

Lecture 4:

From Linear to Logistic Regression

Ho-min Park

homin.park@ghent.ac.kr

powersimmani@gmail.com

Lecture Contents

Part 1: Advanced Linear Regression

Part 2: Transition to Classification

Part 3: Completing Logistic Regression

Part 1/3:

Advanced Linear Regression

1. Review and Connection to Previous Lecture
2. Revisiting Linear Regression Assumptions
3. Polynomial Regression and Basis Expansion
4. Ridge Regression (L2 Regularization)
5. Lasso Regression (L1 Regularization)
6. Elastic Net
7. Feature Selection and Importance
8. Limitations of Linear Regression

Linear Regression: Foundation & Challenges

Linear Regression
Minimizes Sum of Squared Errors

$$\hat{y} = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \dots + \beta_n x_n$$

Normal Equation
Solution



Gradient Descent
Optimization

Linearity

Independence

Homoscedasticity

Overfitting

Multicollinearity

Noise

Goal: Extend Linear Regression Capabilities

Linear Regression: Key Assumptions

1 Linearity

Relationship between X and y is linear

2 Independence

Observations are independent of each other

3 Homoscedasticity

Constant variance of residuals

4 Normality

Residuals follow normal distribution

5 No Multicollinearity

Predictors are not highly correlated

? Violation Check

What happens when assumptions are violated?



Impact of Assumption Violations

Affects **prediction accuracy** and **interpretation** of model results

1 Linearity

✓ Linear

✗ Non-linear

2 Independence

✓ Independent

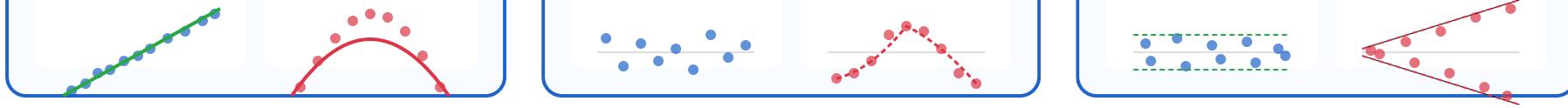
✗ Autocorrelated

3 Homoscedasticity

✓ Constant Variance

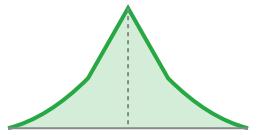
✗ Heteroscedasticity

Visual Guide to Assumptions

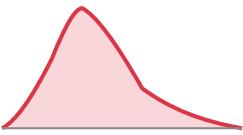


4 Normality of Residuals

✓ Normal Distribution



✗ Skewed Distribution



5 No Multicollinearity

✓ Low Correlation

	X ₁	X ₂	X ₃
X ₁	1.0	0.2	0.1
X ₂	0.2	1.0	0.3
X ₃	0.1	0.3	1.0

✗ High Correlation

	X ₁	X ₂	X ₃
X ₁	1.0	0.9	0.8
X ₂	0.9	1.0	0.9
X ₃	0.8	0.9	1.0

Polynomial Regression & Basis Expansion

Key Insight

Linear models can fit non-linear relationships

Transform Features:

$$x \rightarrow x, x^2, x^3, \dots$$

Example Equation:

$$y = \beta_0 + \beta_1 x + \beta_2 x^2 + \beta_3 x^3$$



Still "linear" in parameters (coefficients)



Trade-off: Model flexibility vs. overfitting risk



Choosing polynomial degree: validation approach



Other basis functions:

Logarithmic, Exponential, Trigonometric



Interactive Polynomial Regression Simulator

Explore polynomial regression with different degrees using Desmos!

[Launch Interactive Simulator →](#)

Ridge Regression (L2 Regularization)

PROBLEM

Large coefficients
lead to overfitting



SOLUTION

Add penalty term
for large coefficients

Cost Function:

$$RSS + \lambda \sum \beta_i^2$$

(L2 Penalty)



Coefficient Shrinkage

Shrinks coefficients toward zero (but not exactly zero)



Hyperparameter λ

Controls regularization strength



Effect of λ

Larger $\lambda \rightarrow$ more regularization, simpler model

✓ Key Benefit

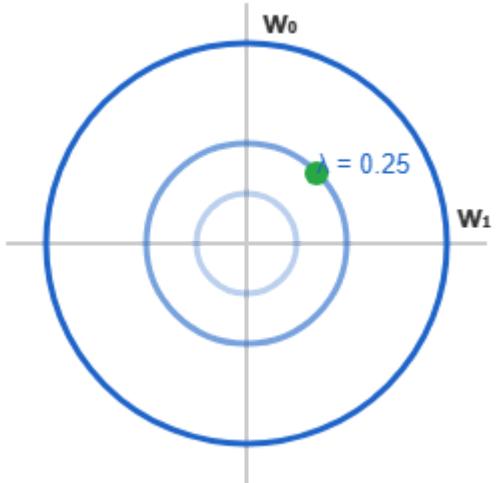
Helps with multicollinearity problem



2D Visual Understanding

L2 (Ridge) 

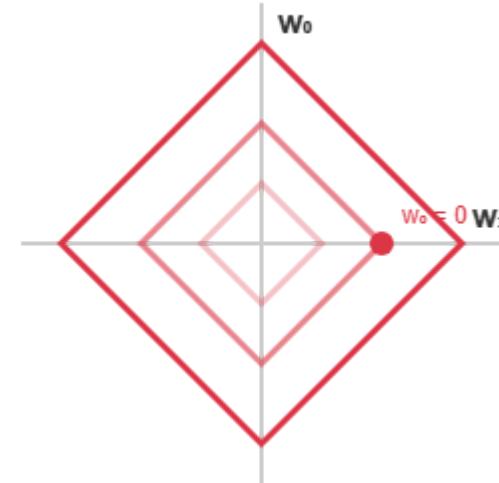
L1 (Lasso) 



Circle shape

$$w_0^2 + w_1^2 = \text{constant}$$

Gradually shrinks coefficients



Diamond shape

$$|w_0| + |w_1| = \text{constant}$$

Can make coefficients exactly zero



Interactive: Effect of λ on Coefficients

λ (Lambda) = **0.0**



💡 Move the slider to change λ value. As λ increases, coefficients shrink toward zero!

β_1 (original: 5.00)

β_2 (original: 3.00)

β_3 (original: 4.00)

5.00

3.00

4.00

Coefficient Value



Geometric Intuition

Ridge regression finds the optimal solution by balancing two objectives:

