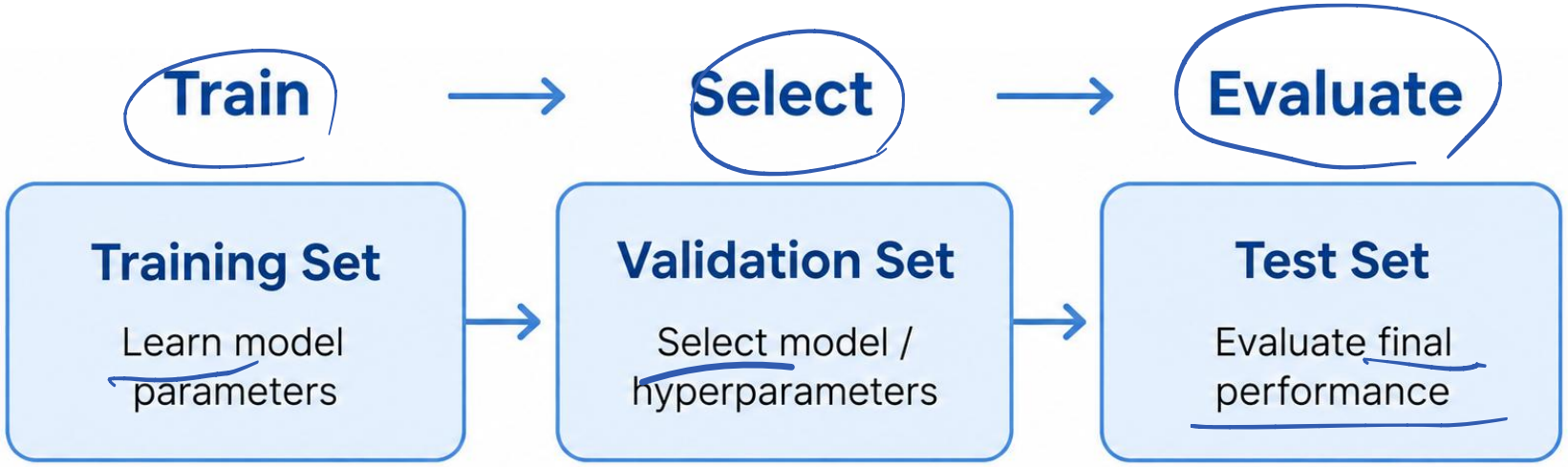
- The test set should remain unseen until the final evaluation



Evaluation Workflow



- Step 1: Train
  - Learn model parameters using the training set
- Step 2: Select
  - Choose the model and hyperparameters using the validation set
- Step 3: Evaluate
  - Measure final performance using the test set



Model Selection

- Which learning rate should we choose?

Learning Rate	Training Accuracy	Validation Accuracy
0.001	82%	80%
0.01	91%	88%
0.1	96%	76%

- Model selection is based on validation performance, not training performance

Cross-Validation

- A single train-validation split may not always be reliable
- Example: same dataset, different random splits
  - Split A: validation accuracy = 74%
  - Split B: validation accuracy = 83%

- Evaluation can depend on which samples happen to be in the validation set
- Can we evaluate the model using multiple splits?

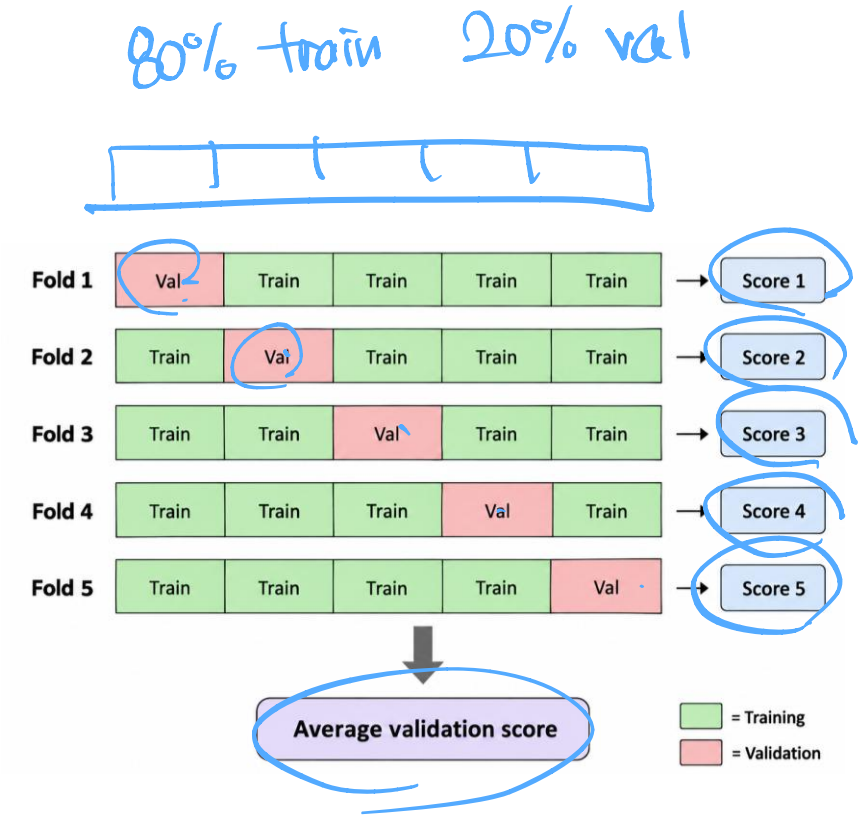
K-Fold Cross-Validation

- K-fold cross-validation evaluates a model using multiple train-validation splits

- Process
  - Divide the data into K folds
  - Use one fold for validation and the others for training
  - Repeat K times
  - Average the validation scores

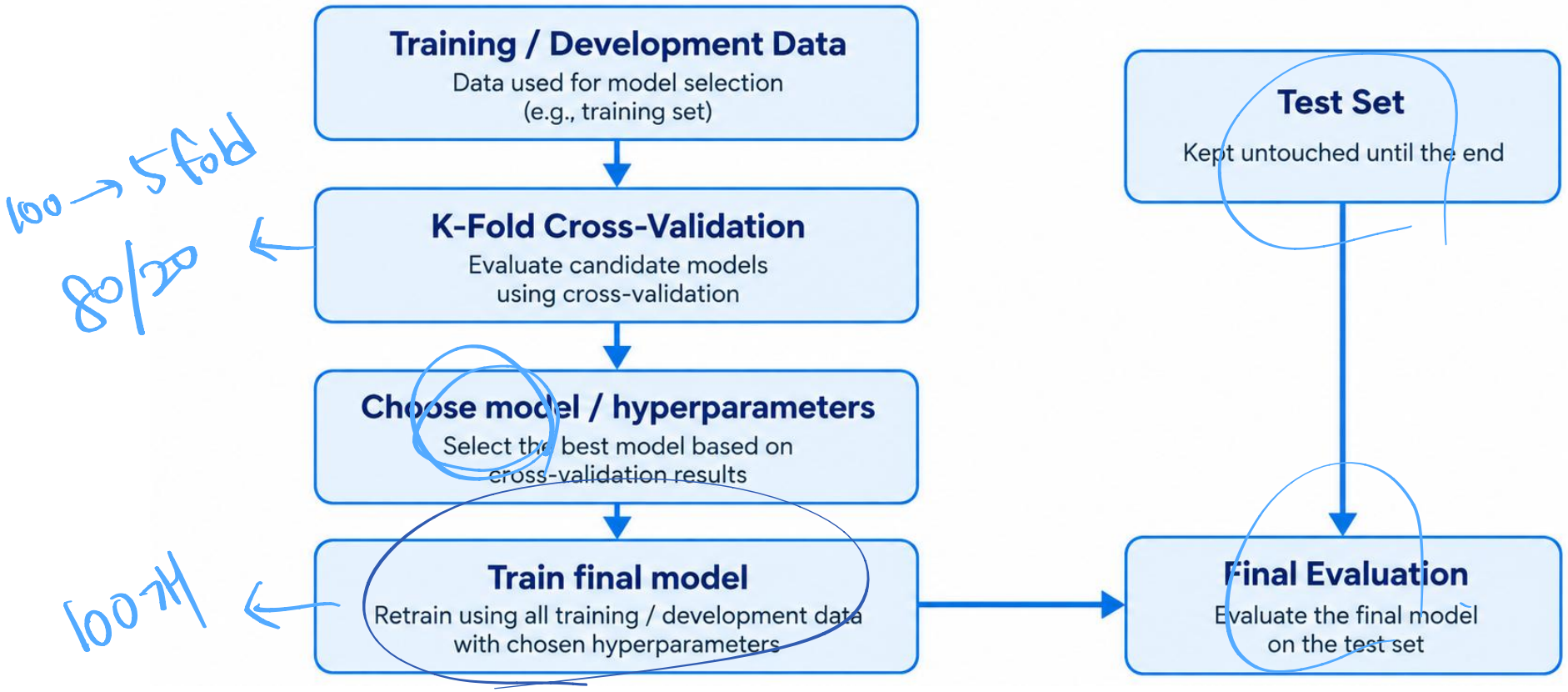
$$\text{CV Score} = \frac{1}{K}\sum_{k=1}^{K} \text{Score}_k$$

model 정확히.



