Korea Aerospace University

2AI4203: Machine Learning

Lec06: Support Vector Machines & Boosting

Jiyoon Shin

KAU College of AI Convergence

Oct. 07, 2026

Korea Aerospace University

Linear Model

Recap:

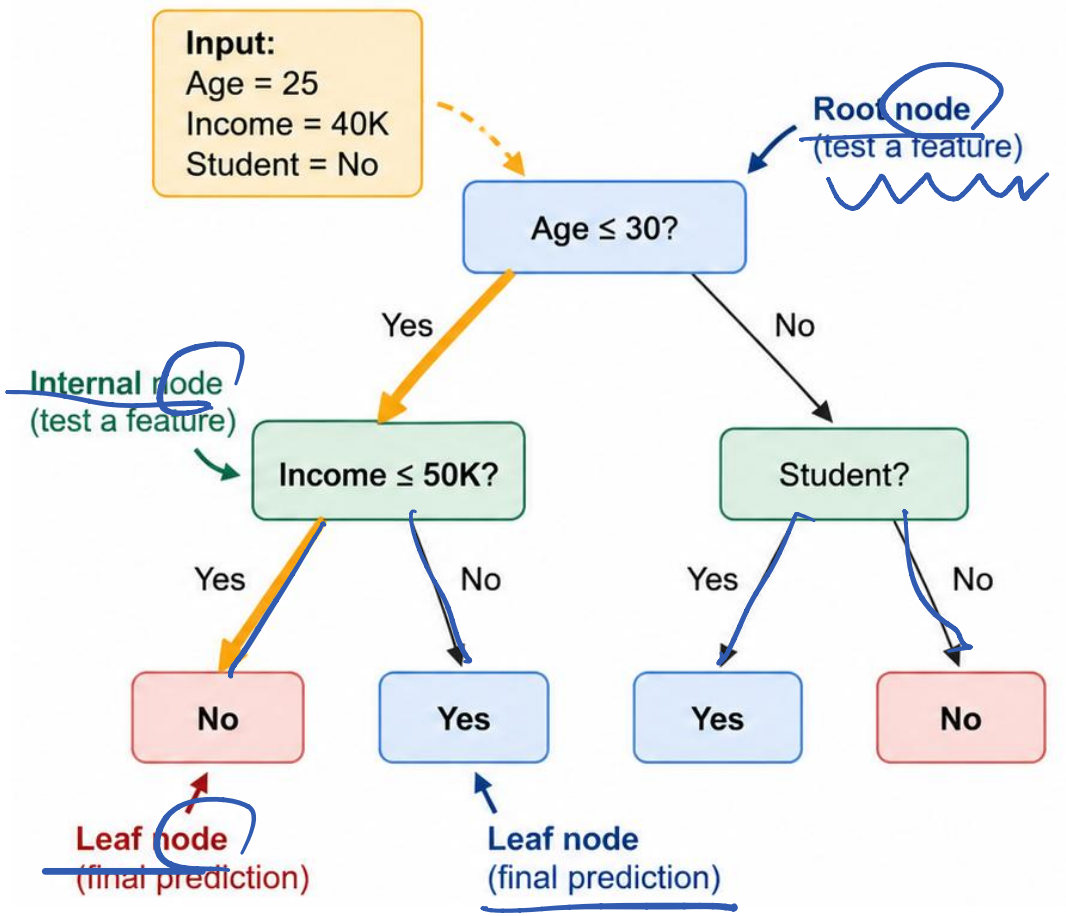
Tree-Based Machine Learning Models

Decision Trees

- A decision tree predicts by applying a sequence of simple decision rules

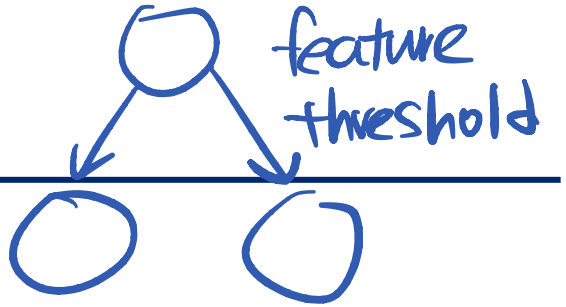
- Internal node: tests a feature
- Branch: outcome of the test
- Leaf node: final prediction

$\hat{y} = w^T x + b$ model



Impurity

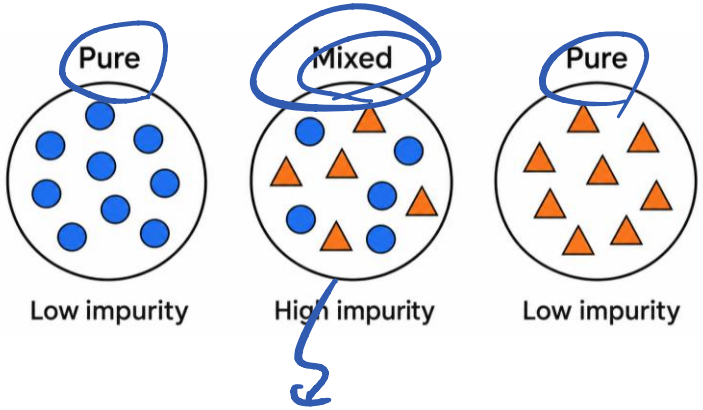
pure: 순수. impure: 섞인 정도.



- Classification trees choose splits that create purer child nodes
- Gini Impurity

$$G = 1 - \sum_{k=1}^{K} p_k^2$$

$= 1 - 0.5^2 - 0.5^2$



- Entropy

$$H = -\sum_{k=1}^{K} p_k \log_2 p_k$$

$= -0.5 \log_2 0.5 - 0.5 \log_2 0.5$

blue : 5 → 0.5

orange : 5 → 0.5

- p_k: proportion of class k
- Pure node → $G = 0, H = 0$
- More mixed → larger impurity

Choosing Splits

- Weighted impurity

$$I_{\text{split}} = \frac{n_L}{n} I_L + \frac{n_R}{n} I_R$$

- Impurity reduction

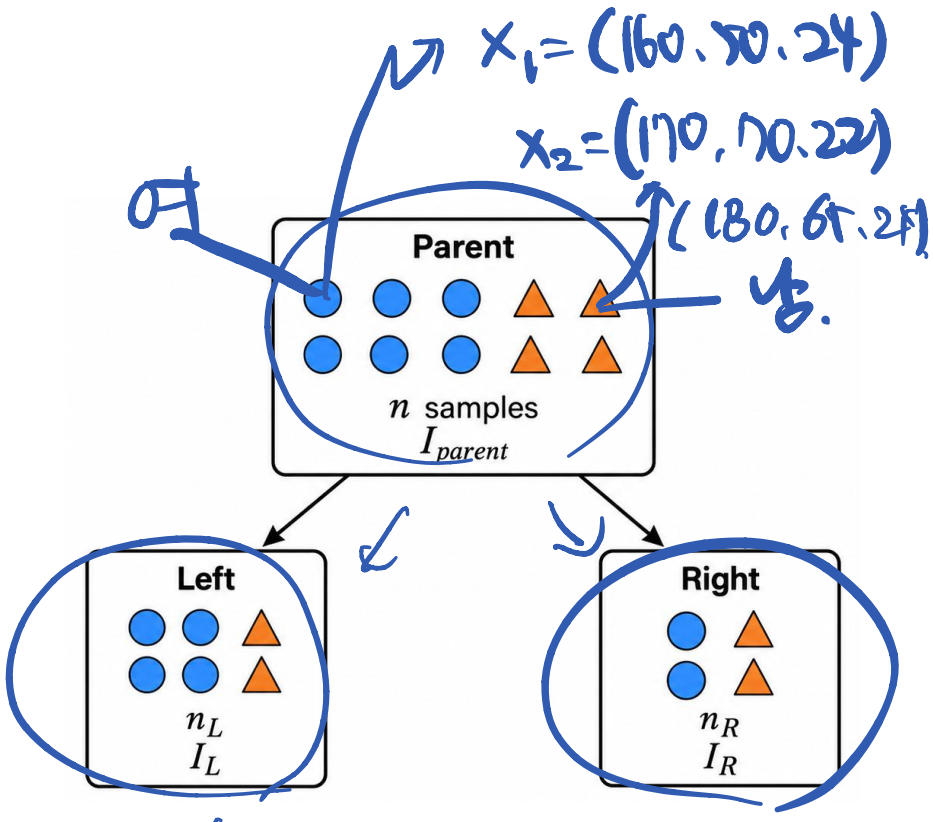
$$\Delta I = I_{parent} - I_{split}$$

  - Larger $\Delta I \rightarrow$ better split
  - Evaluate different features and thresholds

- Choose

$$(j^*, t^*) = \arg\max_{j,t} \Delta I(j,t)$$

  - Repeat recursively to grow the tree



Regression Trees

- Regression trees predict continuous values
- Each leaf predicts the mean target value in its region