1 Minimize Prediction Error (RSS)

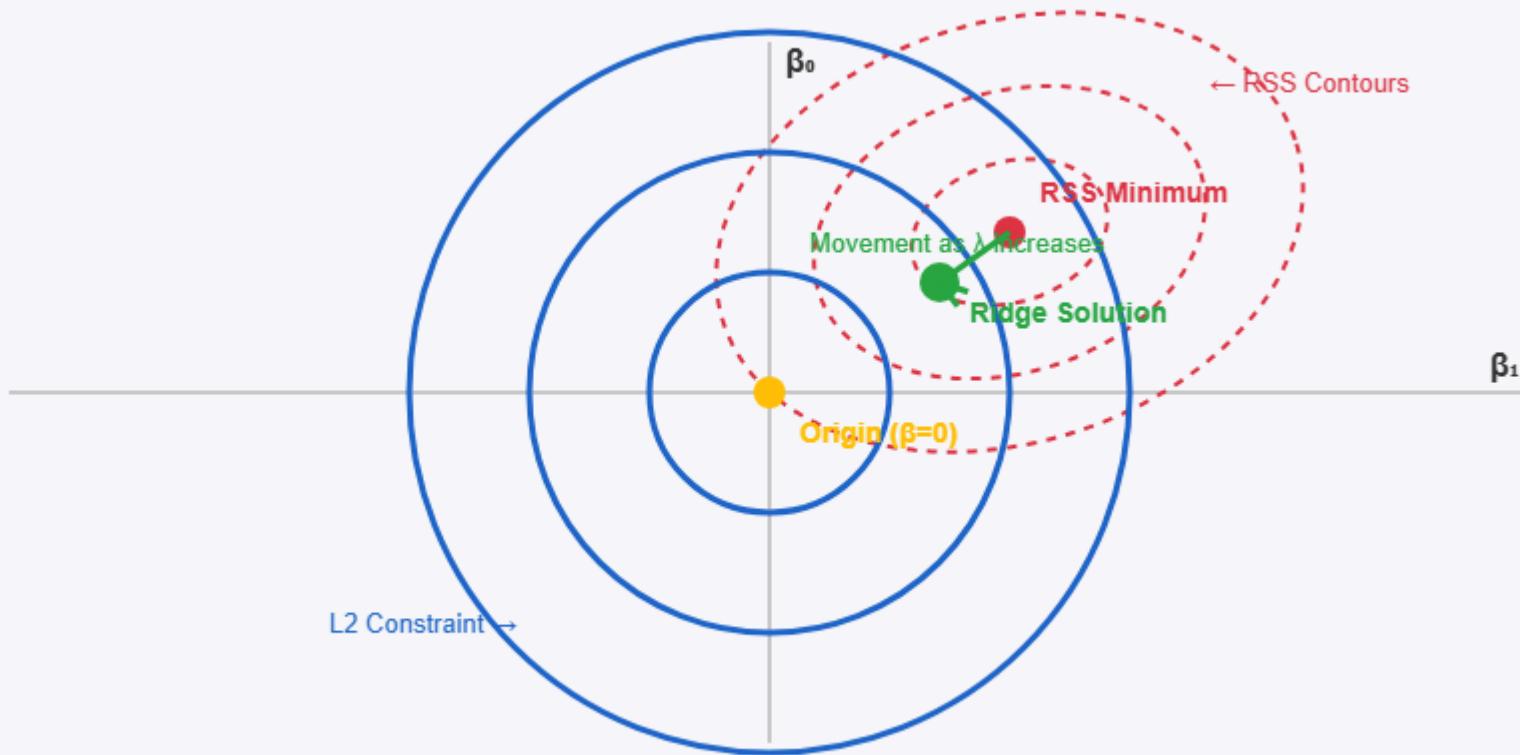
Want to fit the training data well

2 Minimize Coefficient Magnitude ($\lambda \sum \beta_i^2$)

Want to keep the model simple

$$\text{Loss} = \text{RSS} + \lambda \sum \beta_i^2$$

The solution moves from the RSS optimal point
toward the origin ($\beta=0$) as λ increases



Key Insight: L2 penalty creates a circular constraint. The optimal solution is where the RSS contour (ellipse) touches the L2 constraint circle. As λ increases, the circle gets smaller, pulling coefficients toward zero.



Why Ridge Helps with Multicollinearity

When features are highly correlated (multicollinear), ordinary linear regression can produce very large coefficients that cancel each other out.

⚠ Problem Example:

If $x_1 \approx x_2$:

$$y = 100 \cdot x_1 - 99 \cdot x_2$$

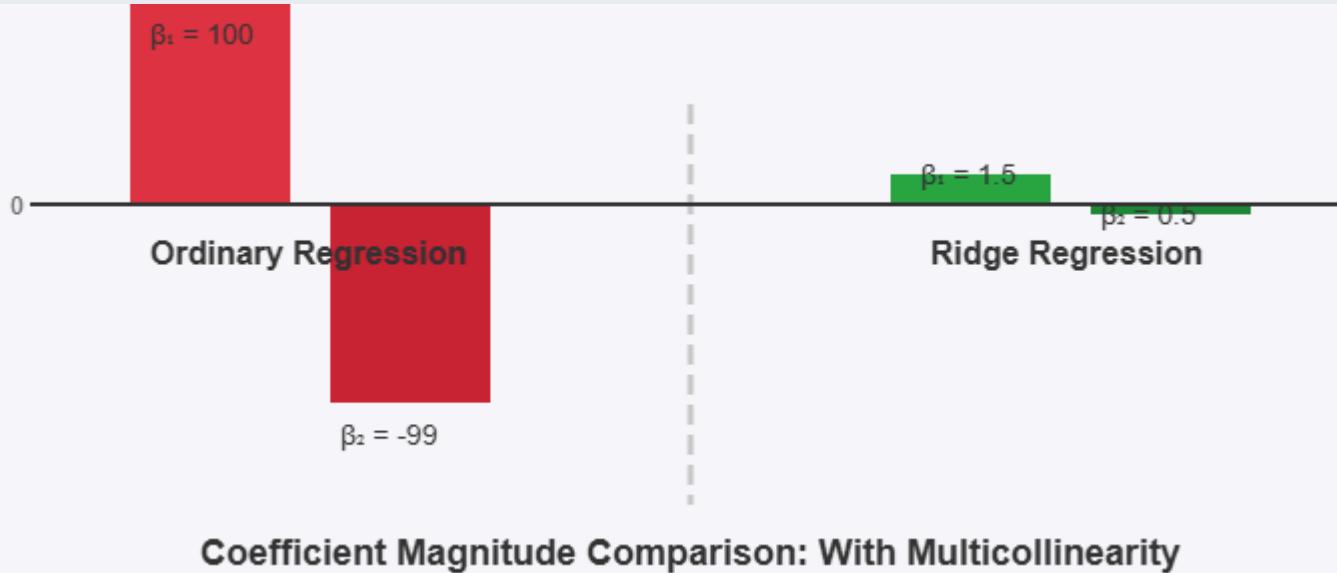
These large coefficients are unstable and sensitive to small data changes!

✓ Ridge Solution:

By penalizing large coefficients:

$$y = 1.5 \cdot x_1 + 0.5 \cdot x_2$$

The model becomes more stable and generalizes better!



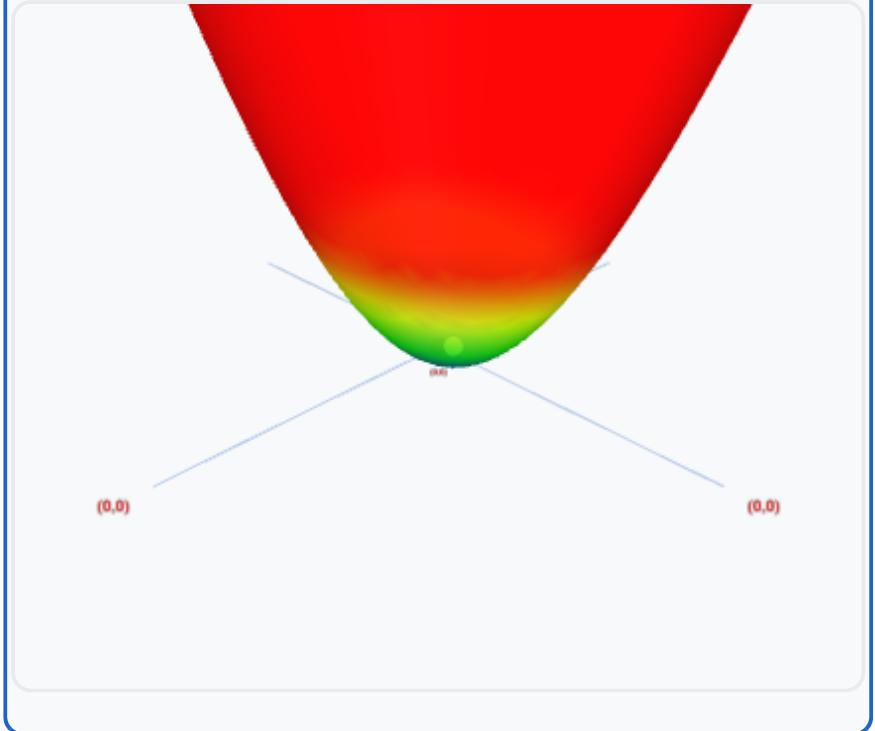
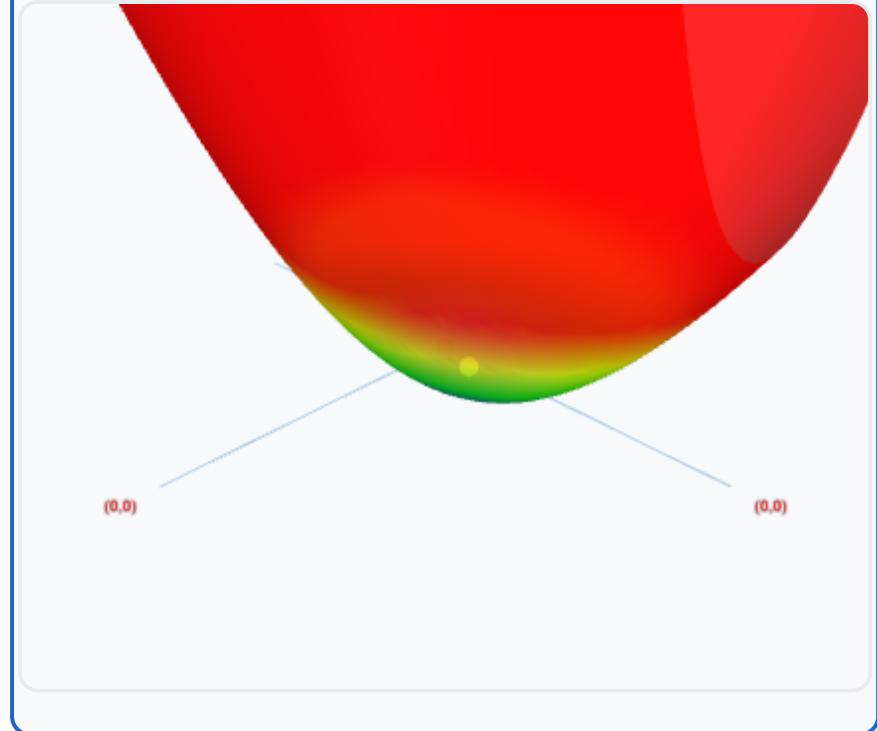
3D Loss Surface Visualization

Compare how the loss surface changes with and without L2 regularization.