Final Test

- Cross-validation is used for model selection
- The test set is still kept separate



Korea Aerospace University

Part 2. Performance Metrics

Performance Metrics

- A performance metric summarizes how well a model performs
- But there is no single metric that works best for every problem
  - Different tasks require different metrics
  - Different metrics emphasize different types of errors
  - The right metric depends on what matters in the application

- Example
  - Actual price: $500K / Predicted price: $520K → How large is the error?
  - Actual class: Cat / Predicted class: Dog → Is the prediction correct?

Regression Metrics

- For regression, both targets and predictions are continuous values

$$y_i = \text{target}$$
$$\hat{y}_i = \text{prediction}$$

- For each sample, the prediction error is

$$e_i = y_i - \hat{y}_i$$

$e_i = \hat{y}_i - y_i$?

- Common metrics:
  - Mean Absolute Error (MAE)
  - Mean Squared Error (MSE)
  - Root Mean Squared Error (RMSE)
- For all three metrics, lower is better

MAE

- Mean Absolute Error (MAE) measures the average absolute prediction error

$$\text{MAE} = \frac{1}{N}\sum_{i=1}^{N}|y_i - \hat{y}_i|$$

- Example
  - Actual

$$[3, 5, 8]$$

  - Prediction

$$[2, 7, 7]$$

  - Absolute errors

$$[1, 2, 1]$$

  - Therefore,

$$\text{MAE} = \frac{1+2+1}{3} = \frac{4}{3} \approx 1.33$$

MSE

- Mean Squared Error (MSE) averages the squared prediction errors

$$\text{MSE} = \frac{1}{N}\sum_{i=1}^{N}(y_i - \hat{y}_i)^2$$

- Example
  - Actual

    $[3, 5, 8]$
  - Prediction

    $[2, 7, 7]$
  - Errors

    $[1, -2, 1]$
  - Therefore,

    $$MSE = \frac{1 + 4 + 1}{3} = 2$$

RMSE

- Root Mean Squared Error (RMSE) is the square root of MSE

$$\text{RMSE} = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(y_i - \hat{y}_i)^2}$$

- Example
  - From the previous example,

$$\text{MSE} = 2$$

  - Then,

$$\text{RMSE} = \sqrt{2} \approx 1.41$$

Comparing Regression Metrics

$y, \hat{y}$ $

Metric	Main idea	Large errors	Unit
MAE	Absolute errors	Less sensitive 1.33	Same as target $
MSE	Squared errors	More sensitive 2	Squared unit $^2
RMSE	Square root of MSE	More sensitive 1.41	Same as target $

MAE보다는↑

Classification Metrics

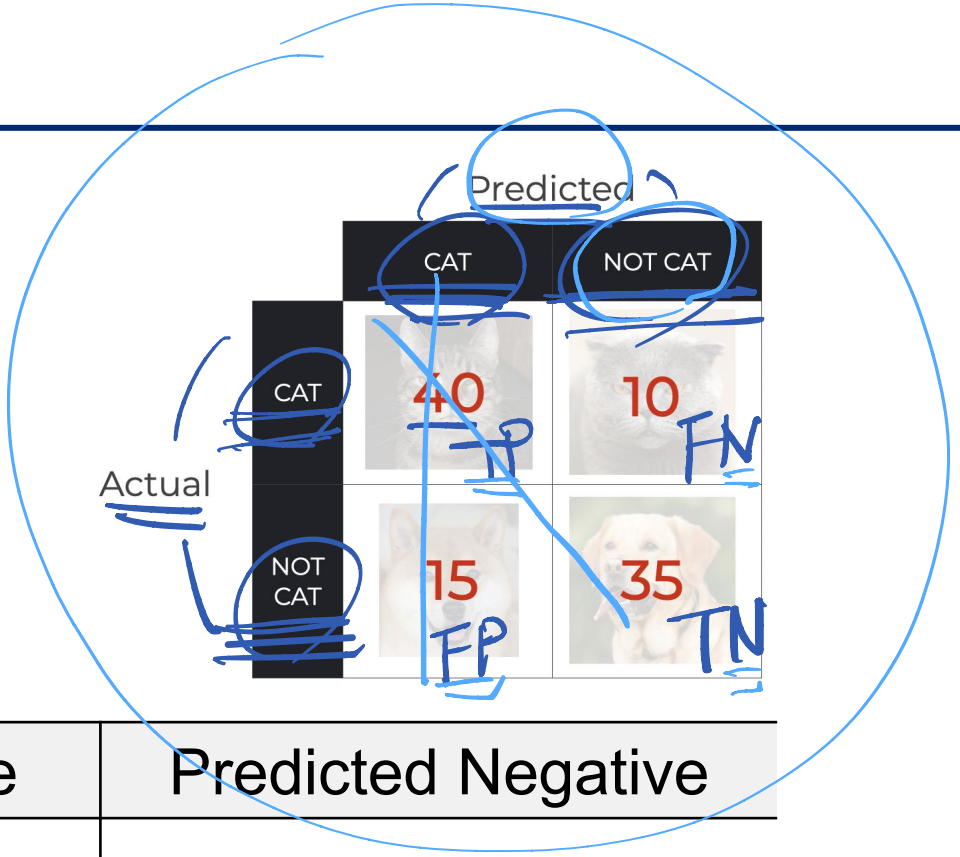
- For classification, we compare predicted class labels with true labels

- We first focus on binary classification: positive vs. negative
- Common metrics:
  - Accuracy
  - Precision
  - Recall
  - F1 Score

- These metrics can be understood using a confusion matrix

Confusion Matrix

- TP: Positive → correctly predicted positive
- TN: Negative → correctly predicted negative
- FP: Negative → incorrectly predicted positive
- FN: Positive → incorrectly predicted negative



	Predicted Positive	Predicted Negative
Actual Positive	TP – correctly detected	FN – missed
Actual Negative	FP – false alarm	TN – correctly rejected

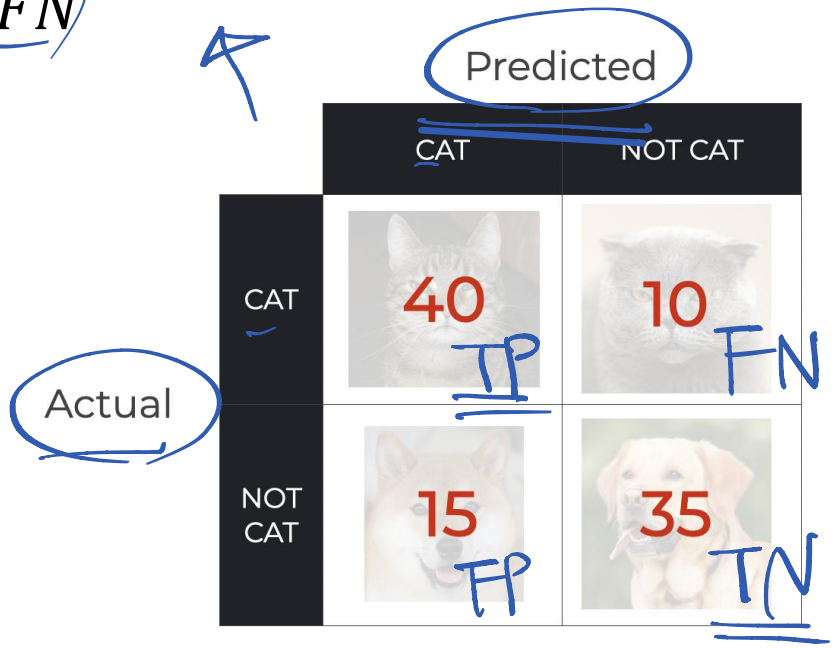
Accuracy

- Accuracy measures the fraction of correct predictions

$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN}$$

- Example
  - 100 predictions:
    - 90 correct
    - 10 incorrect

$$\text{Accuracy} = \frac{90}{100} = 90\%$$



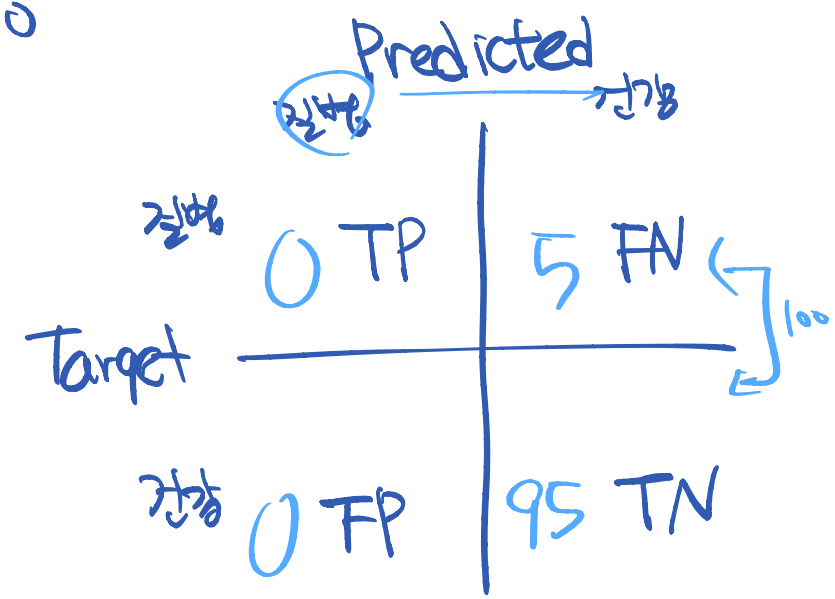
- Accuracy works well for balanced classes and equally important errors