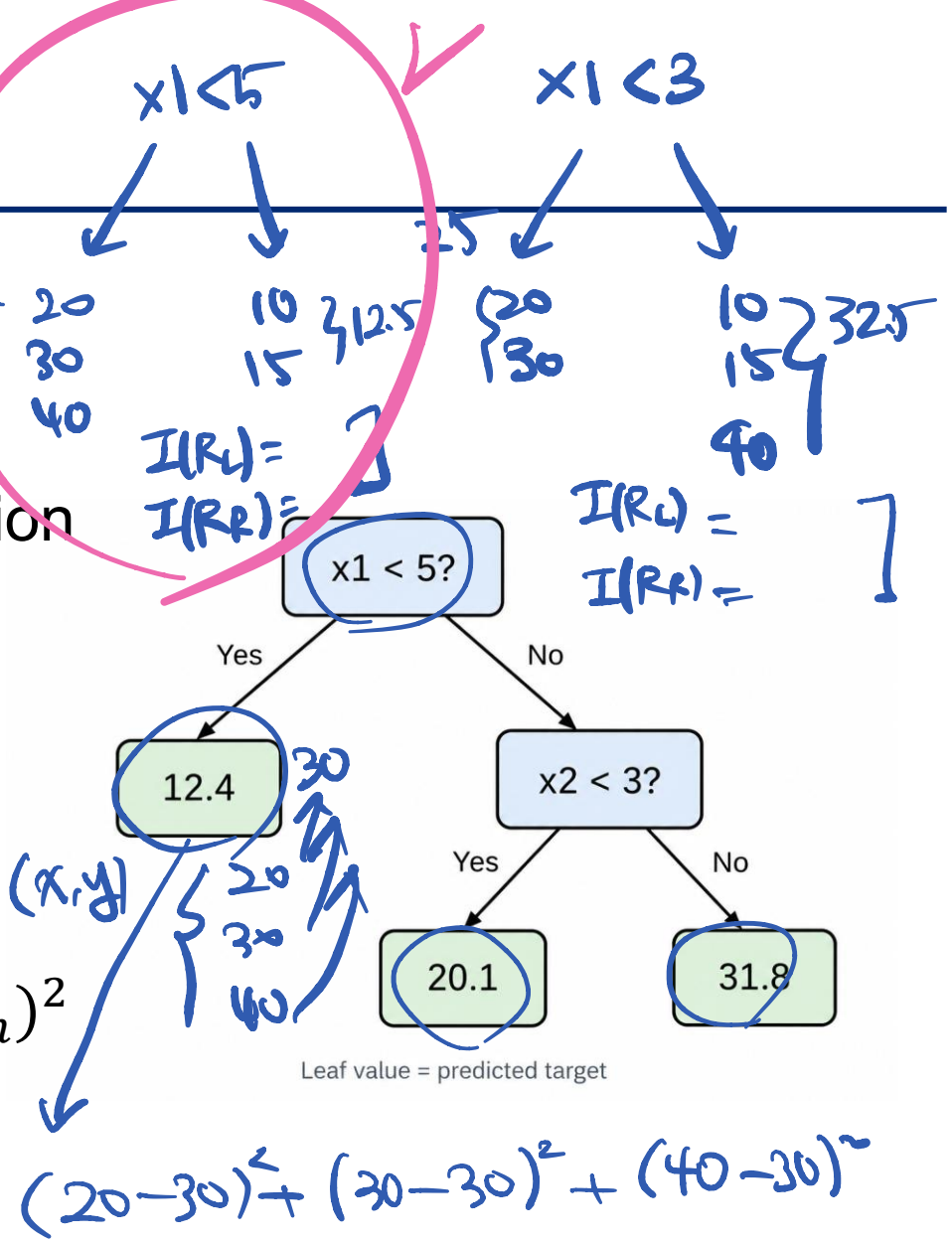
$$c_m = \frac{1}{|R_m|} \sum_{i:\, \mathbf{x}_i \in R_m} y_i$$

- Square error within a region

$$I(R_m) = \sum_{i:\, \mathbf{x}_i \in R_m} (y_i - c_m)^2$$

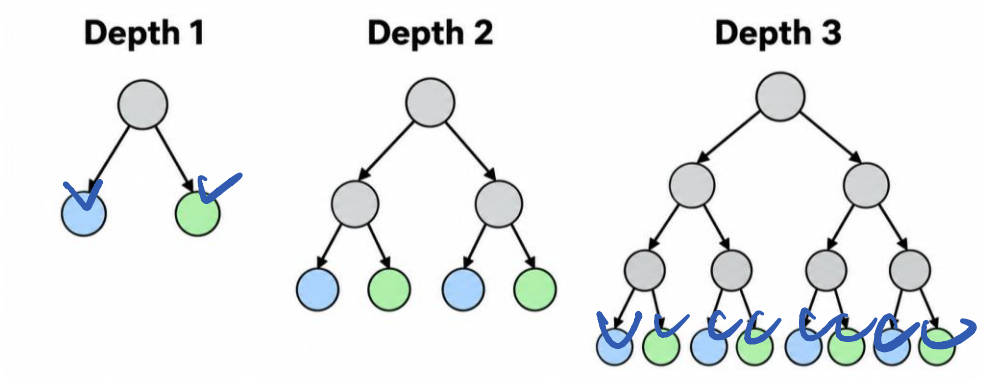
- Choose the split that minimizes

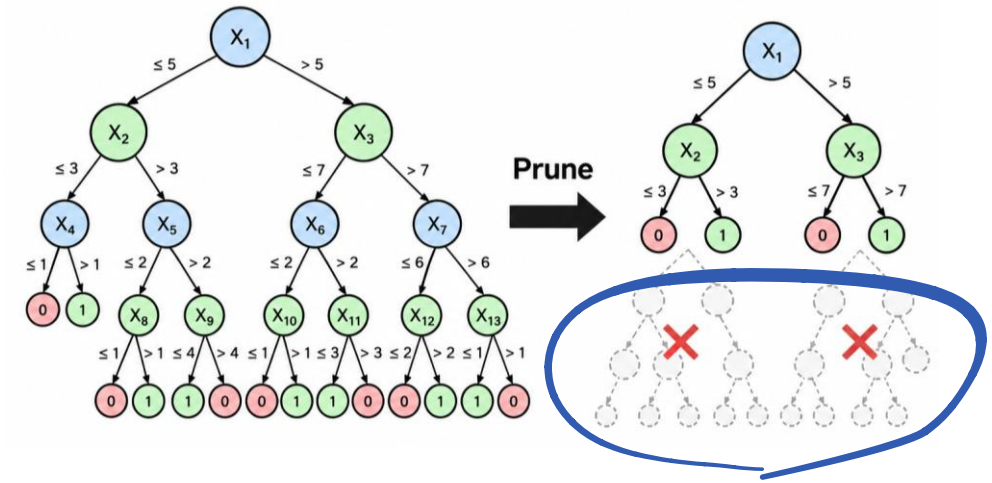
$$I(R_L) + I(R_R)$$



Tree Complexity

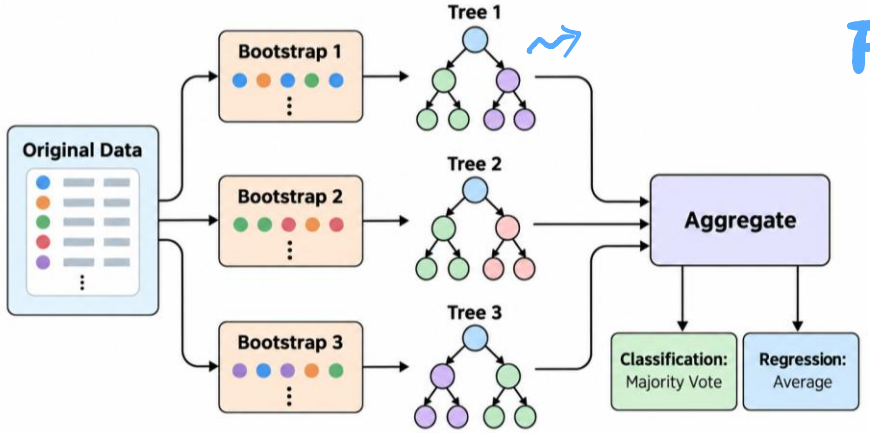
- Increasing tree depth
  - More decision regions
  - Lower training error
  - Higher risk of overfitting
- Control complexity
  - max_depth
  - $min_samples_split$
  - $min_samples_leaf$
  - Pruning
- Model selection
  - Choose complexity using validation data or cross-validation

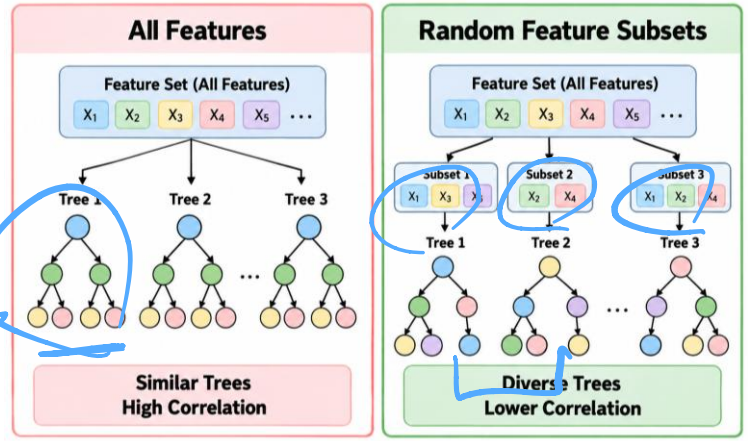


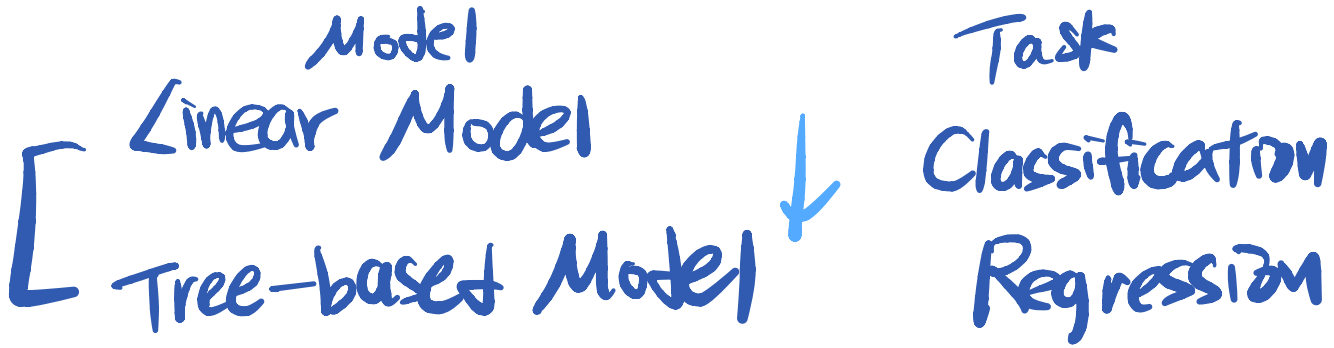


Bagging & Random Forests

- Bagging
  - Train trees on different bootstrap samples
  - Aggregate predictions
    - Classification → majority vote
    - Regression → average
- Random Forests
  - Bagging + Random Feature Subsampling
  - Random feature subsets → more diverse trees
  - Lower correlation → lower variance
- OOB samples
  - Unselected samples can provide an internal performance estimate