The white sphere marks the optimal coefficient values.

Ordinary Least Squares (OLS)

Ridge Regression (L2)



λ (Lambda):

1.0

Rotation Speed:

1.0

 **Tip:** Increase λ to see how the optimal point (white sphere) moves closer to the origin (red sphere at center), representing smaller coefficient values and a simpler model.

Lasso Regression (L1 Regularization)

L1 Penalty:

$$RSS + \lambda \sum |\beta_i|$$



Sparse Solutions

Makes some coefficients exactly zero



Feature Selection

Automatically selects important features



Hyperparameter λ

Controls sparsity level



Interpretability

Creates simpler, more interpretable models

Ridge vs Lasso

Ridge (L2)

vs

Lasso (L1)

Ridge shrinks coefficients, but Lasso can make them **exactly zero** → Feature selection!

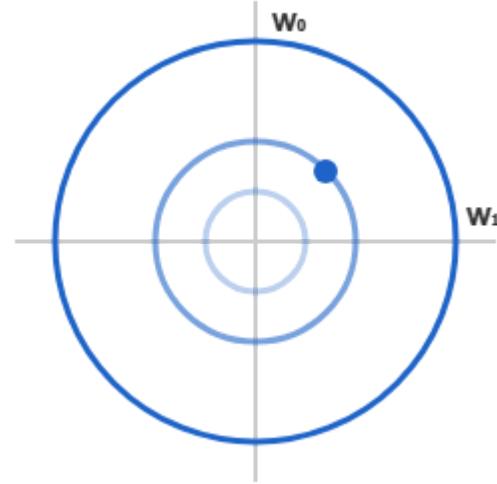


Use Cross-Validation to choose optimal λ



2D Visual Understanding

L2 (Ridge) ●

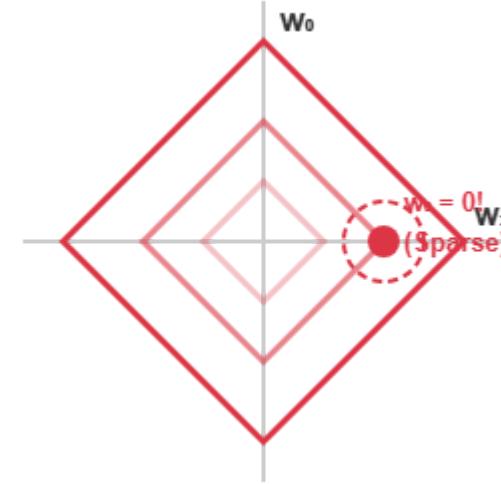


Circle shape

$$w_0^2 + w_1^2 = \text{constant}$$

Gradually shrinks coefficients

L1 (Lasso) ●



Diamond shape

$$|w_0| + |w_1| = \text{constant}$$

Can make coefficients exactly zero



Interactive: Effect of λ on Lasso Coefficients

λ (Lambda) = 0.0



💡 Move the slider to change λ value. Notice how Lasso makes coefficients exactly zero!

β_1 (original: 5.00)

5.00

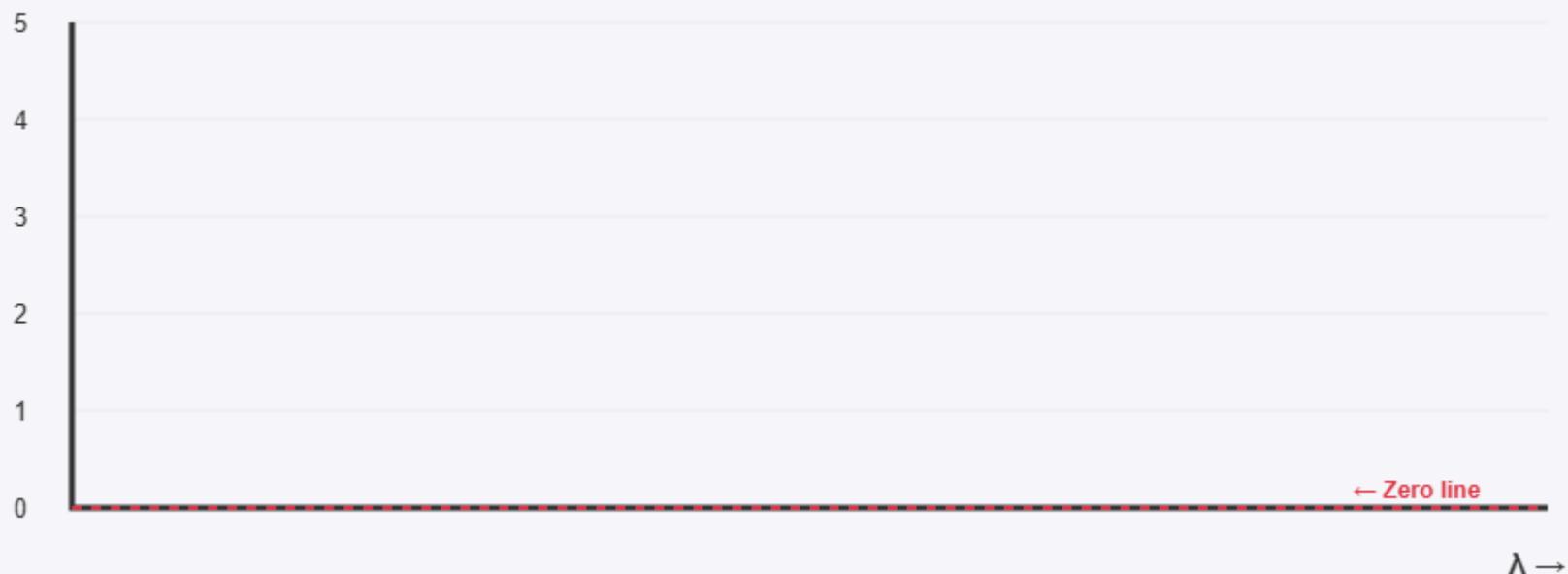
β_2 (original: 3.00)

3.00

β_3 (original: 4.00)

4.00

Coefficient Value



Geometric Intuition: Why Lasso Creates Sparsity

Lasso regression finds the optimal solution by balancing two objectives:

1 Minimize Prediction Error (RSS)

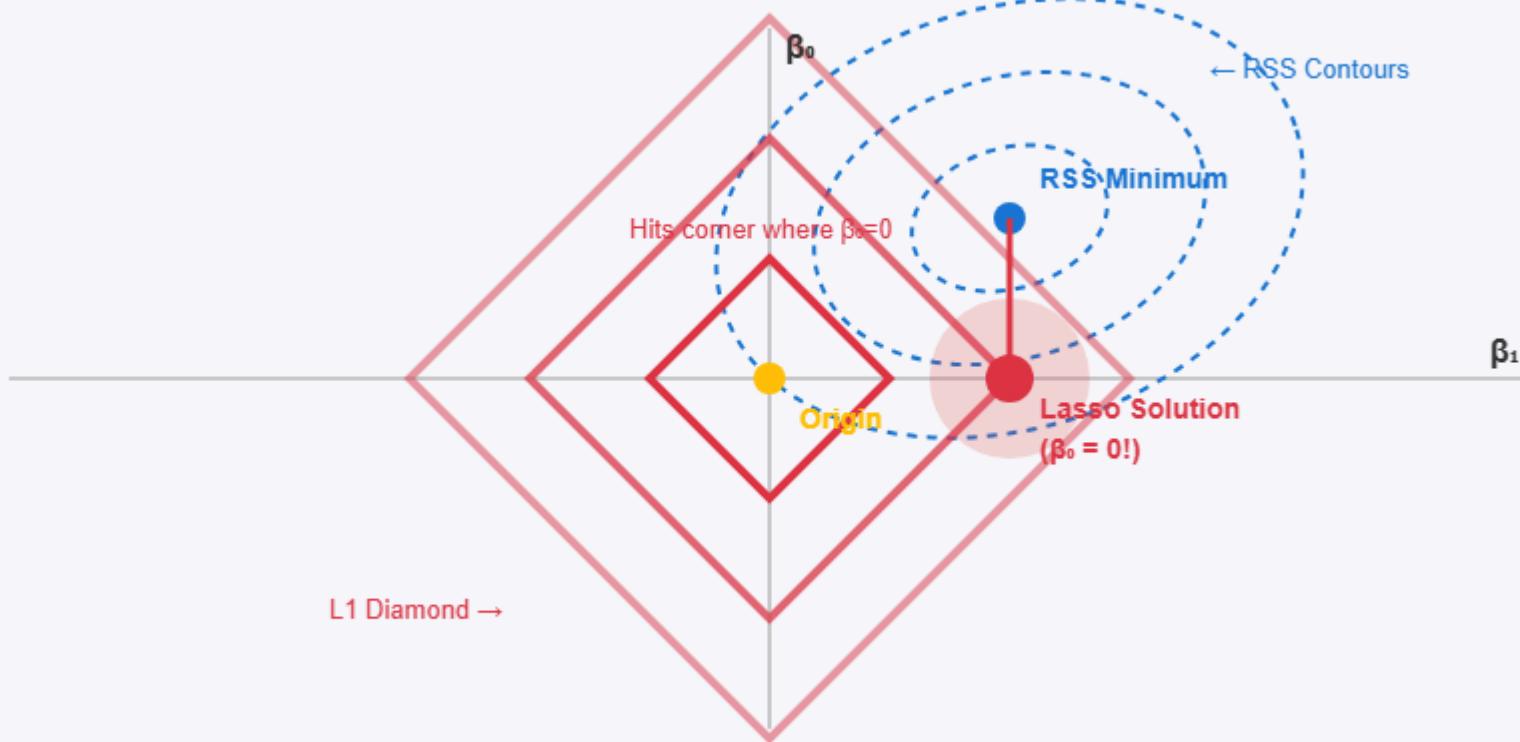
Want to fit the training data well

2 Minimize Coefficient Magnitude ($\lambda \sum |\beta_i|$)

Want to keep the model simple with sparse coefficients

$$\text{Loss} = \text{RSS} + \lambda \sum |\beta_i|$$

The diamond shape of L1 constraint
has corners where coefficients become exactly zero