When Accuracy Fails

- Accuracy may mislead
- Example
  - Suppose we have 100 patients:
    - 95 healthy
    - 5 have a disease
  - A model predicts everyone as healthy

$$\text{Accuracy} = \frac{95}{100} = 95\%$$

  - But
    - TP = 0
    - FN = 5
    - Recall = 0%



Precision

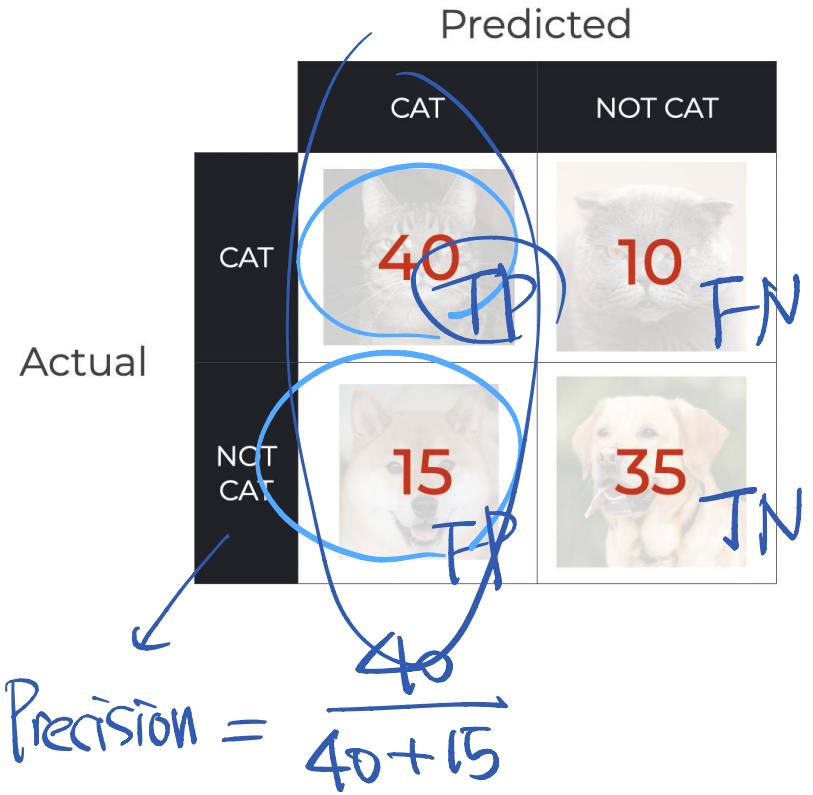
- Precision asks: “Of all predicted positives, how many are actually positive?”

$$\text{Precision} = \frac{TP}{TP + FP}$$

- High precision → few false positives

- Example
  - The model predicts 10 patients as having a disease:
    - 8 actually have the disease → TP
    - 2 are healthy → FP

$$\text{Precision} = \frac{8}{10} = 80\%$$



Recall

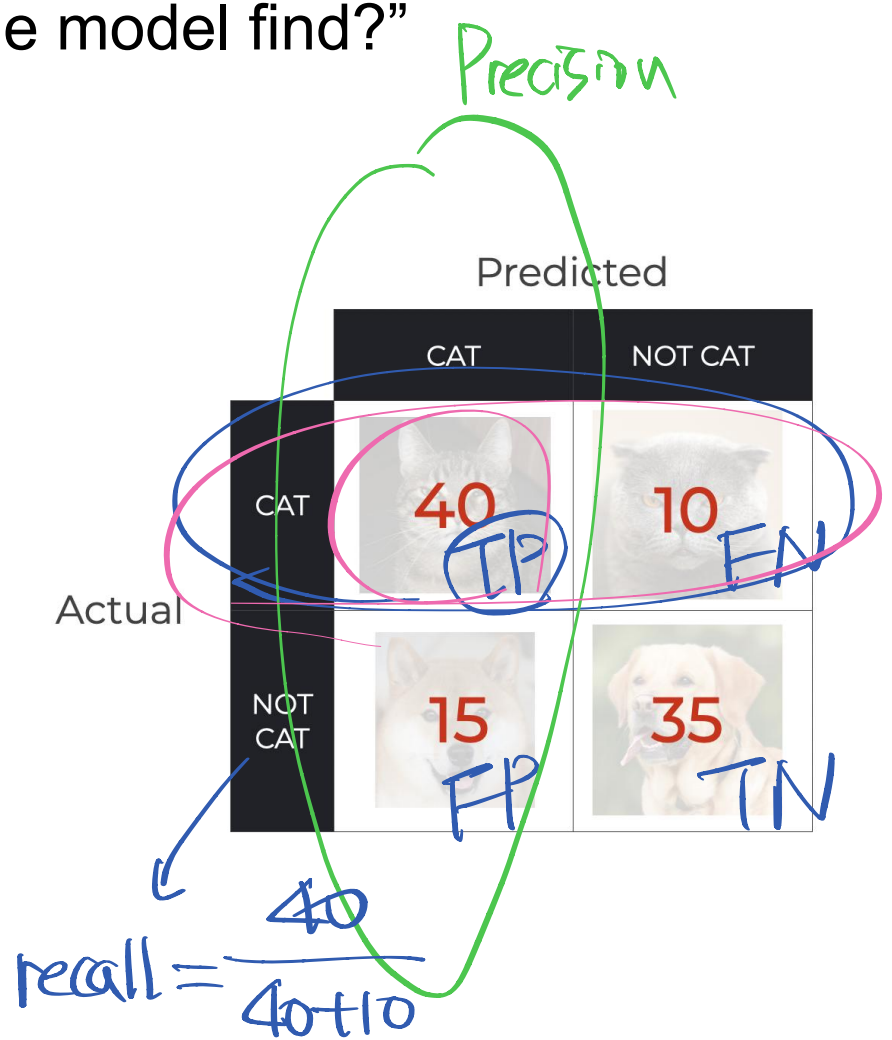
- Recall asks: “Of all actual positives, how many did the model find?”

$$\text{Recall} = \frac{TP}{TP + FN}$$

- High recall → few false negatives

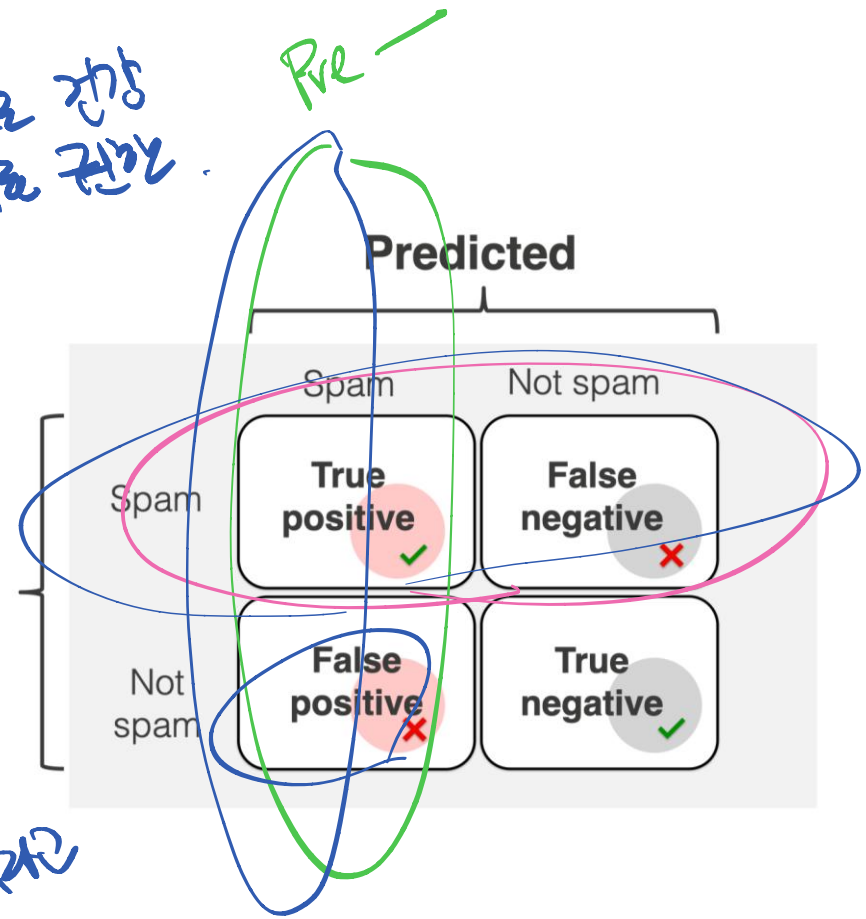
- Example
  - There are 10 patients with a disease:
    - 8 detected → TP
    - 2 missed → FN

$$\text{Recall} = \frac{8}{10} = 80\%$$



Precision vs. Recall

- Precision $= \frac{TP}{TP+FP}$
  - When false positives are costly → reduce FP
  - Examples
    - Spam detection
    - Recommending expensive treatments
    - Automatic account banning
- Recall $= \frac{TP}{TP+FN}$
  - When false negatives are costly → reduce FN
  - Examples
    - Disease screening
    - Fraud detection
    - Safety defect detection