Korea Aerospace University

Part 1. Feature Scaling

Different Feature Scales

- Features can have very different numerical ranges

Feature	Example Range
Age	18 – 80
Annual Income	20,000 – 200,000
Rating	1 – 5

- Numerical scale does not necessarily indicate feature importance
- Some machine learning methods are sensitive to these scale differences

Why Scaling Matters

- Large-scale features can dominate distance-based calculations

$$d(\mathbf{x}, \mathbf{z}) = \sqrt{\sum_{j=1}^{d} (x_j - z_j)^2}$$

$X = (18, 20{,}000, 1)$

$Z = (22, 200{,}000, 4)$

- Example

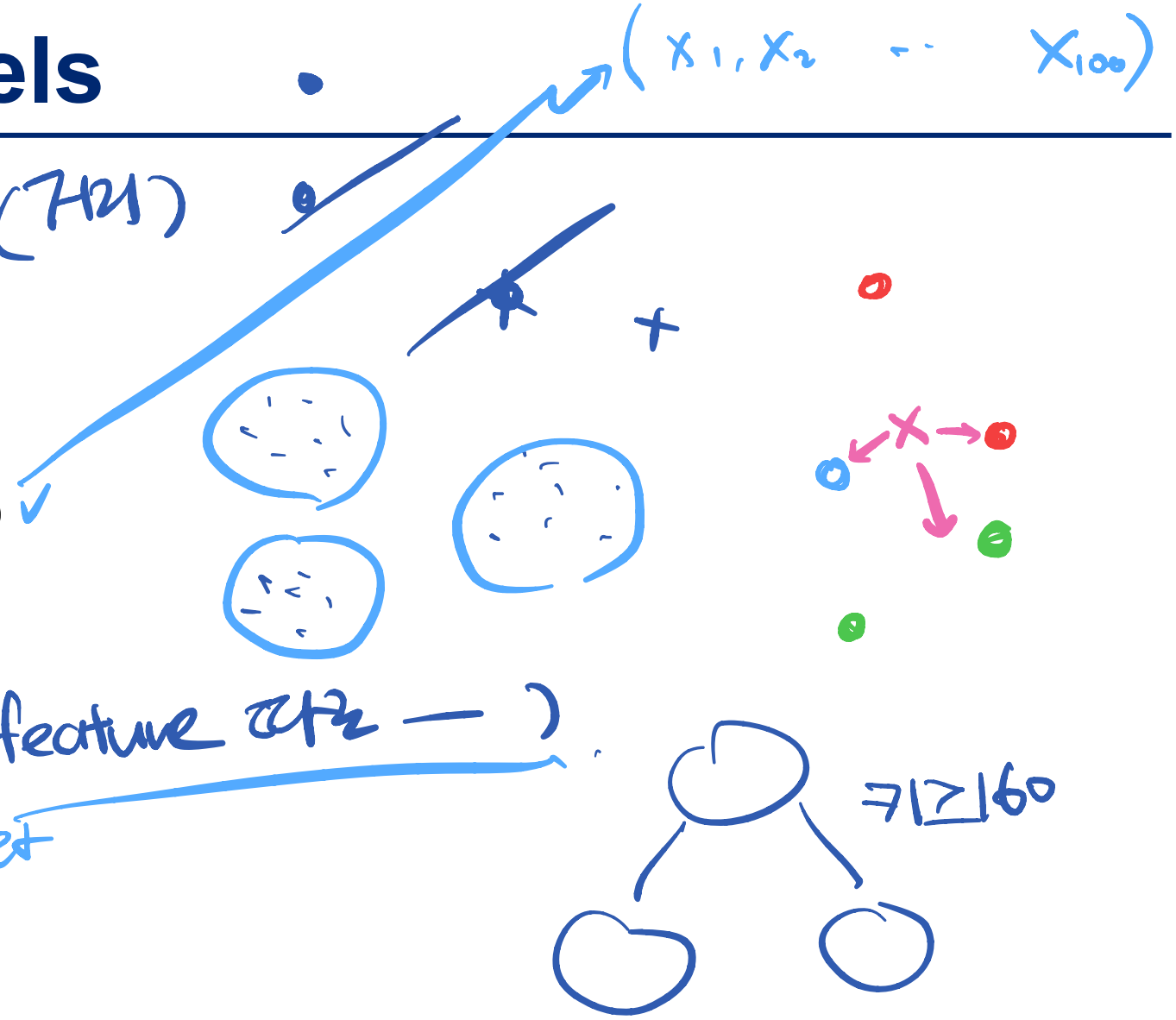
$$\Delta\text{Age} = 10$$

$$\Delta\text{Income} = 20{,}000$$

  - The income feature dominates the distance because of its numerical scale
  - Scaling puts features on comparable numerical scales

Scale-Sensitive Models

- Often sensitive to feature scaling
  - Support Vector Machines (SVM)
  - K-Nearest Neighbors (KNN)
  - K-Means Clustering
  - Principal Component Analysis (PCA)
  - Neural Networks
- Usually less sensitive
  - Decision Trees
  - Random Forests
  - Gradient-Boosted Trees
  - Tree-based models mainly rely on feature ordering and thresholds

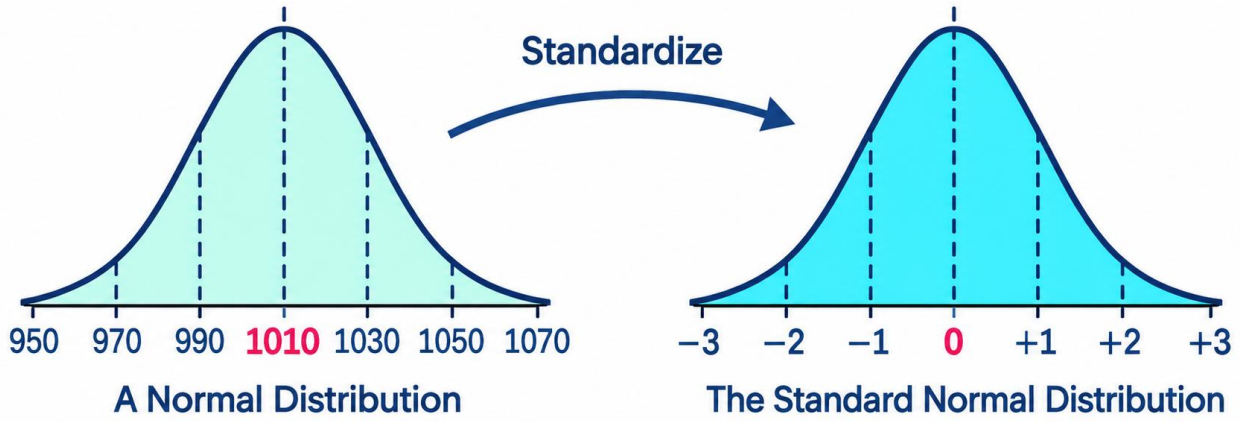


Standardization

- Transform each feature using its mean and standard deviation

$$x_j' = \frac{x_j - \mu_j}{\sigma_j}$$

  - μ_j: mean of feature j
  - σ_j: standard deviation of feature j
  - For the data used to fit the scaler:
    - Mean ≈ 0
    - Standard deviation ≈ 1



- Example

$$x = 80, \quad \mu = 50, \quad \sigma = 15$$

$$x' = \frac{80 - 50}{15} = 2$$

Min-Max Scaling

- Transform each feature to a fixed range

$$x_j' = \frac{x_j - x_j^{\min}}{x_j^{\max} - x_j^{\min}}$$

  - Usually maps the training values to

$$x_j' \in [0, 1]$$

- Example

$$x = 60, \quad x_{\min} = 20, \quad x_{\max} = 100$$

$$x' = \frac{60 - 20}{100 - 20} = 0.5$$

Avoiding Data Leakage

- Incorrect

$$\text{All Data} \longrightarrow \text{Fit Scaler} \longrightarrow \text{Train/Validation/Test Split}$$

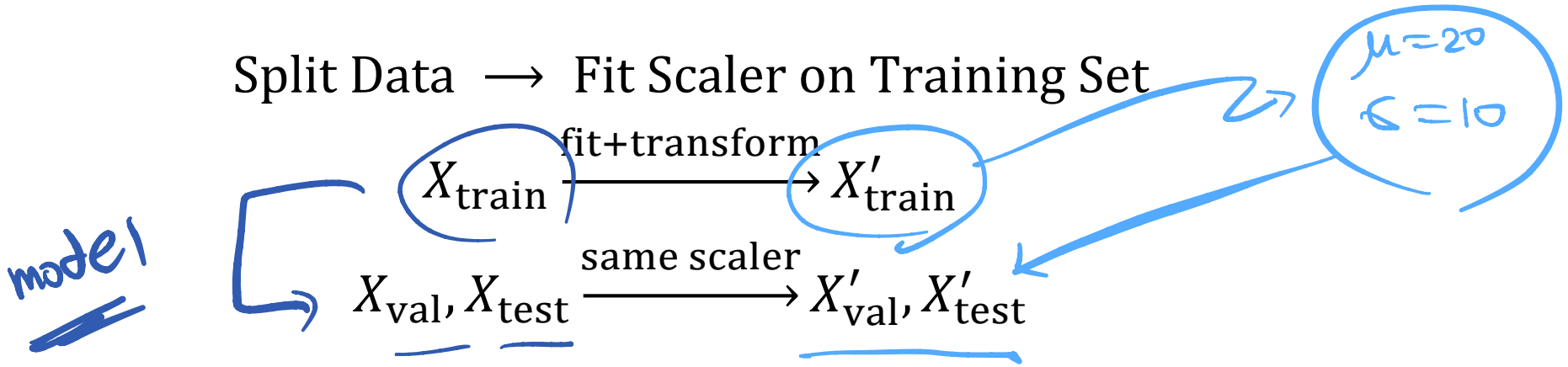
  - Validation/test information leaks into preprocessing