 **Key Insight:** L1 penalty creates a diamond-shaped constraint. The optimal solution tends to hit the corners of the diamond, where one or more coefficients are exactly zero. This is why Lasso performs automatic feature selection!



Why Sparsity Matters: Feature Selection

When you have many features, not all of them are useful. Lasso automatically identifies and removes irrelevant features by setting their coefficients to zero.



Real-world Scenario:

Predicting house prices with 100 features:

Many features like "color of doorknob" are irrelevant

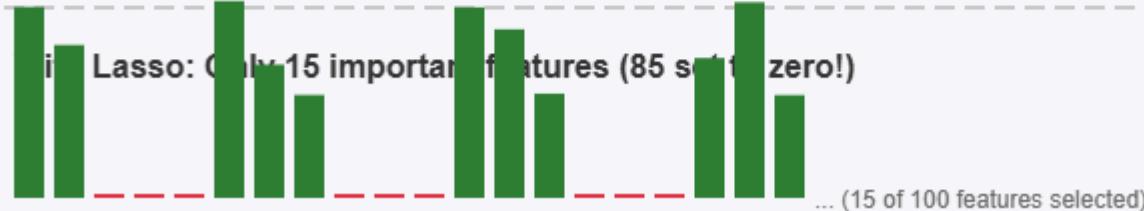
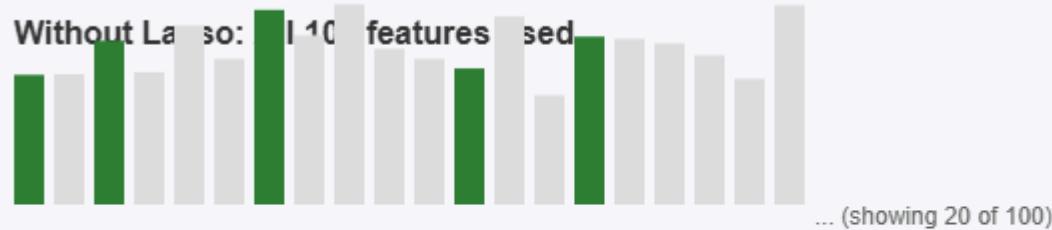
Lasso identifies that only 15 features truly matter and sets the rest to zero.



Benefits of Sparsity:

- 1. Interpretability:** Simpler models are easier to understand
- 2. Efficiency:** Fewer features mean faster predictions
- 3. Generalization:** Removes noise, improving performance on new data
- 4. Cost savings:** Don't need to collect/store irrelevant features

100 Features: Which ones matter?



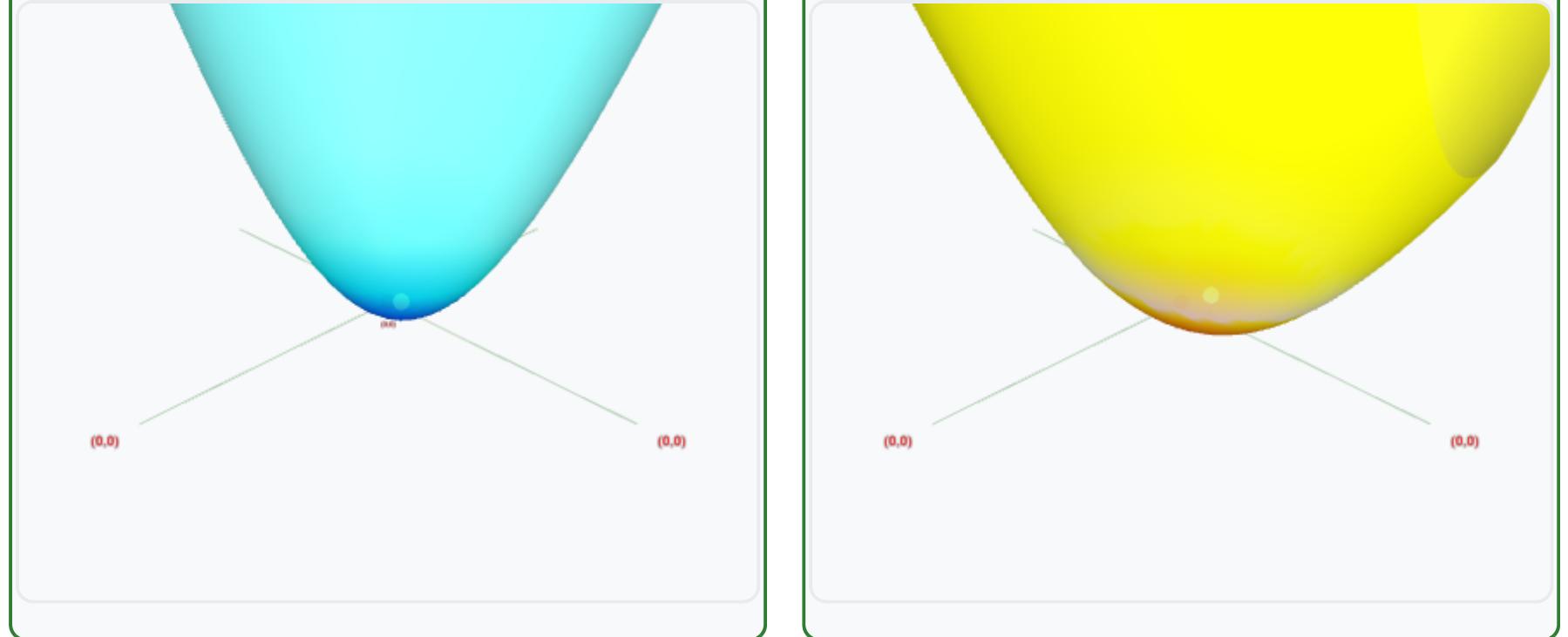
3D Loss Surface Visualization

Compare how the loss surface changes between Ridge (L2) and Lasso (L1) regularization.

Notice how Lasso's solution (white sphere) tends to align with axes, indicating zero coefficients.

Ridge Regression (L2)

Lasso Regression (L1)



λ (Lambda): 1.0

Rotation Speed: 1.0

Tip: Increase λ to see how Lasso's optimal point (white sphere) moves toward the axes, while Ridge's point moves smoothly toward the origin. This visualizes why Lasso creates sparse solutions!

Elastic Net: Best of Both Worlds

Ridge
(L2)

+

Lasso
(L1)

=

Elastic Net
Combined Power

Cost Function:

$$RSS + \lambda_1 \sum \beta_i^2 + \lambda_2 \sum |\beta_i|$$



Gets benefits of **both methods**



Feature selection like Lasso



Stability like Ridge



Useful when predictors are **highly correlated**

Hyperparameters: α (mixing) and λ (strength)

$\alpha = 0$
Ridge

$0 < \alpha < 1$
Elastic Net

$\alpha = 1$
Lasso

Feature Selection and Importance

Why Feature Selection Matters
Interpretability + Efficiency

Selection Methods

Filter

Correlation-based

Wrapper

RFE (Recursive Feature Elimination)

Embedded

Lasso regularization



Coefficient Magnitude

Indicates feature importance



Standardization

Standardize features before comparing coefficients



Beware

Multicollinearity affects importance interpretation



Permutation Importance

Alternative robust method

Balance: Model Simplicity Predictive Power

Example: Applying Feature Selection Methods

Filter Method

Step 1: Calculate Correlation

Compute correlation between each feature and target

Wrapper Method (RFE)

Step 1: Train Full Model

Fit model with all features

Embedded Method (Lasso)

Step 1: Set Regularization

Choose penalty parameter λ

```
corr_matrix = X.corrwith(y)
```

Step 2: Set Threshold

Select features with $|correlation| > 0.5$

```
threshold = 0.5
```

Step 3: Select Features

Keep features above threshold

```
selected = corr_matrix[abs(corr_matrix) > 0.5]
```

✓ Result

Fast, independent of model
Good for initial screening

```
model.fit(X_all, y)
```

Step 2: Rank Features

Identify least important feature

```
importance = model.coef_
```

Step 3: Remove & Repeat

Eliminate feature, retrain model

```
X_new = X.drop(worst_feature)  
model.fit(X_new, y)
```

✓ Result

Model-specific selection
More accurate but slower

```
model = Lasso(alpha=0.1)
```

Step 2: Train Model

L1 penalty shrinks coefficients

```
model.fit(X, y)
```

Step 3: Extract Features

Select features with non-zero coefficients

```
selected = X.columns[model.coef_ != 0]
```

✓ Result

Automatic selection
Built into training process

Limitations of Linear Regression

✗ Cannot handle non-linear decision boundaries naturally

✗ Assumes continuous output (not suitable for classification)

✗ Sensitive to outliers without robust variants

✗ Feature engineering required for complex patterns

✗ Interpretation difficult with many interaction terms

✗ Limited for categorical outcomes

Why We Need Classification Algorithms



Critical Question:
What if output is categorical (yes/no)?