F1 Score

- F1 Score is the harmonic mean of precision and recall

$$F_1 = 2 \cdot \frac{\text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}}$$

- It is high only when both precision and recall are high
- Useful when we want a single metric that balances precision and recall

- Example
  - Precision = 0.8
  - Recall = 0.6

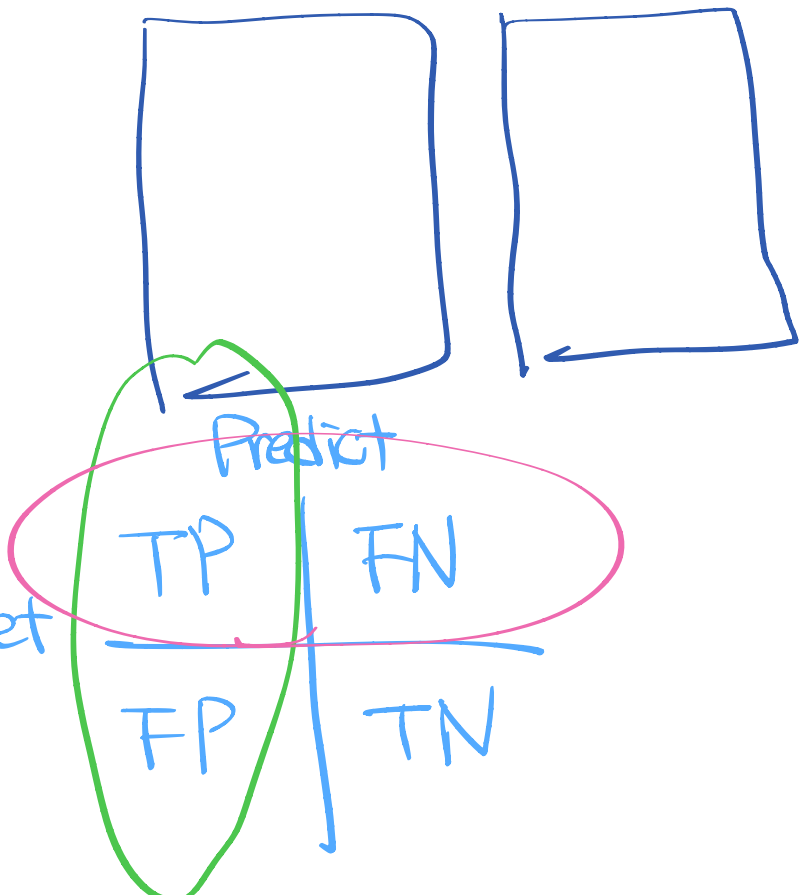
$$F_1 = 2 \cdot \frac{0.8 \times 0.6}{0.8 + 0.6} \approx 0.69$$

Choosing a Metric

- Regression
  - MAE → typical error, less affected by large errors
  - MSE / RMSE → penalize large errors more strongly

- Classification
  - Accuracy → overall correctness
  - Precision → when false positives are costly
  - Recall → when false negatives are costly
  - F1 → when both precision and recall matter

- There is no universally best metric



Korea Aerospace University

Part 3. Generalization

Generalization (일반화)

- Generalization is the ability of a model to perform well on unseen data

- A good model should
  - Learn useful patterns from the training data
  - Perform well on new examples
  - Avoid simply memorizing the training samples

- The goal is to learn patterns that extend beyond the training examples

Training vs. Validation Error

- We usually consider two types of error
  - Training error: error on the training data
  - Validation error: error on unseen validation data

- Good generalization

$$\text{Training Error} \approx \text{Validation Error} \approx \text{Low}$$

- Possible underfitting

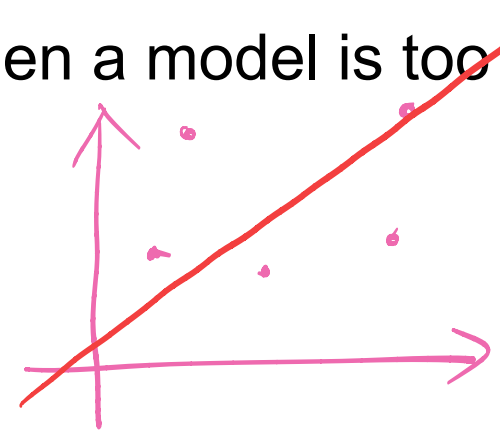
$$\text{Training Error} \approx \text{Validation Error} \approx \text{High}$$

- Possible overfitting

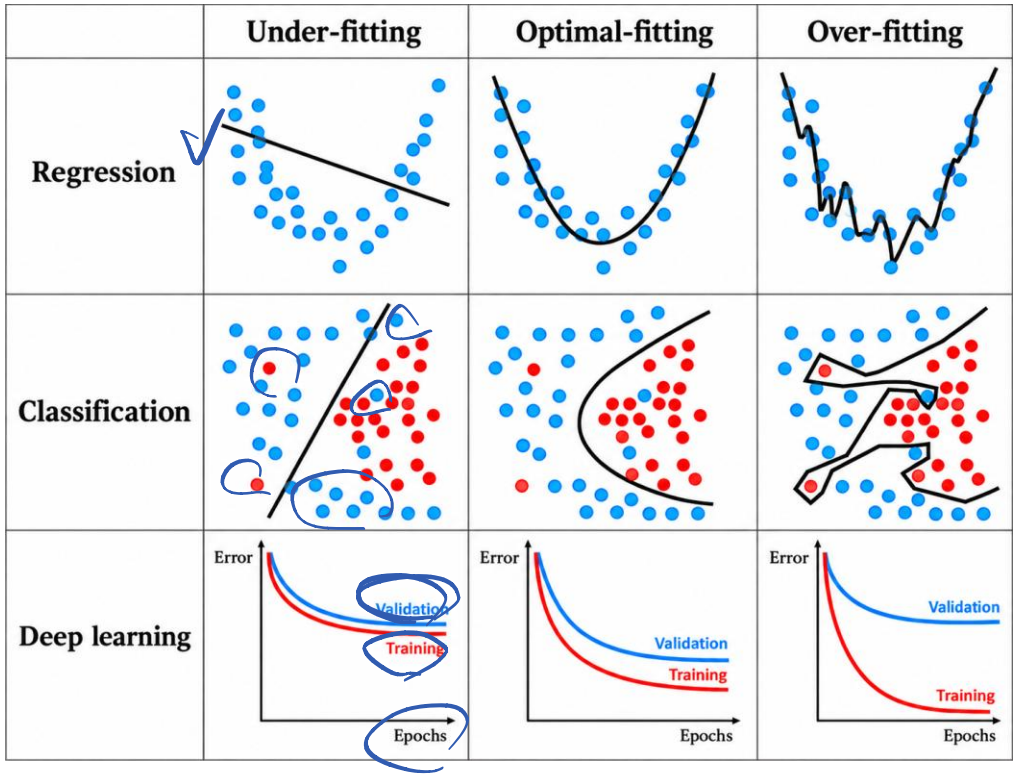
$$\text{Training Error} \ll \text{Validation Error}$$

Underfitting

- Underfitting occurs when a model is too simple to capture the underlying pattern



- Characteristics
  - High training error
  - High validation error
  - Poor performance even on the training data
- Often associated with high bias