- Correct

$$\text{Split Data} \longrightarrow \text{Fit Scaler on Training Set}$$

$$X_{\text{train}} \xrightarrow{\text{fit+transform}} X'_{\text{train}}$$

$$X_{\text{val}}, X_{\text{test}} \xrightarrow{\text{same scaler}} X'_{\text{val}}, X'_{\text{test}}$$



  - Fit the scaler on training data only
  - Transform validation/test data using the same fitted scaler

Korea Aerospace University

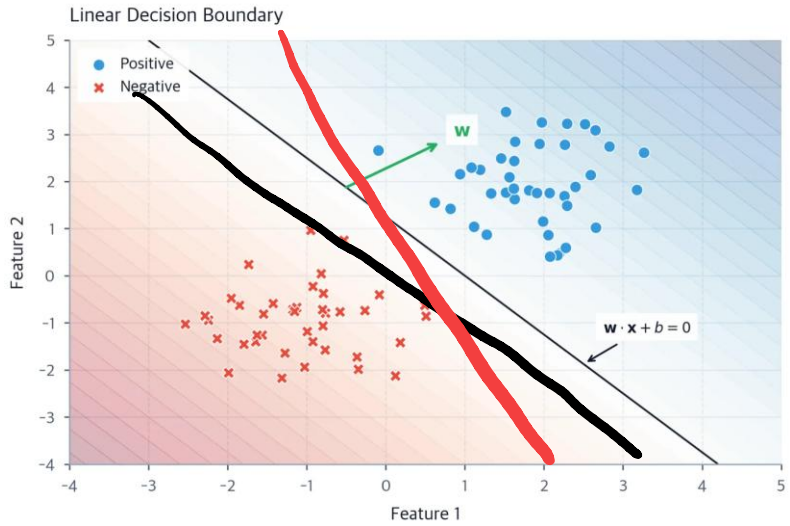
Part 2. Support Vector Machines

(SVM)

Linear Classification

linear regression

$$f(\mathbf{x}) = \mathbf{w}^T\mathbf{x} + b$$



- Decision boundary

$$\mathbf{w}^T\mathbf{x} + b = 0$$

- Prediction

$$\hat{y} = \begin{cases} +1, & f(x) \geq 0 \\ -1, & f(x) < 0 \end{cases}$$

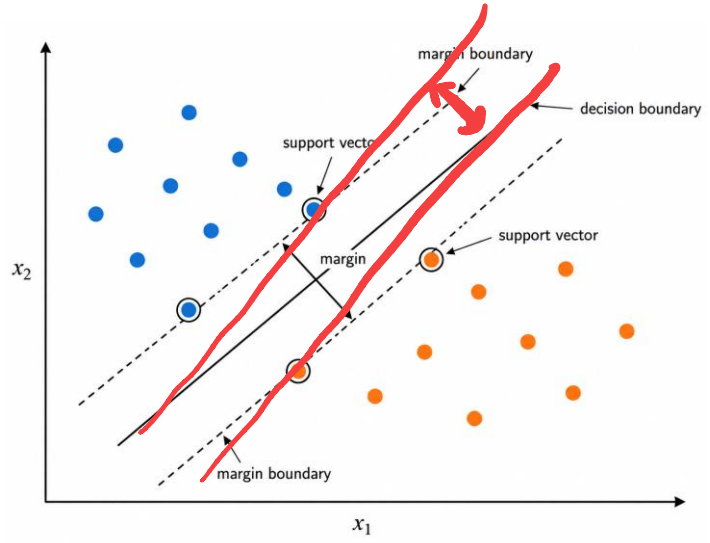
$\hat{y} = \begin{cases} 1 & f(x) \geq 0 \\ 0 & f(x) < 0 \end{cases}$

  - Many different linear boundaries may separate the same training data
  - Which boundary should we choose?

Maximum Margin

- SVM chooses the separating boundary with the largest margin
- The margin is the region between the boundary and the closest samples
- A larger margin keeps the boundary farther from the closest samples

$$\text{SVM} \longrightarrow \text{Maximum} - \text{margin boundary}$$

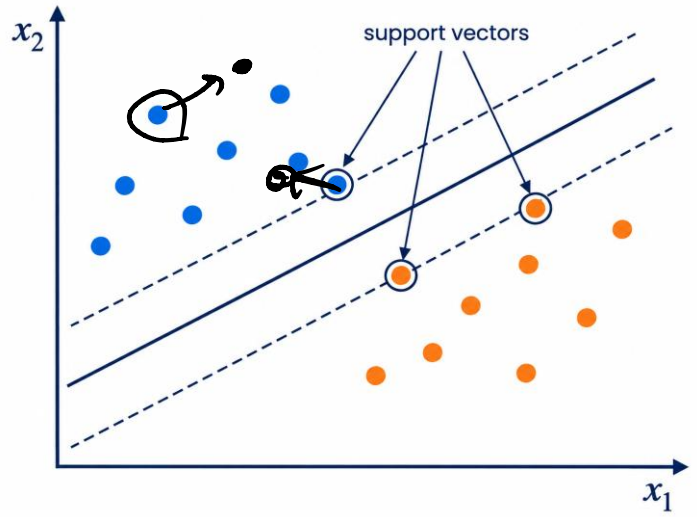


Support Vectors

SVM = Support vector machine

- Support vectors are the training samples closest to the decision boundary
- They determine the position of the maximum-margin boundary
- Samples far from the boundary typically do not determine the solution

Support vectors $\longrightarrow$ closest samples to the boundary



Margin

- Decision boundary
- Margin boundaries
- Margin width

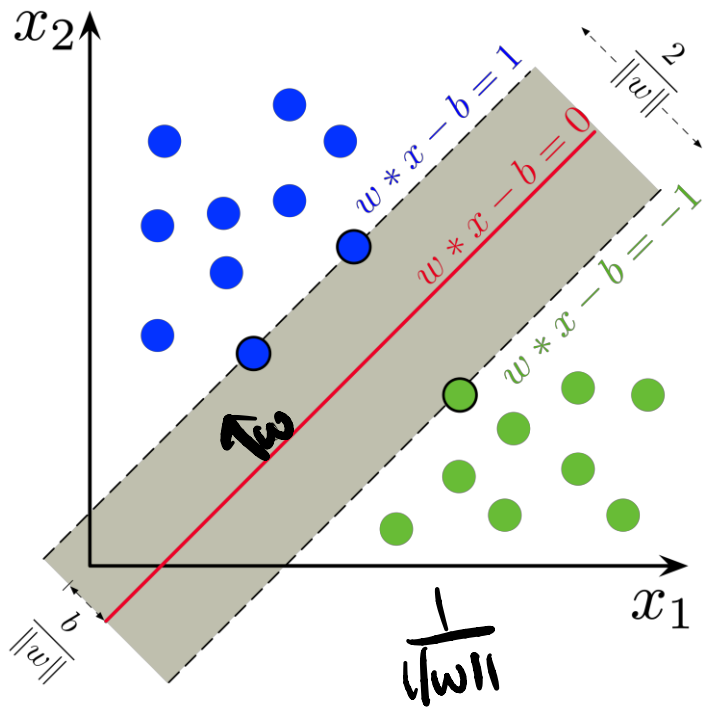
$$\mathbf{w}^T\mathbf{x} + b = 0$$

$$\mathbf{w}^T\mathbf{x} + b = 1$$

$$\mathbf{w}^T\mathbf{x} + b = -1$$

$$\frac{2}{||\mathbf{w}||}$$

Larger margin $\Leftrightarrow$ Smaller $||\mathbf{w}||$



Hard-Margin SVM

- For

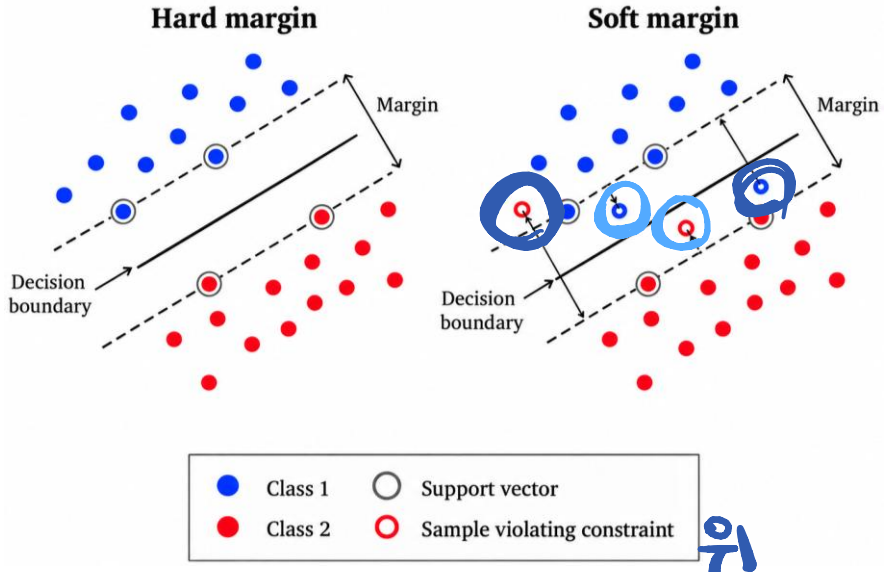
$$y_i \in \{-1, +1\}$$

maximizing the margin is equivalent to

$$\min_{\mathbf{w},b} \frac{1}{2}||\mathbf{w}||^2$$

subject to

$$y_i(\mathbf{w}^T\mathbf{x}_i + b) \geq 1$$



- All training samples must be correctly classified and outside the margin
- Works only when the data is linearly separable