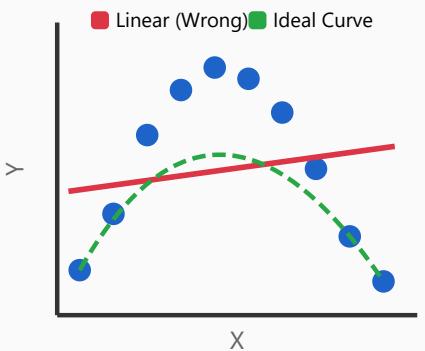


Visual Demonstrations of Limitations

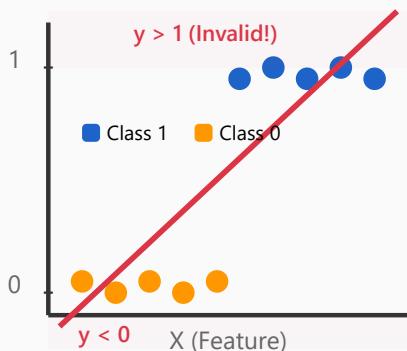
Non-linear Patterns

Classification Problem

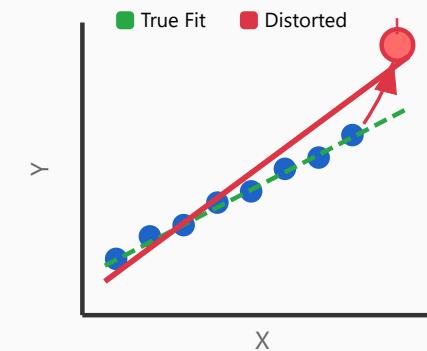
Outlier Sensitivity



Linear regression forces a straight line through curved data, causing systematic errors

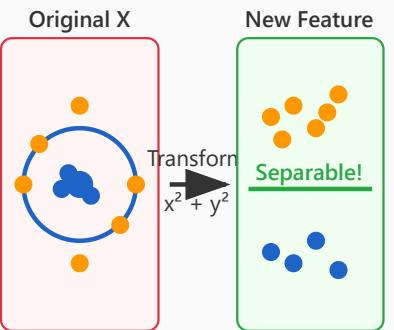


Linear regression outputs can exceed [0,1] range, making probability interpretation impossible



A single outlier can dramatically shift the regression line, distorting all predictions

Complex Patterns Need Engineering



Non-linear patterns require manual feature transformation (like x^2 terms) to become linearly separable

Interpretation Complexity

Simple Model

$$y = \beta_0 + \beta_1 x_1 + \beta_2 x_2$$

✓ Easy to interpret

Terms
3
→ 20+

↓
Add interactions

Complex Model

$$\begin{aligned} &: \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \beta_3 x_1^2 + \beta_4 x_2^2 + \beta_5 x_1 x_2 + \\ &\times \text{Hard to explain coefficient meanings} \end{aligned}$$

Adding polynomial and interaction terms exponentially increases model complexity and reduces interpretability

Categorical Outcomes



?

Linear Output: $y = 1.7$
What does 1.7 mean!?

What we need:
 $P(\text{Cat})=0.1, P(\text{Dog})=0.8, P(\text{Bird})=0.1$

Linear regression cannot naturally output probabilities for multiple categories

Part 2/3:

Transition to Classification

- 9. Regression vs Classification Problems
- 10. Linear Classifier Concepts
- 11. Perceptron Algorithm
- 12. Decision Boundaries and Linear Separability
- 13. Why Linear Regression Fails for Classification
- 14. Odds and Log Odds
- 15. Introduction to Sigmoid Function
- 16. Properties of Logistic Function

Regression vs Classification Problems

Regression

Predict continuous values

Examples:

Price, Temperature, Age

VS

Classification

Predict discrete categories

Examples:

Spam/Ham, Disease/Healthy

Key Difference:

Output Space: \mathbb{R} (continuous) vs. Finite Set (discrete)

Regression Output

$$y \in \mathbb{R}$$

Scalar/Vector: $\hat{y} = 25.7^{\circ}\text{C}$

Multi-output: $\hat{y} = [150.2, 175.8, 163.4] \text{ kg}$

Classification Output

$$y \in \{0, 1, \dots, K-1\}$$

Binary: $\hat{y} = 1$ (Spam), $\hat{y} = 0$ (Ham)

Multi-class: $\hat{y} = 2$ (Cat), or $\hat{y} = [0, 0, 1, 0]$



Why linear regression fails:

Predictions outside [0,1] range



Need for probability:

$P(y=1|x)$ interpretation

Today's Focus:

Binary Classification (Two Classes)

Linear Classifier Concepts

Goal:

Find a line (hyperplane) that separates two classes

Decision Function:

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

Classification Rule:

if $f(\mathbf{x}) > 0 \rightarrow$ Class 1

else \rightarrow Class 0



Geometric View

Distance from decision boundary



Weight Vector w

Perpendicular to decision boundary



Bias Term b

Shifts boundary position



Multiple Features

Hyperplane in high-dimensional space

Perceptron Algorithm

Simplest Linear Classifier

Frank Rosenblatt

1957

Update Rule:

$$\mathbf{w} \leftarrow \mathbf{w} + \eta(y - \hat{y})\mathbf{x}$$

Activation Function:

Step Function

(0 or 1)



Iteratively adjusts weights for misclassified points



Converges if data is linearly separable



Limitation: No convergence for non-separable data



Historical Significance: Foundation of neural networks

Decision Boundaries and Linear Separability

Linearly Separable

Classes can be separated by straight line/plane

VS

Non-Linearly Separable

Requires curved boundary



XOR Problem: Classic example of non-linear separability

Real-world: Most datasets are not perfectly separable

Solutions:

Feature Transformation

Kernel Methods

Non-Linear Models



Linear Classifiers

Fast, interpretable, good baseline



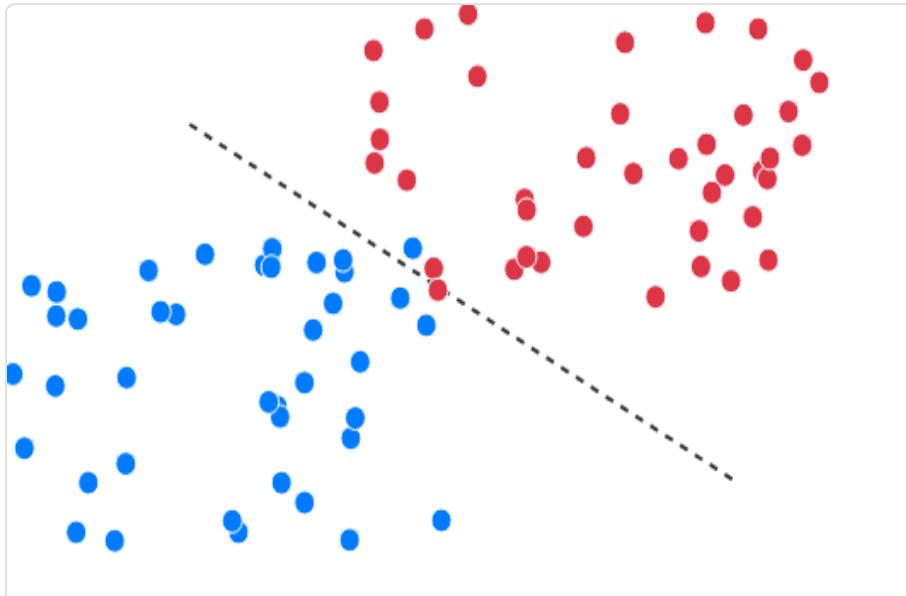
When to Use

High-dimensional data, large datasets



2D Decision Boundary Visualization

✓ Linearly Separable (2D)