Overfitting

- Overfitting occurs when a model fits the training data too closely

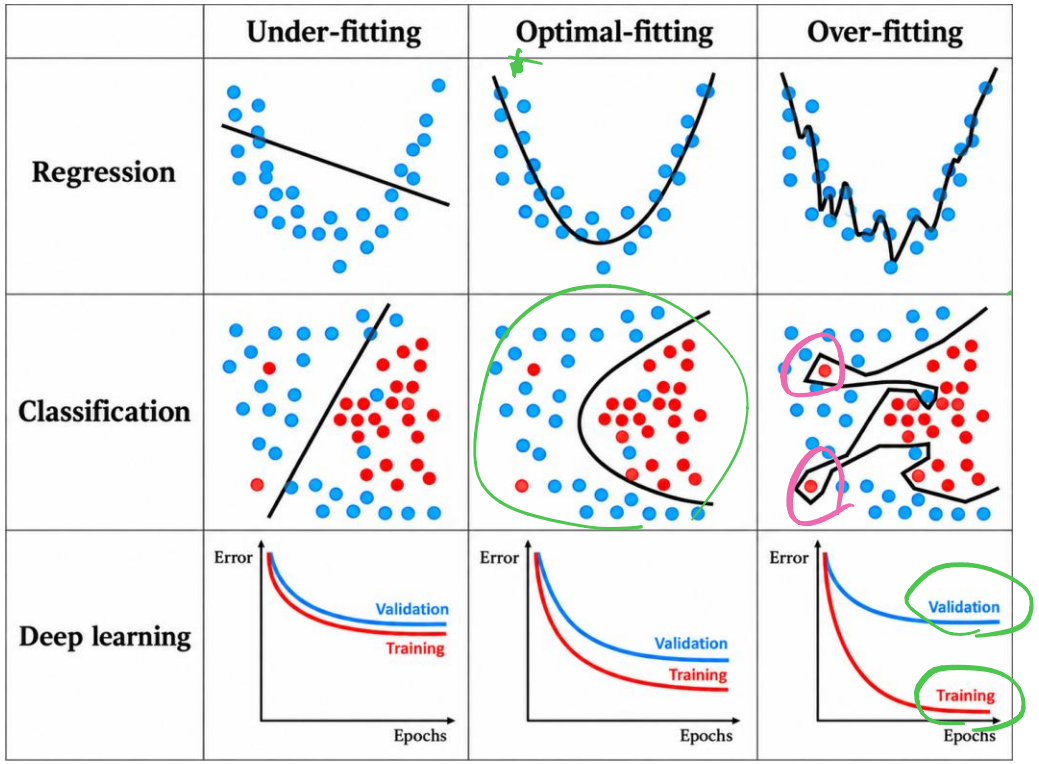
- Characteristics
  - Very low training error
  - Much higher validation error
  - Learns noise or training-specific details

overfitting

- Often associated with high variance

분산

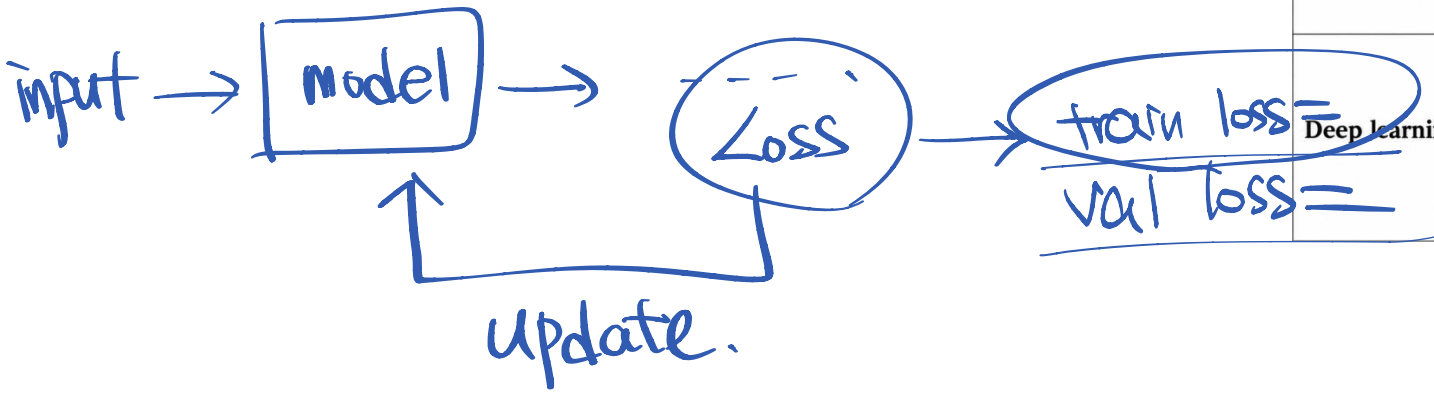
다양성

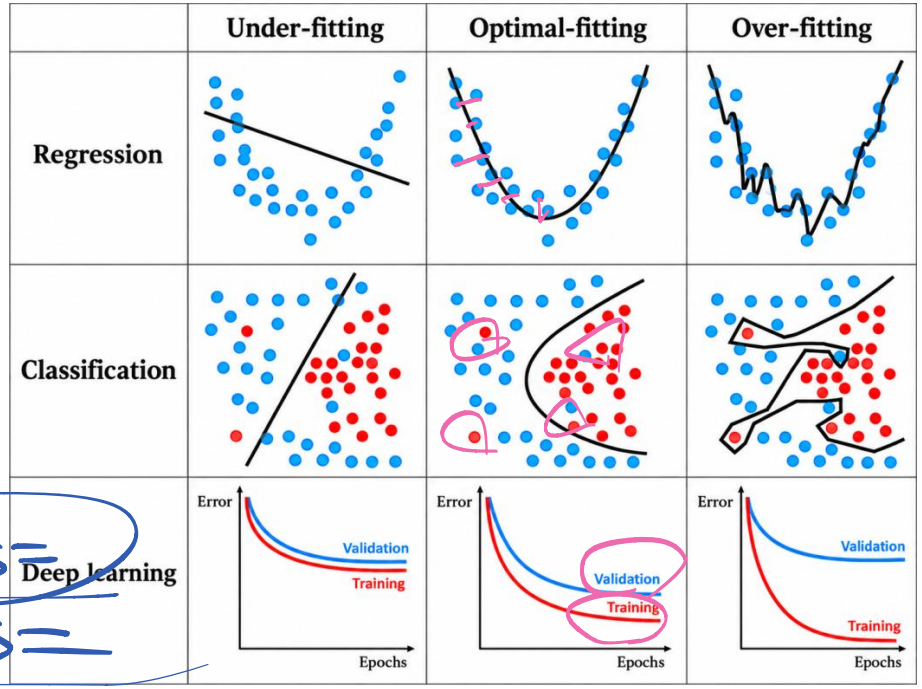


Good Fit

- A well-generalized model captures the underlying pattern, not every detail

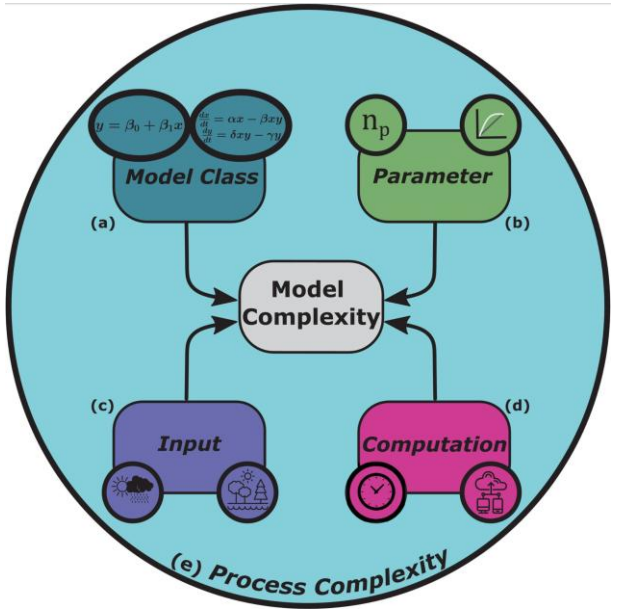
- Characteristics
  - Low training error
  - Low validation error
  - Small gap between training/validation performance