Soft-Margin SVM

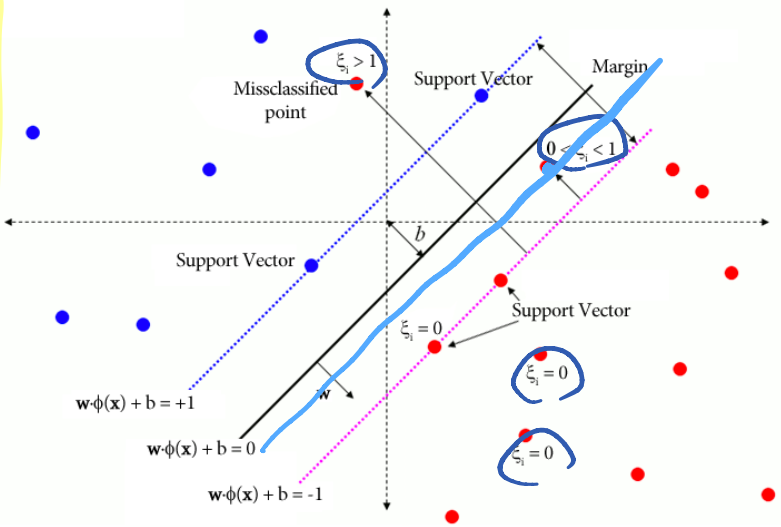
- Real-world data may contain overlap, noise, or outliers
- Allow some samples to enter the margin or be misclassified

subject to

$$\min_{\mathbf{w},b,\xi} \frac{1}{2}||\mathbf{w}||^2 + C\sum_{i=1}^{n}\xi_i$$

$$y_i(\mathbf{w}^T\mathbf{x}_i + b) \geq 1 - \xi_i$$

$$\xi_i \geq 0$$



- ξ_i: slack variable (margin violation)
- C: penalty for violations

① $\xi_i = 0$

$y_i(w^T x_i + b) \geq 1$

② $1 < \xi_i$

$w^T x_i + b \geq 1$

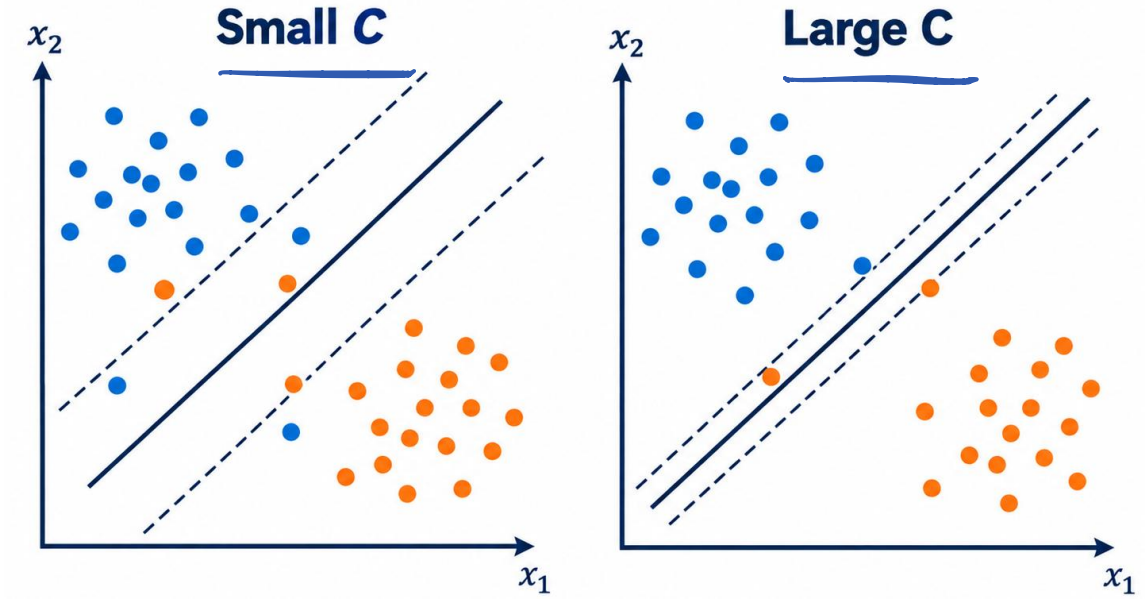
$y_i = -1$

③ $0 < \xi_i < 1 \rightarrow w^T x_i + b \geq 1$

$y_i = +1$

The Role of C

- Small C
  - Allows more margin violations
  - Wider margin
  - Stronger regularization
  - Higher risk of underfitting
- Large C
  - Penalizes violations strongly
  - Fits training data more closely
  - Narrower margin
  - Higher risk of overfitting



$$C \longrightarrow \text{margin} - \text{error trade} - \text{off}$$

- Choose C using validation data or cross-validation

Hinge Loss

$y_i(w^T x_i + b) \geq 1 - \xi_i$

$$L_{\text{hinge}} = \max(0, 1 - yf(x))$$

$\xi_i \geq 1 - y_i(w^T x_i + b)$

$\xi_i \geq 1 - y f(x)$

- If

  $yf(x) \geq 1$

  then

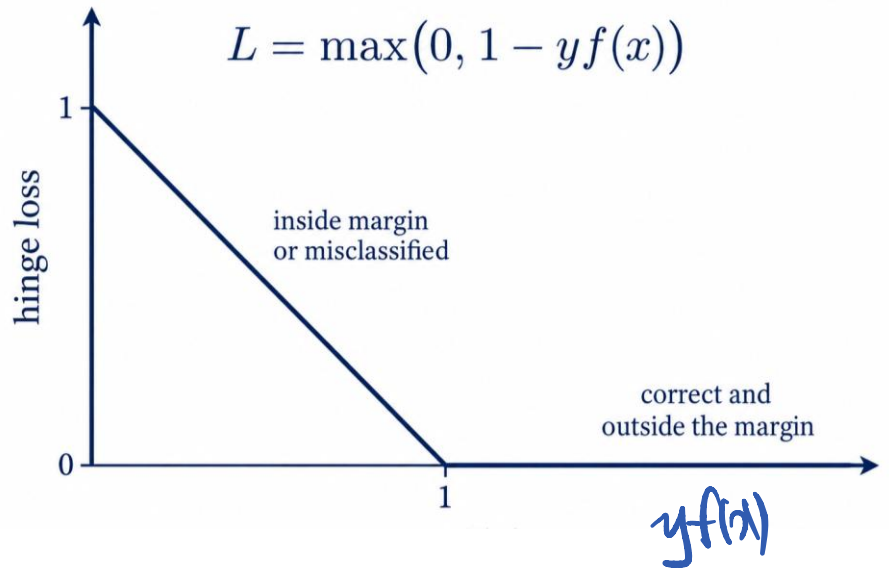
  $L_{\text{hinge}} = 0$

- If

  $yf(x) < 1$

  then

  $L_{\text{hinge}} > 0$



- Penalizes samples inside the margin or on the wrong side of the boundary

SVM Objective

$$\frac{1}{2}||\mathbf{w}||^2 + C\sum_{i=1}^{n}\max(0, 1 - y_i f(\mathbf{x}_i))$$

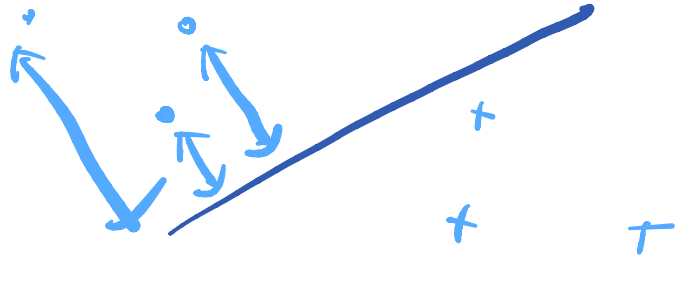
$$\frac{1}{2}||\mathbf{w}||^2 + C\sum_{i} L_{hinge,i}$$

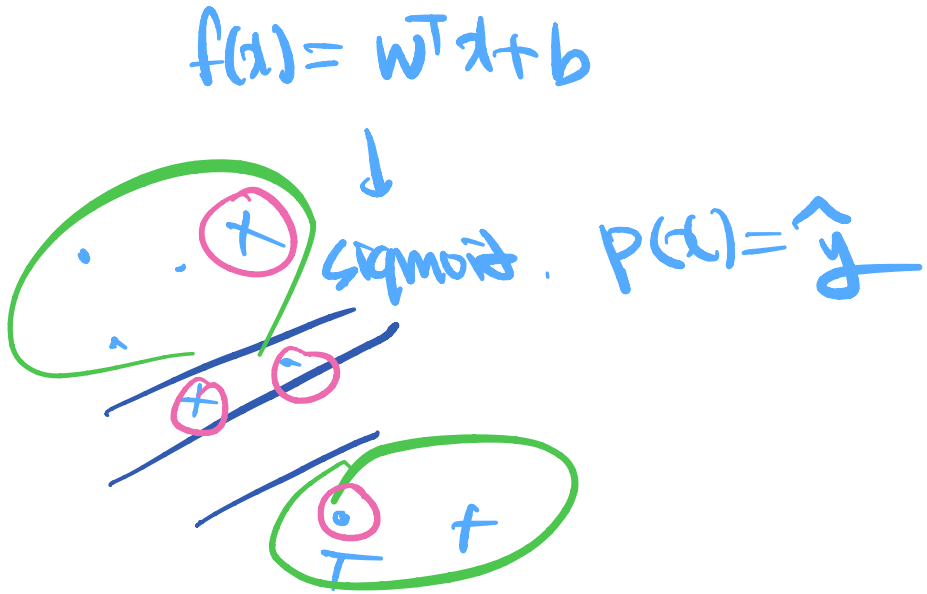
- First term encourages a larger margin
- Second term penalizes margin violations
- C controls the balance between the two