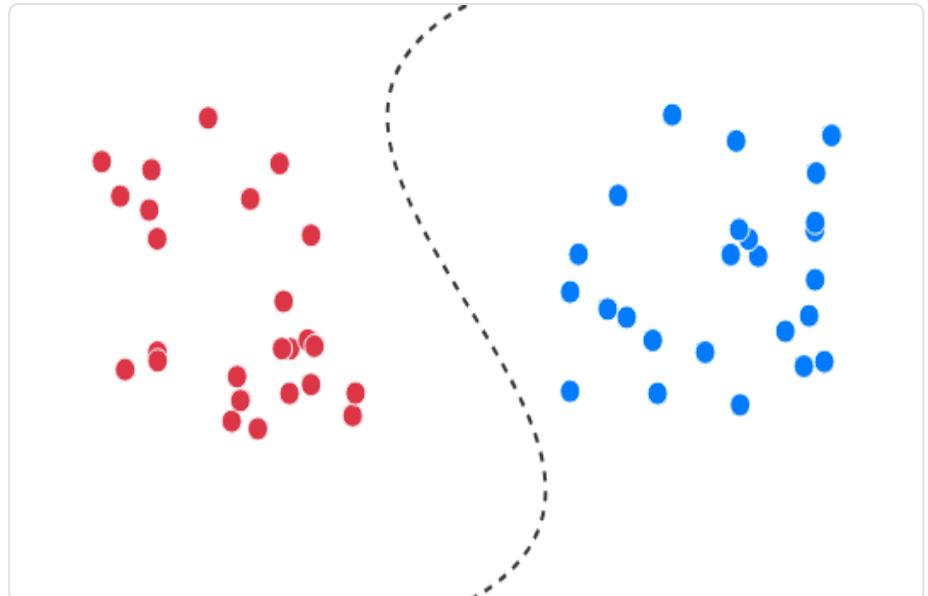
● Class A ● Class B

Generate Data

Clear

A single straight line can perfectly separate the two classes

X Non-Linearly Separable (XOR)



● Class A ● Class B

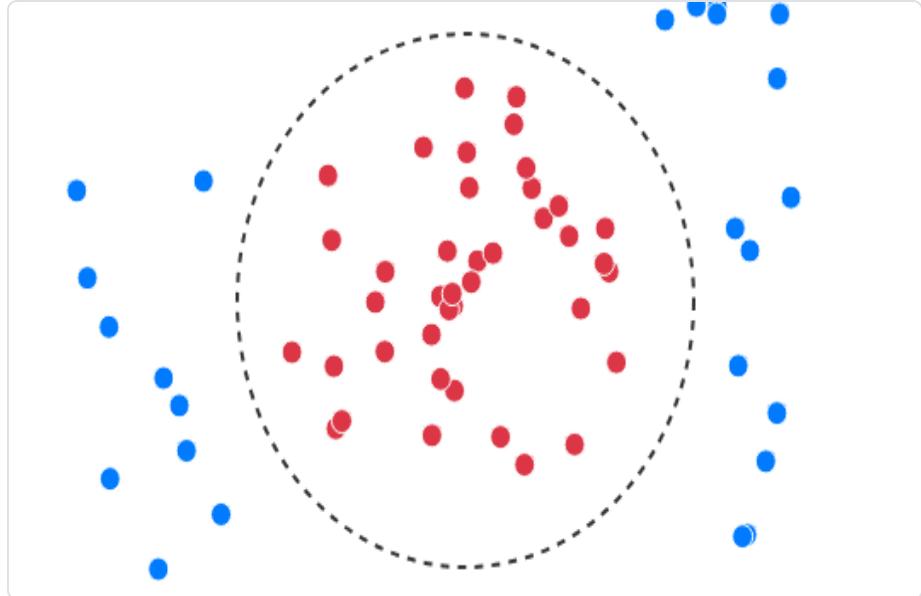
Generate XOR

Clear

XOR pattern requires a curved (non-linear) boundary

X Non-Linearly Separable (Circles)

X Non-Linearly Separable (Moons)

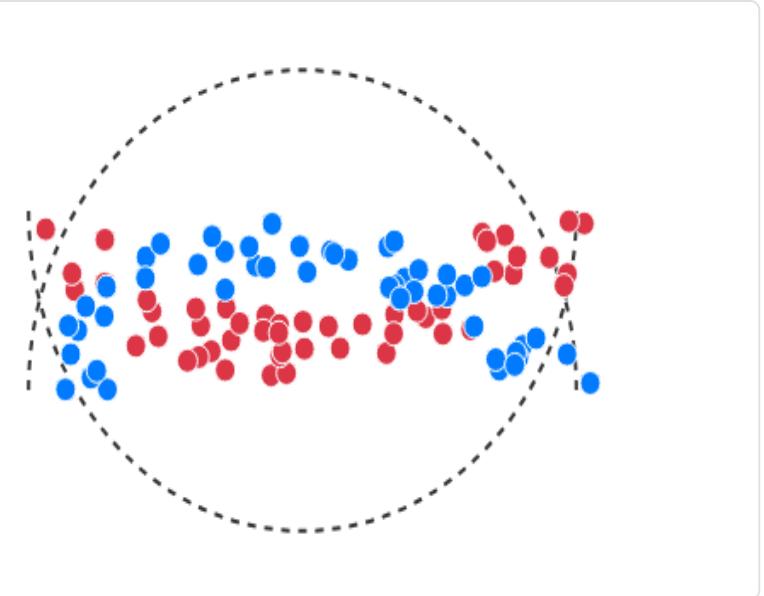


● Inner Class ● Outer Class

Generate Circles

Clear

Concentric circles require a circular boundary



● Class A ● Class B

Generate Moons

Clear

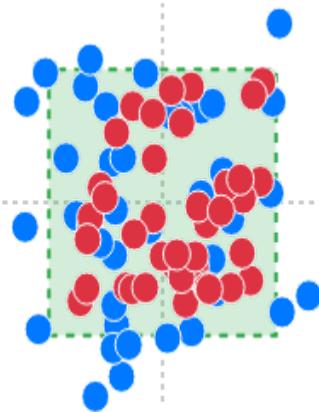
Interleaved crescent shapes require a complex boundary



3D Decision Boundary Visualization

✓ Linearly Separable (3D)

X Non-Linearly Separable (3D Spheres)



● Class A ● Class B

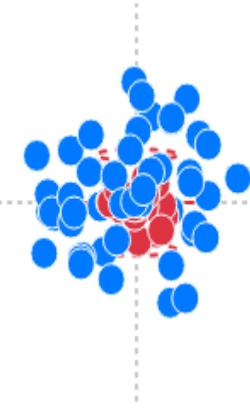
Generate Data

Rotate View

Stop

Clear

A single plane can separate classes in 3D space



● Inner Class ● Outer Class

Generate Spheres

Rotate View

Stop

Clear

Concentric spheres require a spherical boundary surface

Why Linear Regression Fails for Classification

- 1 Predictions can be < 0 or > 1
- 2 Treats classes as ordered numerical values
- 3 Sensitive to outliers in feature space
- 4 Squared loss inappropriate for binary outcomes

Example:

Predict probability of disease (should be in $[0,1]$)

Linear Regression Output:

Any real number

$\in \mathbb{R}$

→
NEED

Required Output:

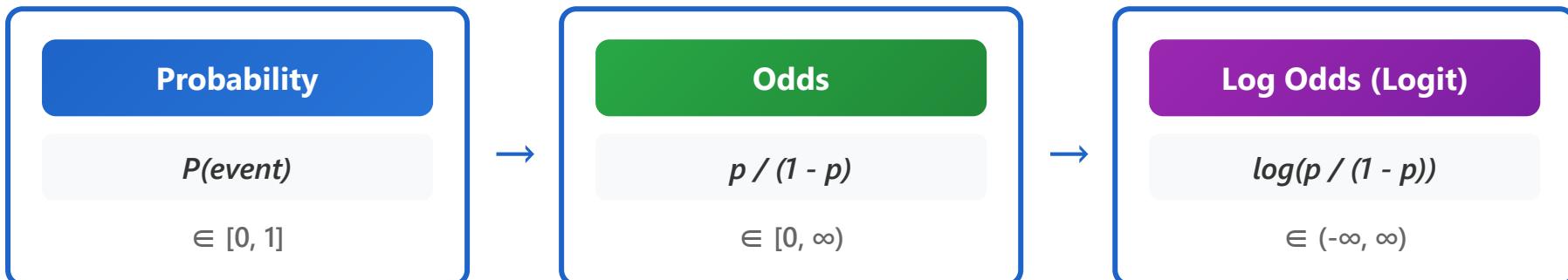
Bounded $[0,1]$

Probability

Solution Requirement:

Output bounded to $[0,1]$ representing probability

Odds and Log Odds



Key Insight:

Log odds can take any real value!