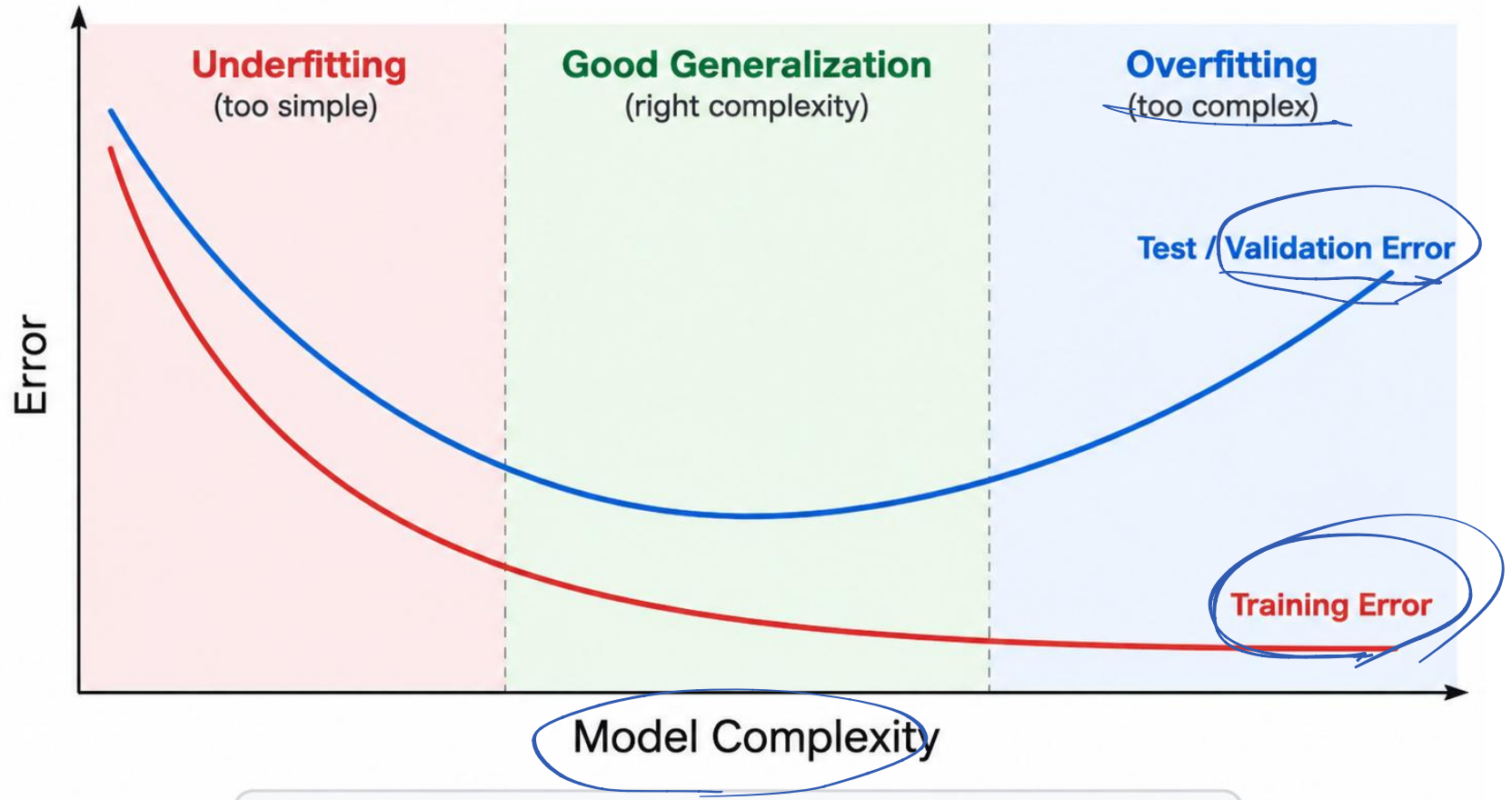
Model Complexity

- Model complexity describes how flexible a model is
- Examples
  - Polynomial degree
  - Tree depth
  - Number of parameters
  - Number of layers in a neural network



- As complexity increases, the model can fit more complicated patterns
  - Too little complexity → risk of underfitting
  - Too much complexity → risk of overfitting

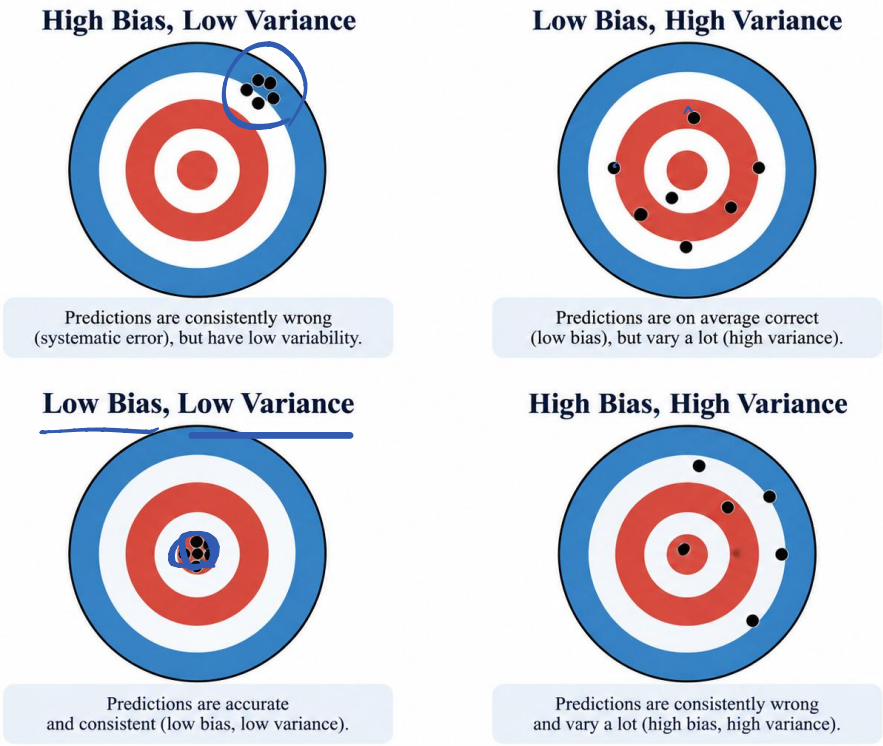
Model Complexity and Error



- Low complexity → **underfitting**
- Moderate complexity → **good generalization**
- High complexity → **overfitting**

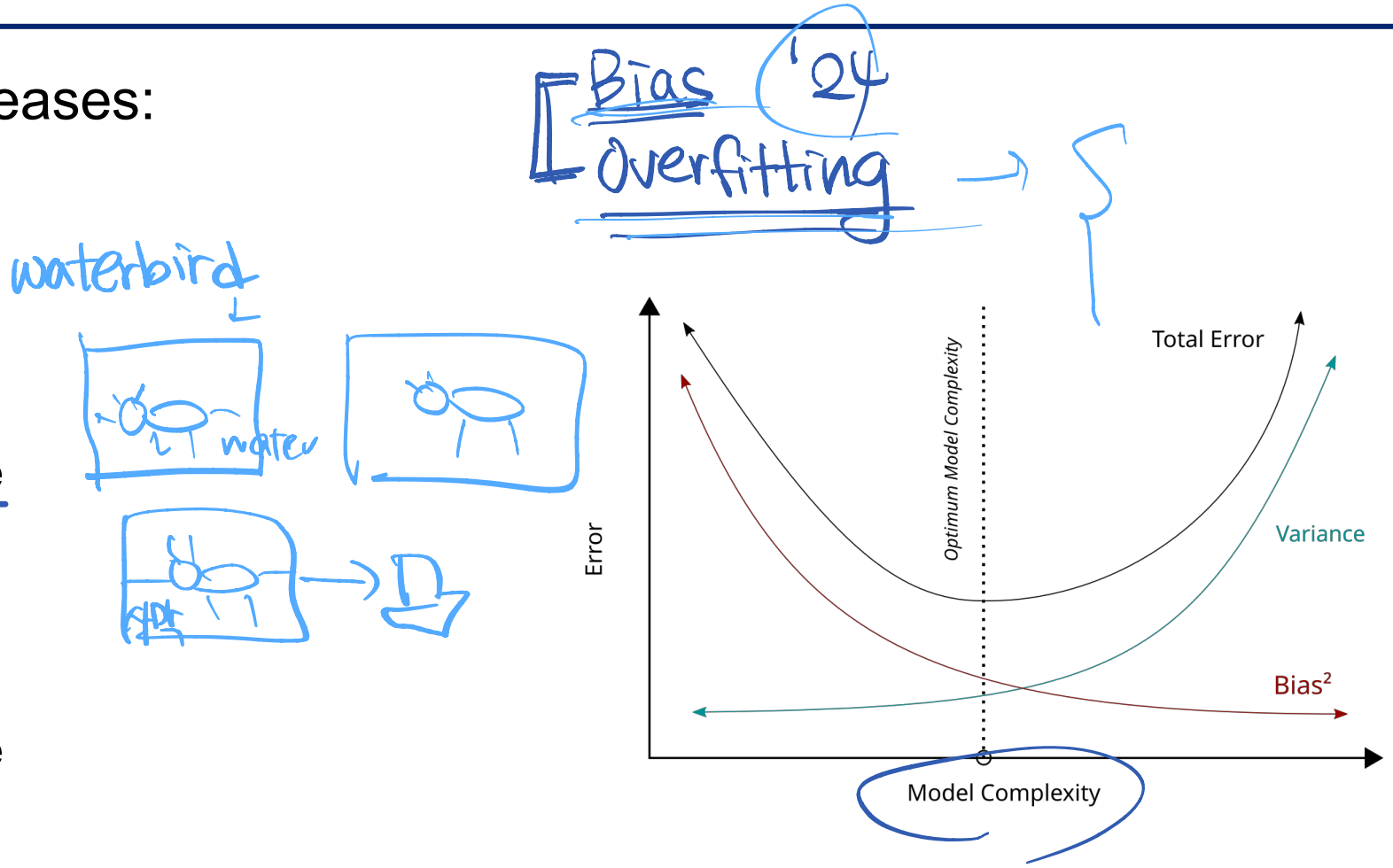
Bias and Variance

- Bias (편향)
  - Error from overly simple assumptions
  - High bias
    - Model is too simple
    - Misses important patterns
    - Often leads to underfitting
- Variance (분산)
  - Sensitivity to changes in the training data
  - High variance
    - Model changes substantially with different training samples
    - Often leads to overfitting