Linear SVM vs. Logistic Regression

	Logistic Regression	SVM
Main idea	Model class probabilities	Maximize the margin
Loss	Cross-entropy	Hinge loss
Output	Probability	Decision score
Boundary	Linear	Linear

- Both can learn linear decision boundaries
- They use different learning objectives





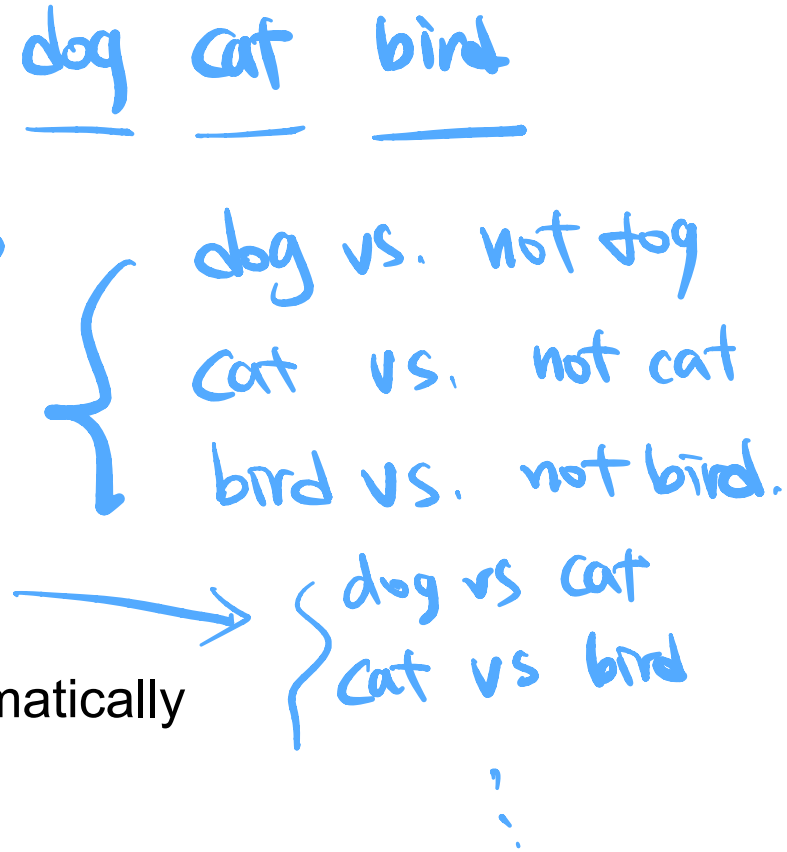
Multiclass SVM

- Basic SVM formulation is binary
- Multiclass classification can combine multiple binary SVMs

- One-vs-Rest (OvR)
  - Train one classifier for each class against all others
- One-vs-One (OvO)
  - Train classifiers for pairs of classes

$$K \text{ classes} \rightarrow \frac{K(K-1)}{2} \text{ classifiers (OvO)}$$

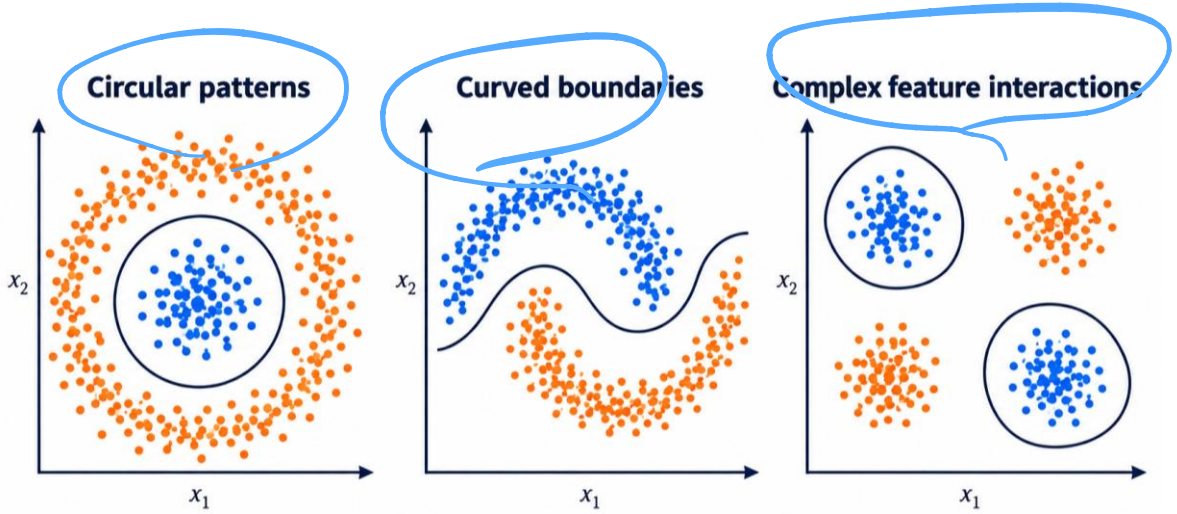
  - Common SVM libraries handle multiclass classification automatically



Beyond Linear Boundaries

- Linear SVM produces a linear decision boundary
- Some datasets cannot be separated well by a straight line or hyperplane
- Nonlinear class structures require more flexible boundaries

- Examples
  - Circular patterns
  - Curved boundaries
  - Complex feature interactions



- Can SVMs learn nonlinear decision boundaries?

Korea Aerospace University

Part 3. Kernel Methods

Feature Mapping

- Transform the original input into a new feature space

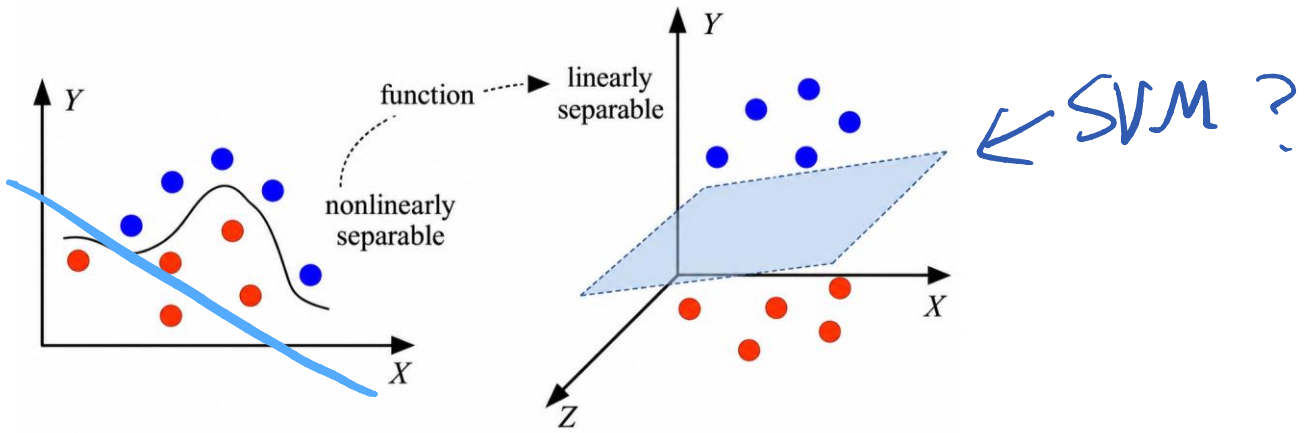
$$\mathbf{x} \longrightarrow \phi(\mathbf{x})$$

linear decision

- Example

$$x \longrightarrow \phi(x) = \begin{bmatrix} x \\ x^2 \end{bmatrix}$$

  - Data that is not linearly separable in the original space may become linearly separable after transformation

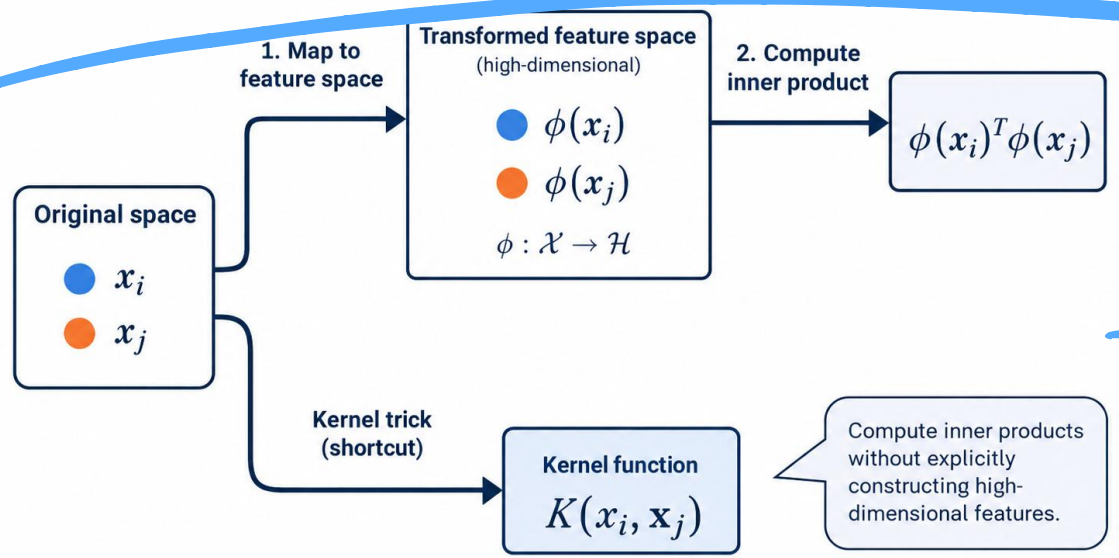


The Kernel Trick

- Compute inner products in the transformed space using a kernel function

$$K(\mathbf{x}_i, \mathbf{x}_j) = \phi(\mathbf{x}_i)^T \phi(\mathbf{x}_j)$$

  - Computes similarity in the transformed feature space
  - Avoids explicitly constructing high-dimensional features



$$x \longrightarrow \phi(x) = (x, x^2)$$

①

$x_1 = 2 \rightarrow \phi(x_1) = (2, 4)$

$x_2 = 3 \rightarrow \phi(x_2) = (3, 9)$

$(2, 4)^T (3, 9) = 2 \cdot 3 + 4 \cdot 9$

$= 6 + 36 = 42$

$x_3 = 4 \rightarrow \phi(x_3) =$

$x_5 = 5 \rightarrow \phi(x_5) =$

②

$K(x_i, x_j) = x_i x_j + x_i^2 x_j^2$

$K(2, 3) = 2 \cdot 3 + 4 \cdot 9$

$= 42$

Common Kernels

$k(x_i, x_j) = \phi(x_i)^T \phi(x_j)$

- Linear

$$K(\mathbf{x}, \mathbf{z}) = \mathbf{x}^T \mathbf{z}$$

- Polynomial

$$K(\mathbf{x}, \mathbf{z}) = (\mathbf{x}^T \mathbf{z} + c)^d$$

  - d: degree
  - c: constant

- RBF (Gaussian)