Linear model for log odds instead of probability

Example:

$$P = 0.8 \rightarrow \text{Odds} = 4 \rightarrow \text{Log odds} = 1.39$$

Bridge to Logistic Regression Formulation

Introduction to Sigmoid Function

Definition:

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

Output:

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



Maps any real number to (0,1) range



Perfect for probability interpretation

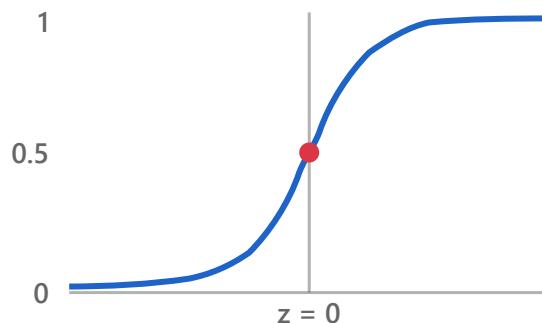


S-shaped curve: Smooth transition between 0 and 1



Solves the "prediction outside [0,1]" problem

S-Shaped Curve



Input $z=0 \rightarrow$ Output $\sigma=0.5$ (decision threshold)

Properties of Logistic Function

Symmetry

$$\sigma(z) + \sigma(-z) = 1$$

Monotonic

Always increasing, never decreases

Smooth

Differentiable everywhere (no jumps)

Bounded

Output always in (0,1), never exactly 0 or 1

Interpretable

Steepness indicates confidence

Derivative

Simple form for optimization

Symmetry Property:

$$\sigma(z) + \sigma(-z) = 1$$

Derivative:

$$\sigma'(z) = \sigma(z)(1 - \sigma(z))$$

(useful for gradient descent)

Comparison to Step Function

Logistic (Smooth)



vs

Step (Hard Threshold)



Smooth transition vs. Abrupt change

Part 3/3:

Completing Logistic Regression

- 17.** Defining the Logistic Regression Model
- 18.** Maximum Likelihood Estimation (MLE)
- 19.** Binary Cross-Entropy Loss
- 20.** Applying Gradient Descent
- 21.** Multiclass - One-vs-Rest Strategy
- 22.** Softmax Regression
- 23.** Categorical Cross-Entropy
- 24.** Regularized Logistic Regression
- 25.** Real-World Cases and Implementation

Defining the Logistic Regression Model

Complete Formulation:

$$\begin{aligned} P(y=1|x) &= \sigma(w^T x + b) \\ &= 1 / (1 + e^{-(w^T x + b)}) \end{aligned}$$

Step 1
 $w^T x + b$
Linear combination

Step 2
 $\sigma(\dots)$
Sigmoid function

Output
 $P \in [0,1]$
Probability

Decision Rule:
Predict Class 1 if
 $P(y=1|x) \geq 0.5$

💡 Threshold can be adjusted based on application needs

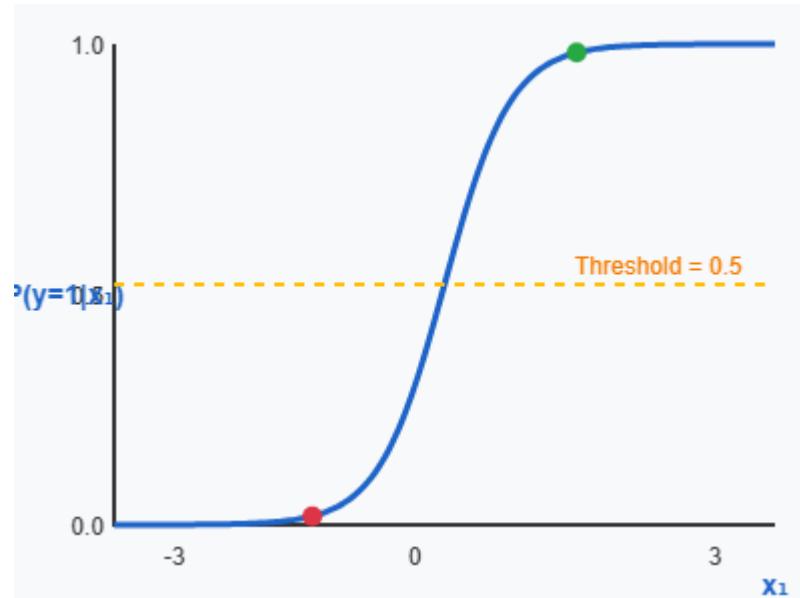
w Weight Vector w
Model parameters to learn

b Bias b
Intercept term to learn

Training Goal: Find optimal w and b from data

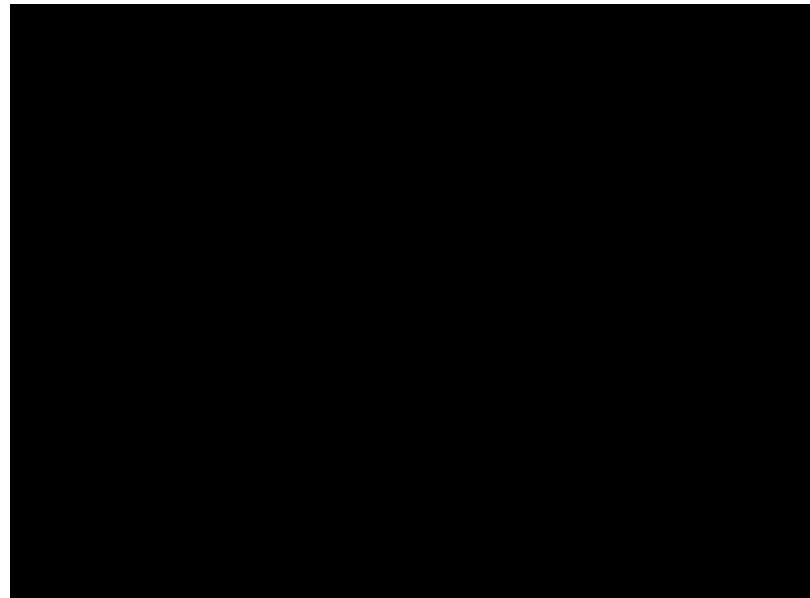
Input-Output Visualization

2D: Single Feature (x_1)



Sigmoid curve: $P(y=1|x_1) = \sigma(w_1x_1 + b)$

3D: Two Features (x_1, x_2)



Decision surface: $P(y=1|x_1, x_2) = \sigma(w_1x_1 + w_2x_2 + b)$



Drag to rotate, scroll to zoom

Maximum Likelihood Estimation (MLE)

📌 Logistic Regression Model (from previous slide):

$$p = \sigma(\mathbf{w} \cdot \mathbf{x} + b) = 1 / (1 + e^{-(\mathbf{w} \cdot \mathbf{x} + b)})$$

🎯 Parameters to Optimize:

w, b

How do we find the best values?

MLE Goal: Find w, b that maximize the probability of observing our training data

1 Likelihood Function:

$$L(\mathbf{w}, \mathbf{b}) = \prod P(y_i | \mathbf{x}_i; \mathbf{w}, b)$$

2

Log-Likelihood (easier to optimize): $\ell(\mathbf{w}, \mathbf{b}) = \sum \log P(y_i | \mathbf{x}_i; \mathbf{w}, b)$

3

For Binary (where $p = \sigma(\mathbf{w} \cdot \mathbf{x} + b)$): $\ell = \sum [y_i \log(p) + (1-y_i)\log(1-p)]$

Maximizing log-likelihood = Minimizing negative log-likelihood (Loss)

?

Why MLE?

Principled probabilistic approach to find w, b



Next:

This leads to Binary Cross-Entropy Loss

Binary Cross-Entropy Loss

Loss Function:

$$L = -[y \log(\hat{y}) + (1-y)\log(1-\hat{y})]$$



Penalizes wrong predictions heavily



Convex function: Guaranteed global minimum



Asymmetric penalty: Confident wrong predictions costly

If $y = 1$

Loss:

$$-\log(\hat{y})$$

Minimized when $\hat{y} \rightarrow 1$

If $y = 0$

Loss:

$$-\log(1-\hat{y})$$

Minimized when $\hat{y} \rightarrow 0$



Asymmetric penalty structure



Confident wrong predictions are costly



Interactive 3D Loss Surface

Drag to rotate • Scroll to zoom

 **Reset View**

 **Toggle Wireframe**

 **Auto Rotate**

 Interact with the 3D surface: Click and drag to rotate, scroll to zoom in/out

3D Surface Visualization: This interactive plot shows the Binary Cross-Entropy loss landscape with

X-axis: y (True Label, 0 or 1)

Z-axis: \hat{y} (Predicted Value, 0-1)

Y-axis: Loss (BCE)

The surface is **not convex** in the traditional sense when visualized this way. Note the two "valleys" at (0,0) and (1,1) where predictions match true labels. Colors: **Blue (low loss)** → **Yellow** → **Red (high loss)**

Applying Gradient Descent

No closed-form solution like linear regression

Solution:

Iterative Optimization: Gradient Descent

Update Rule:

$$\mathbf{w} \leftarrow \mathbf{w} - \eta \nabla L(\mathbf{w})$$

Gradient:

$$\partial L / \partial \mathbf{w} = (\hat{\mathbf{y}} - \mathbf{y}) \mathbf{x}$$

(similar form to linear regression!)