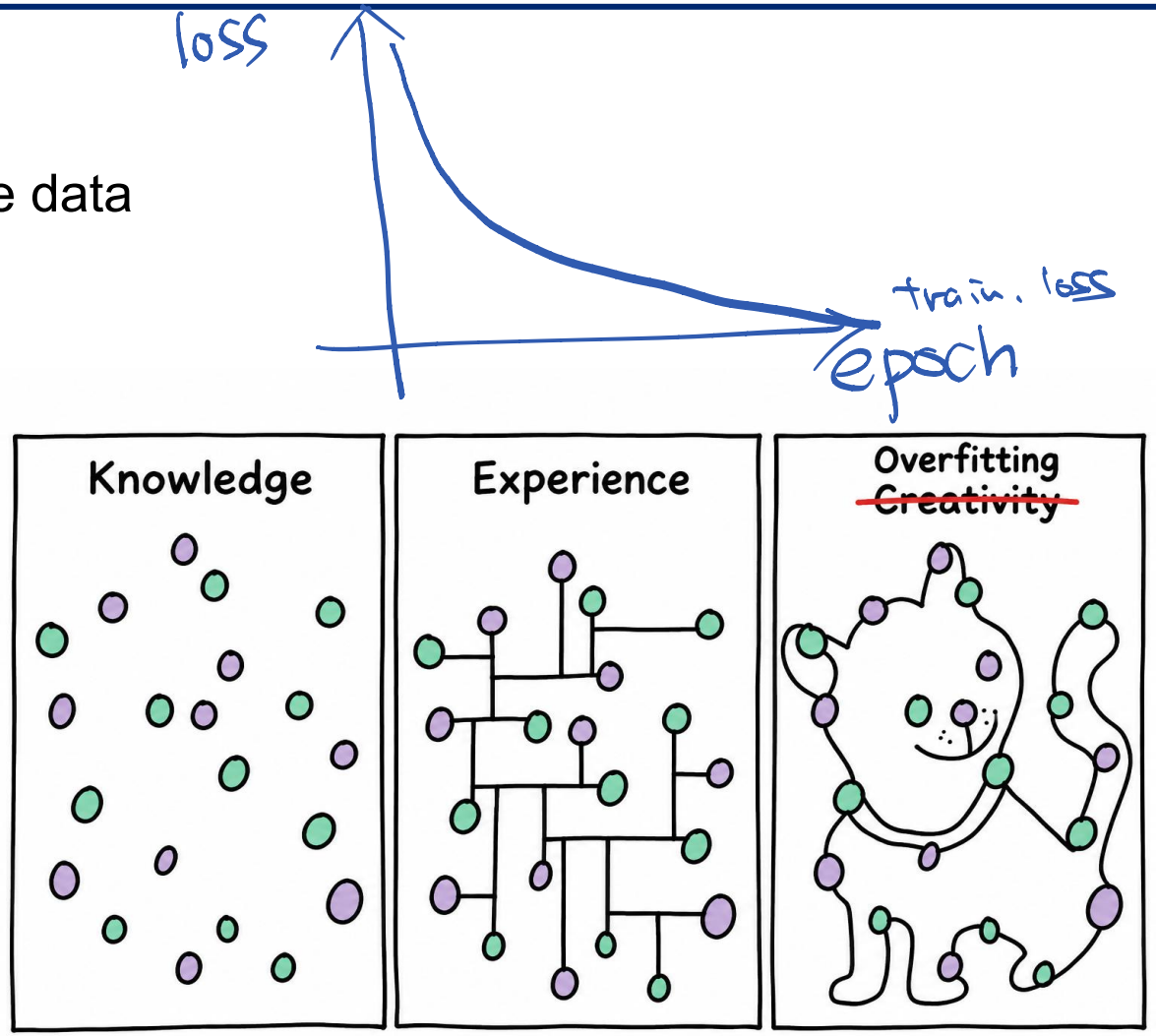
Bias-Variance Tradeoff

- As model complexity increases:
  - Bias decreases
  - Variance increases
- Too simple
  - High bias / Low variance
  - Underfitting
- Too complex
  - Low bias / High variance
  - Overfitting
- Good generalization requires a balance between the two



Why Overfitting Happens

- Overfitting becomes more likely when
  - The model is too complex for the available data
  - The dataset is too small
  - The data contains noise
  - The model is trained for too long
  - Regularization is too weak

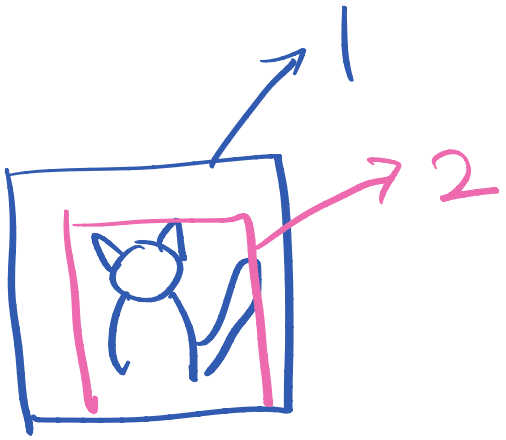


Korea Aerospace University

Part 4. Improving Generalization

Reducing Overfitting

- Several strategies can improve generalization
  - Control model complexity
    - Use a simpler model → ex depth ↓ ...
    - Apply regularization → 규제기법
  - Control training
    - Use early stopping → 학습중지
  - Improve the data
    - Use more training data → data augmentation

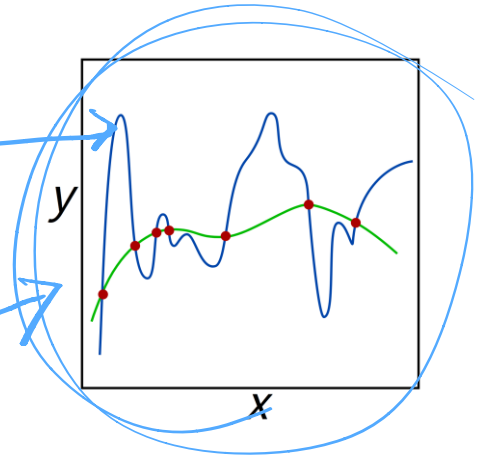


- Goal: reduce the gap between training and unseen-data performance

Regularization

- Regularization discourages a model from becoming unnecessarily complex
- Previously

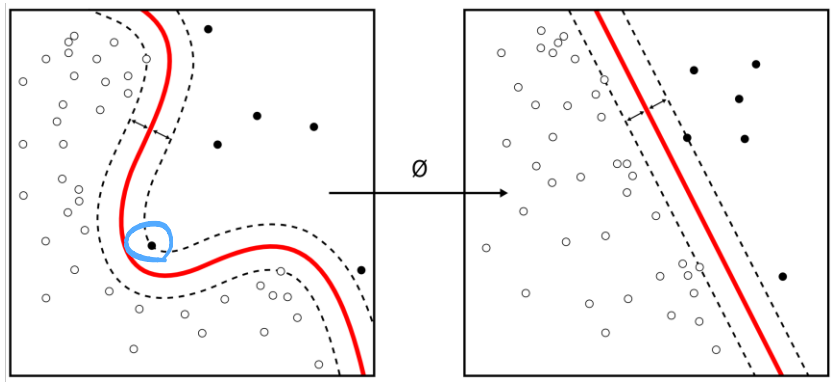
$$\min_{\theta} L_{data}(\theta)$$



- With regularization

$$\min_{\theta} L_{data}(\theta) + \lambda R(\theta)$$

  - L_{data}: prediction error
  - $R(\theta)$: penalty for model complexity
  - λ: regularization strength