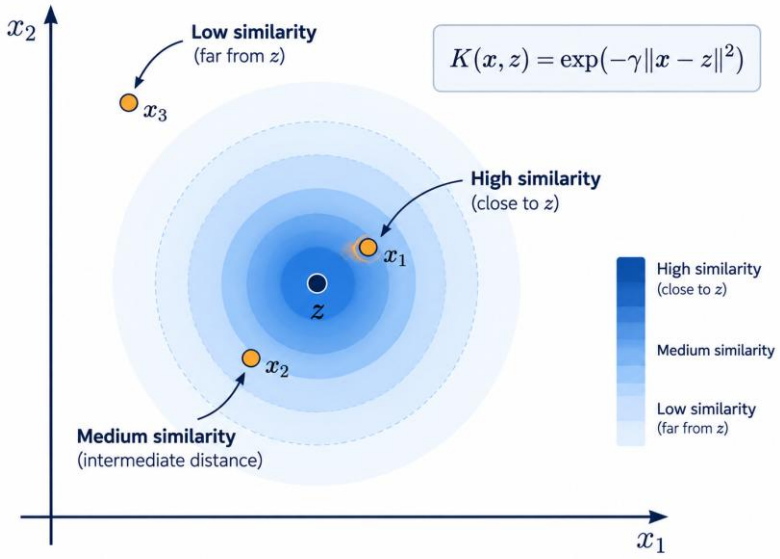
$$K(\mathbf{x}, \mathbf{z}) = \exp(-\gamma \left|\left|\mathbf{x} - \mathbf{z}\right|\right|^2)$$

RBF Kernel

$$K(\mathbf{x}, \mathbf{z}) = \exp(-\gamma ||\mathbf{x} - \mathbf{z}||^2)$$

- Nearby samples have high similarity
- Distant samples have low similarity
- Can produce flexible nonlinear decision boundaries

$$||\mathbf{x} - \mathbf{z}||^2 \downarrow \Longrightarrow K(\mathbf{x}, \mathbf{z}) \uparrow$$



- Because the RBF kernel is distance-based, feature scaling is important

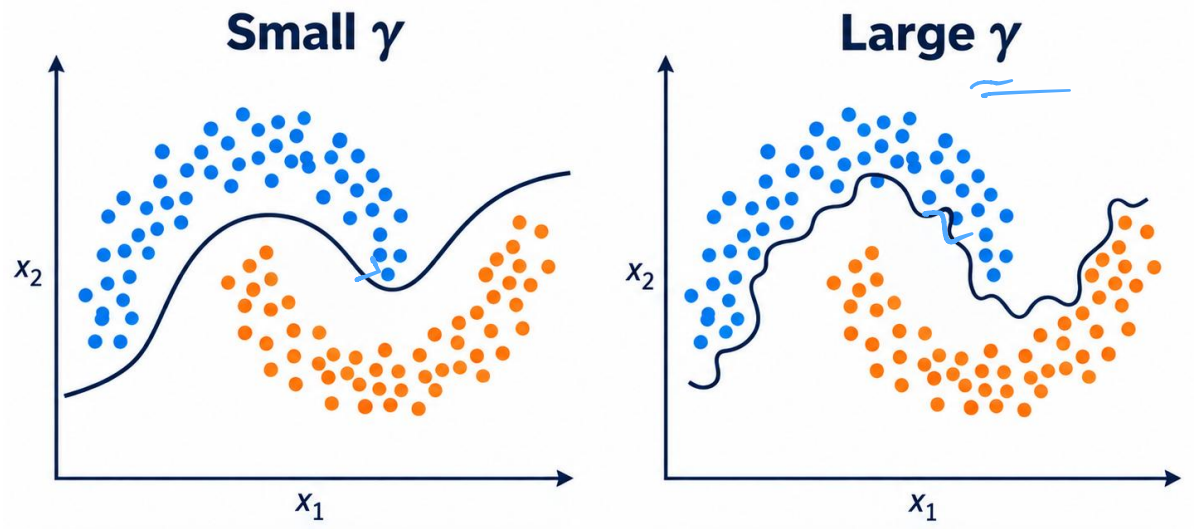
The Role of γ

$$k(x, z) = \exp(-\gamma \|x - z\|^2)$$

- Small γ 0.01
  - Broad influence of each sample
  - Smoother, less complex boundary
- Large γ 1
  - Local influence of each sample
  - More complex boundary
  - Greater risk of overfitting

$x - z = 1$ $\exp(-0.01)$

$x - z = 2$ $\exp(-0.04)$



$x - z = 1$ $\exp(-1)$

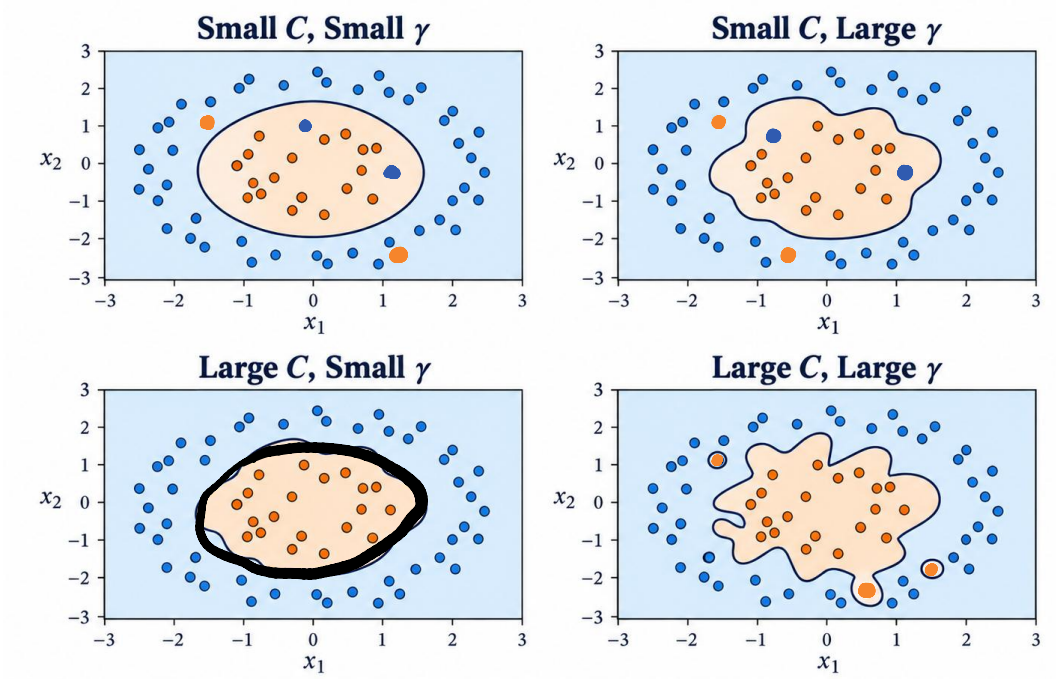
$x - z = 2$ $\exp(-4)$

Tuning C and γ

- C: controls the penalty for margin violations
- γ: controls the range of influence in the RBF kernel

$$(C, \gamma) \longrightarrow \text{decision boundary complexity}$$

- Tune C and γ together
- Use validation data or cross-validation
- Do not tune them using the test set



Linear vs. RBF SVM

	Linear SVM	RBF SVM
Boundary	Linear	Nonlinear
Flexibility	Lower	Higher
Hyperparameters	C	C, γ
Computation	Usually faster	Usually more expensive
Best for	Roughly linear patterns	Nonlinear patterns

Korea Aerospace University

Part 4. Boosting

Bagging vs. Boosting

	Bagging	Boosting
Training	Independent	Sequential
Main idea	Train diverse learners	Correct the current model
Main effect	Reduce variance	Reduce bias / correct errors
Combination	Average / vote	Additive combination
Example	Random Forest	Gradient Boosting

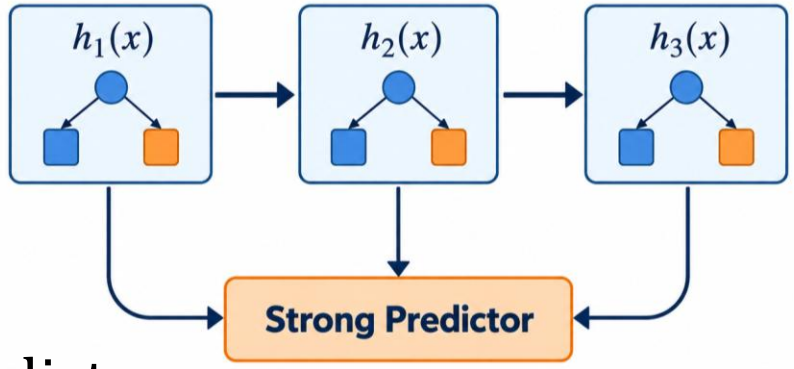
- Bagging trains models independently
- Boosting trains models one after another