Learning Rate η :

Controls step size (typically 0.001-0.1)

Variants:

Batch Gradient Descent

Mini-batch Gradient Descent

Stochastic Gradient Descent

Convergence

Monitor on validation set:

Loss

Accuracy

Iterative Process



1

Initialize Parameters

Movement in Loss Space

Start with initial weights (often random or zeros)

$$w_0 = 0.5$$

2

Compute Prediction

Forward pass through the model

$$\hat{y} = \sigma(w_0 x) = \sigma(0.5 \times 2.0) = 0.73$$

3

Calculate Loss

Measure prediction error

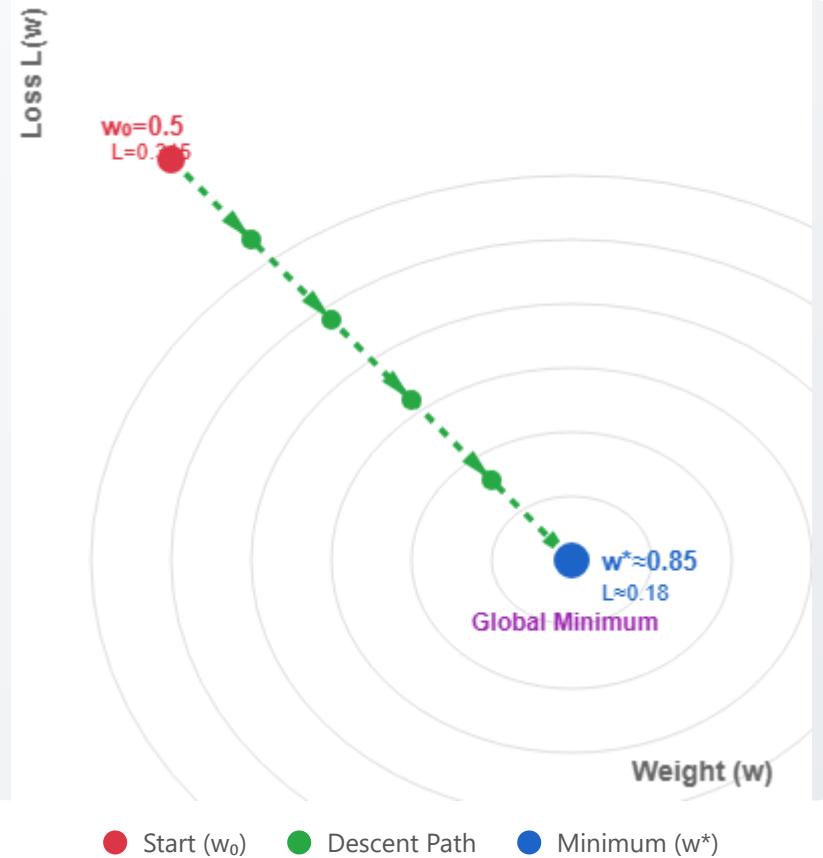
$$L = -[y \log(\hat{y}) + (1-y) \log(1-\hat{y})]$$
$$L = 0.315$$

4

Compute Gradient

Calculate direction of steepest ascent

$$\partial L / \partial w = (\hat{y} - y)x = (0.73 - 1) \times 2.0$$
$$\nabla L = -0.54$$



5

Update Parameters

Move in opposite direction of gradient

$$w_1 = w_0 - \eta \nabla L$$

$$w_1 = 0.5 - 0.1 \times (-0.54) = 0.554$$

6

Repeat

Continue until convergence

Iterate steps 2-5 until $\Delta L < \epsilon$

Concrete Example

Training Data:

$x = 2.0, y = 1$ (positive class)

Initial Weight:

$w_0 = 0.5$

Learning Rate:

$\eta = 0.1$

After Update:

$w_1 = 0.554$ (\downarrow Loss: $0.315 \rightarrow 0.289$)

Multiclass - One-vs-Rest Strategy

So far: Binary classification
(K=2)
Logistic Regression with sigmoid

Now: Multiclass (K > 2)
e.g., Cat / Dog / Bird / ...

Approach ①
One-vs-Rest (OvR)

K Train **K binary classifiers**: Class k vs. all others

σ Each classifier: Independent logistic regression (sigmoid)

✓ **Advantages**

- Simple to implement
- Interpretable
- Parallelizable

Architecture (K=5 example)

Class 1
vs Rest

Class 2
vs Rest

Class 3
vs Rest

Class 4
vs Rest

Class 5
vs Rest



Choose class with highest probability

✗ **Disadvantages**

- ⚠ Probabilities don't sum to 1!
Each classifier outputs independently
- Imbalanced training data
- Ambiguous regions possible



Want probabilities that sum to 1? → **Softmax Regression (Approach ②)**

Softmax Regression

Multinomial Logistic Regression (Native Multiclass)

Softmax Function:

$$P(y=k|x) = e^{w_k^T x} / \sum_j e^{w_j^T x}$$



Outputs: **K probabilities** that sum to 1



Generalizes sigmoid to multiple classes



Each class has its own weight vector w_k



More elegant than one-vs-rest for multiclass



Widely used: Neural network output layer

Weight Vectors per Class

w_1 for Class 1

w_2 for Class 2

w_3 for Class 3

... ...

w_K for Class K

Softmax → $\sum P = 1$

vs. Sigmoid

Softmax Regression - Calculation Example

1
2
3
4

3-Class Classification Example

1 Given Input & Weights

입력 벡터: $x = [2, 3]$

Weight vectors:

$$w_1 = [0.5, 0.3] \rightarrow \text{Class 1 (Cat)}$$

$$w_2 = [0.8, 0.4] \rightarrow \text{Class 2 (Dog)}$$

$$w_3 = [0.2, 0.9] \rightarrow \text{Class 3 (Bird)}$$

2 Calculate Logits ($z = w^T x$)

$$z_1 = w_1^T x = 0.5 \times 2 + 0.3 \times 3 = 1.0 + 0.9 = 1.9$$

$$z_2 = w_2^T x = 0.8 \times 2 + 0.4 \times 3 = 1.6 + 1.2 = 2.8$$

$$z_3 = w_3^T x = 0.2 \times 2 + 0.9 \times 3 = 0.4 + 2.7 = 3.1$$

3 Apply Exponential Function

$$e^{z_1} = e^{1.9} \approx 6.686$$

$$e^{z_2} = e^{2.8} \approx 16.445$$

$$e^{z_3} = e^{3.1} \approx 22.198$$

Sum = 6.686 + 16.445 + 22.198 = **45.329**

4 Calculate Softmax Probabilities

$$P(y=1|x) = e^{z_1} / \text{Sum} = 6.686 / 45.329 \approx \textbf{0.147 (14.7\%)}$$

$$P(y=2|x) = e^{z_2} / \text{Sum} = 16.445 / 45.329 \approx \textbf{0.363 (36.3\%)}$$

$$P(y=3|x) = e^{z_3} / \text{Sum} = 22.198 / 45.329 \approx \textbf{0.490 (49.0\%)}$$

Final Result

Prediction: Class 3 (Bird) with 49.0% probability

Note: All probabilities sum to 1.0 ($0.147 + 0.363 + 0.490 = 1.000$)

Categorical Cross-Entropy

Loss Function for Multiclass Classification

Formula:

$$L = -\sum_k y_k \log(\hat{y}_k)$$

One-hot Encoding:

$$\mathbf{y} = [0, 0, 1, 0, \dots]$$

for true class

Combined with Softmax:

$$\text{Softmax} + \text{CCE}$$

Differentiable end-to-end



Generalizes binary cross-entropy to **K classes**



Penalizes deviation from true class distribution

Gradient:

$$\hat{y}_k - y_k$$

(elegant and simple!)

Binary → 2 classes

Categorical → K classes

Regularized Logistic Regression

Prevent Overfitting with Penalty Terms

L2 (Ridge)

$$\text{Loss} + \lambda \sum w_i^2$$

Smooth coefficient shrinkage

L1 (Lasso)

$$\text{Loss} + \lambda \sum |w_i|$$

Feature selection

Elastic Net

$$L1 + L2$$

Combines L1 and L2 penalties



Hyperparameter λ : Controls regularization strength



Especially important with **high-dimensional data**

Choosing λ :

Cross-validation to choose optimal λ

Real-World Cases and Implementation

Applications:

Spam Detection Medical Diagnosis Credit Scoring

Scikit-learn Implementation:

```
LogisticRegression(  
    penalty='l2',  
    C=1.0  
)
```

Key Hyperparameters:

- Regularization (C)
- Solver
- max_iter

Evaluation Metrics:

Accuracy Precision

Recall F1-score

ROC-AUC

Important Considerations:

 **Class imbalance:** Use class_weight='balanced'

 **Feature scaling:** Standardize for better convergence

Best Practices:

Cross-validation Calibration Threshold tuning

Thank you

Ho-min Park

homin.park@ghent.ac.kr

powersimmani@gmail.com