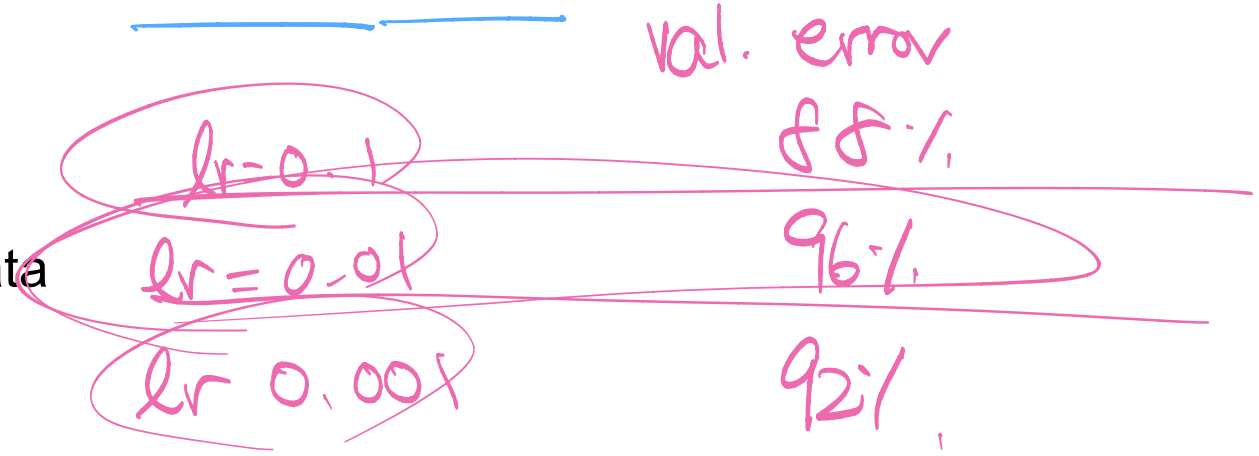


- Regularization trades some training fit for better generalization

Regularization Strength

$$L_{total} = L_{data} + \lambda R(\theta)$$

- Small λ
  - Weak regularization
  - Focus more on fitting training data
  - Higher risk of overfitting
- Large λ
  - Strong regularization
  - Simpler model
  - Higher risk of underfitting
- Choose λ using the validation set



hyperparameter

L1 Regularization

- L1 regularization penalizes the absolute values of the parameters
- Regularized objective

$$L_{total} = L_{data} + \lambda \sum_j |w_j|$$

$\hat{y} = wx + b$

$L_{tot} = L_{data} + \lambda(|w| + |b|)$

- Effect
  - Encourages many weights to become exactly zero
  - Produces a sparse model
  - Can effectively remove unimportant features

$w_1, w_2, \cdots \quad w_{100}$

$w_1, \cdots \quad w_{50}$

$w_{51} \approx 0 \quad w_{100} \approx 0$

L2 Regularization

- L2 regularization penalizes large parameter values
- Regularized objective

$$L_{total} = L_{data} + \lambda \sum_j w_j^2$$

$\hat{y} = wx + b$

$L_{tot} = L_{data} + \lambda(w^2 + b^2)$

- Effect
  - Encourages weights to remain small
  - Reduces model sensitivity to individual training samples
  - Can help reduce overfitting

L1 vs. L2 Regularization

(2, 0.3)

① (1.5, 0) (L1 reg)

② (1.4, 0.15) (L2 reg)

	L1 Regularization	L2 Regularization
Penalty	$\sum_j \lvert w_j \rvert$	$\sum_j w_j^2$
Effect on weights	Can make some weights zero	Makes weights smaller
Result	Sparse model	Small, nonzero weights
Useful for	Feature selection / sparsity	Controlling large weights

기존 (2, 0.3) $L_1 = 2.3$

$L_2 = 4.09$

① $L_1 = 1.5$

$L_2 = 2.25$

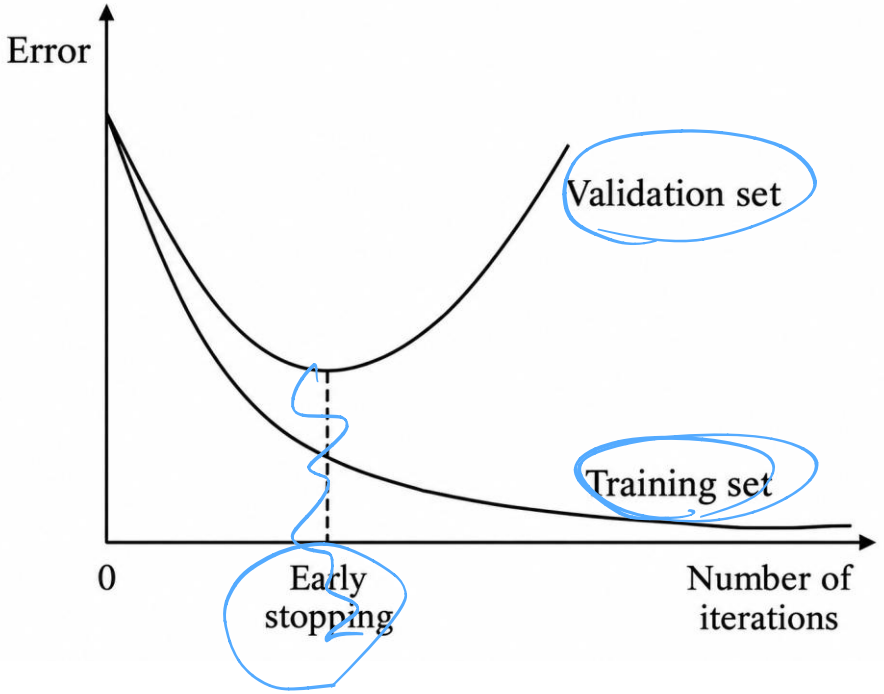
② $L_1 = 1.55$

$L_2 = 1.9825$

~~1.96 + 0~~

Early Stopping

- Training for too long can lead to overfitting

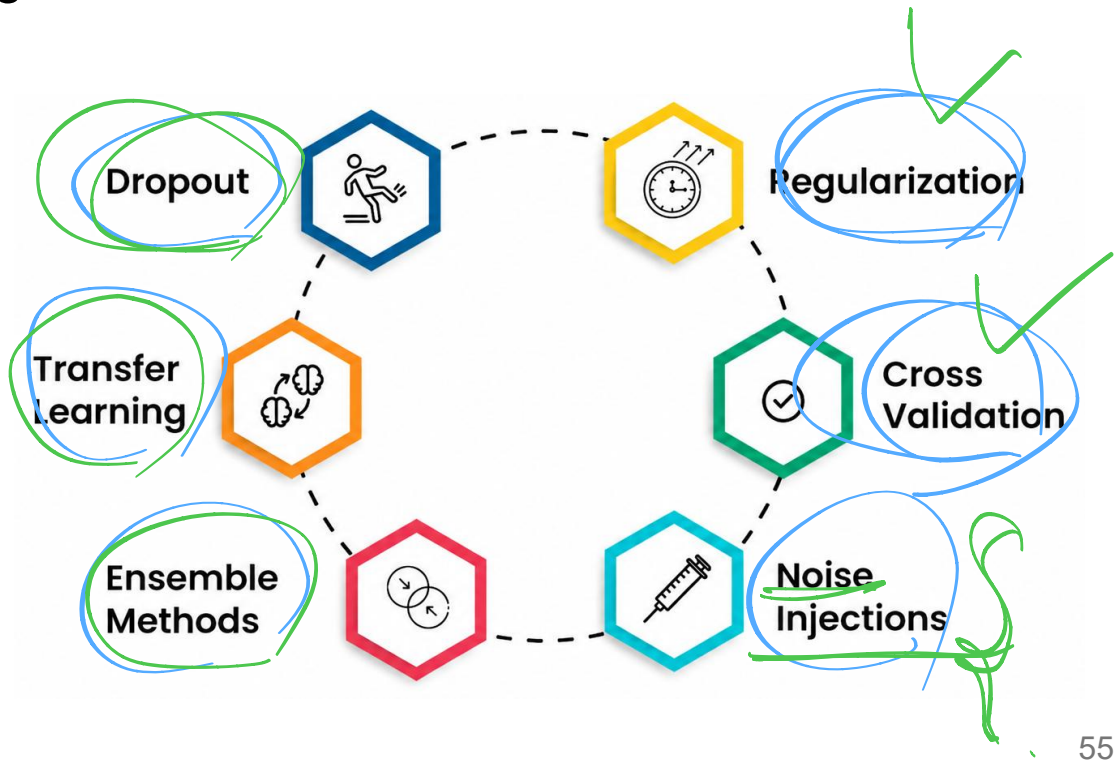


- During training
  - Training error usually continues to decrease
  - Validation error may eventually begin to increase
- Early stopping
  - Stop training when validation performance stops improving
  - Keep the model with the best validation performance

Improving Generalization

- To improve generalization:
  - Choose appropriate model complexity
  - Use regularization to discourage overly complex solutions
  - Use early stopping to avoid training for too long
  - Use more data when possible

- The goal is not to minimize training error, but to perform well on unseen data



Next Lecture

- Lecture 5: Tree-Based Machine Learning Models
  - Decision Trees
  - Splitting Criteria
  - Bagging
  - Random Forests
  - Feature Importance

Korea Aerospace University

Questions?

jiyoonshin@kau.ac.kr