Boosting

- Combine many weak learners into a strong predictor
- Weak learners are typically simple models, such as shallow decision trees
- Each new learner focuses on what the current model predicts poorly

$$F(x) = \sum_{m=1}^{M} \alpha_m h_m(x)$$

- h_m: weak learner
- α_m: weight of learner m

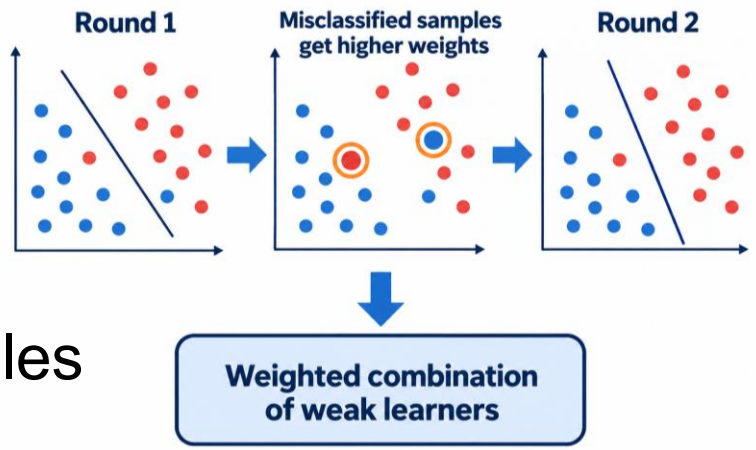


Weak learners $\longrightarrow$ Strong predictor

AdaBoost

Adaptive Boosting

- Start with equal weights for all training samples
- Train a weak learner
- Increase the weights of misclassified samples
- Train the next learner with more focus on difficult samples



Misclassified samples ⟶ Higher weights ⟶ Next learner

- Better learners receive greater weight in the final prediction

$L_1 = L(x_1) + L(x_2) + L(x_3)$

x_1 틀림!

$L_2 = 2L(x_1) + L(x_2) + L(x_3)$

Gradient Boosting

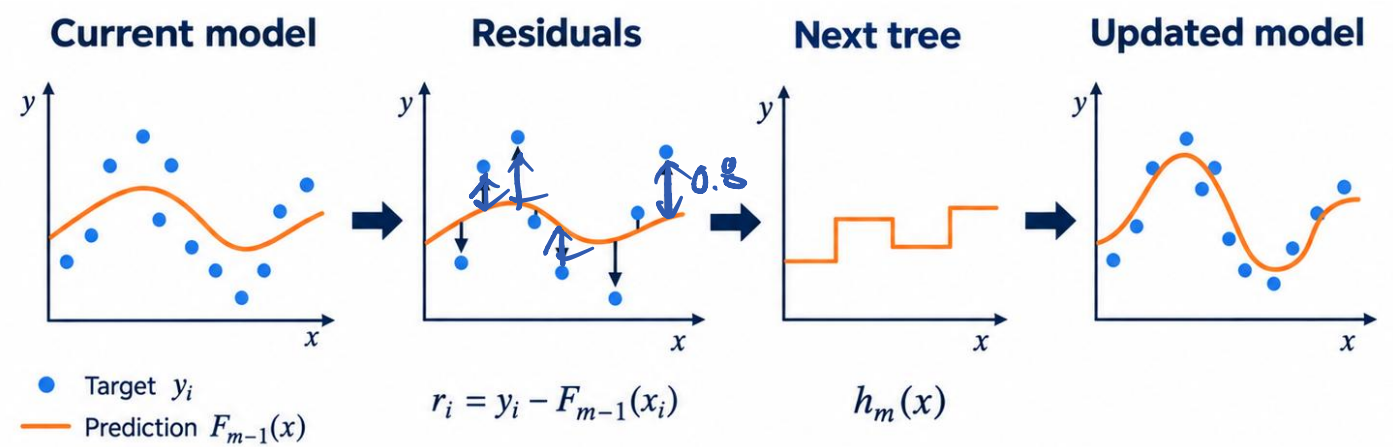
Gradient Descent 비슷!

- Build models sequentially
- Each new learner tries to correct the current model
- For squared-error regression, the correction is based on residuals

$$r_i = y_i - F_{m-1}(\mathbf{x}_i)$$

- Train the next tree to predict r_i

$$\text{Current Model} \longrightarrow \text{Residuals} \longrightarrow \text{Next tree}$$



Learning Residuals

- Targets:

$$y = [10, 20, 30]$$

- Current predictions:

$$\hat{y} = [12, 16, 26]$$

- Residuals:

$$\mathrm{r} = \mathrm{y} - \hat{y} = [-2, 4, 4]$$

  - The next tree learns these residuals
  - Its predictions correct the current model

두번째 model 예측

[-1, 2, 3]

$\hat{y}_2 = [11, 18, 29]$

$y - \hat{y}_2 = [-1, 2, 1]$ ←

세번째 model 예측 = [-0.5, 2, 0.5]

Additive Learning

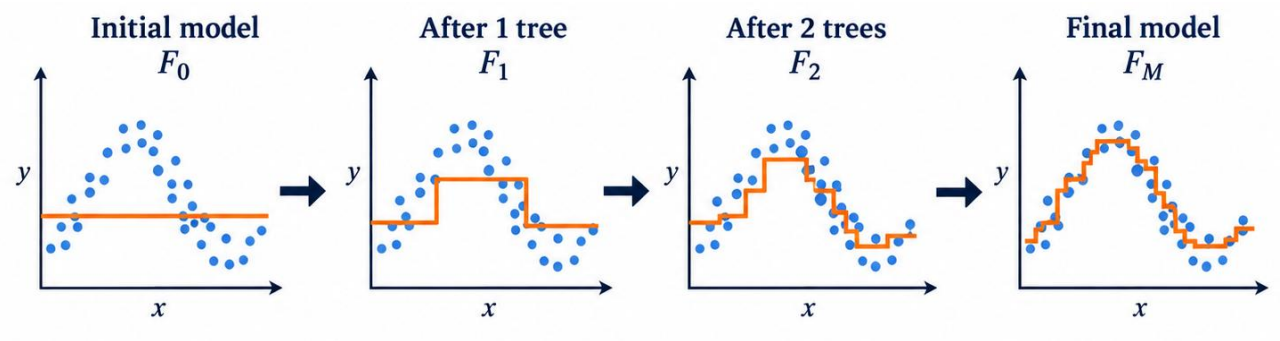
- Update the model sequentially:

$$F_0 \longrightarrow F_1 \longrightarrow F_2 \longrightarrow \ldots \longrightarrow F_M$$

$$F_m(\mathbf{x}) = F_{m-1}(\mathbf{x}) + \eta h_m(\mathbf{x})$$

residual 예측값

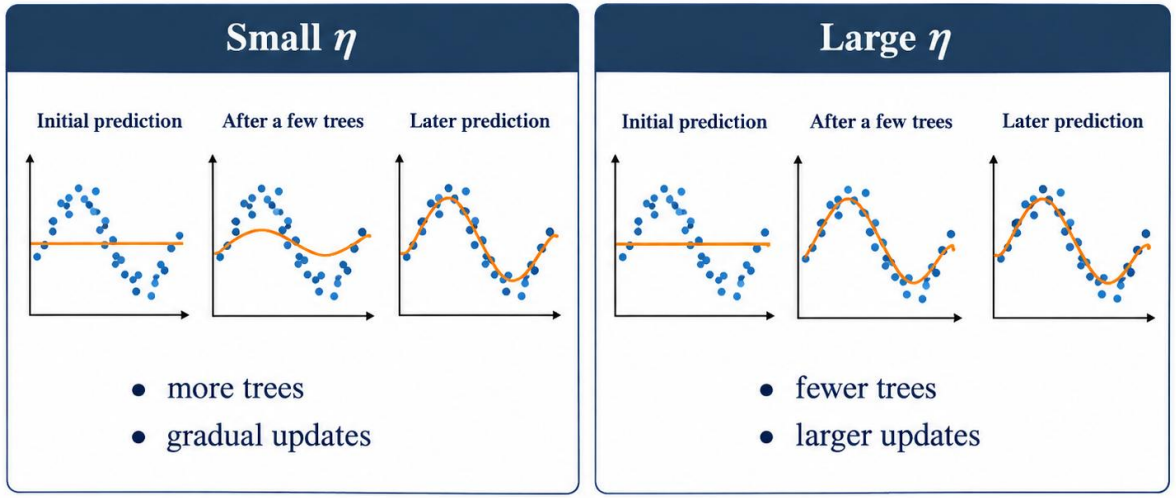
  - F_{m-1}: current model
  - h_m: new tree
  - η: learning rate



- Each new tree makes an additional correction

Learning Rate

- Small η
  - Smaller corrections
  - Usually requires more trees
  - Can improve generalization



- Large η
  - Larger corrections
  - Fewer trees may be needed
  - Greater risk of overfitting

$$\eta \longrightarrow \text{Step size of each correction}$$

  - Learning rate and number of trees should be considered together

XGBoost & LightGBM

- XGBoost Extreme Gradient Boosting
  - Optimized gradient boosting framework
  - Regularization to control model complexity
  - Efficient tree construction
  - Widely used for tabular data
- LightGBM Light Gradient Boosting Machine
  - Designed for speed and scalability
  - Histogram-based splitting
  - Leaf-wise tree growth
  - Efficient for large datasets
- Both are practical extensions of gradient-boosted decision trees



Boosting Methods

Method	Main idea
AdaBoost	Focus on misclassified samples
Gradient Boosting	Correct residuals / loss gradients
XGBoost	Regularized and optimized gradient boosting
LightGBM	Efficient and scalable gradient boosting

- Boosting = sequential model improvement

Next Lecture

- Lecture 7: Dimensionality Reduction & Clustering
  - Unsupervised Learning
  - Dimensionality Reduction & PCA
  - K-Means Clustering
  - Hierarchical Clustering
  - Clustering Evaluation & Visualization

Korea Aerospace University

Questions?

jiyoonshin@kau.ac.kr