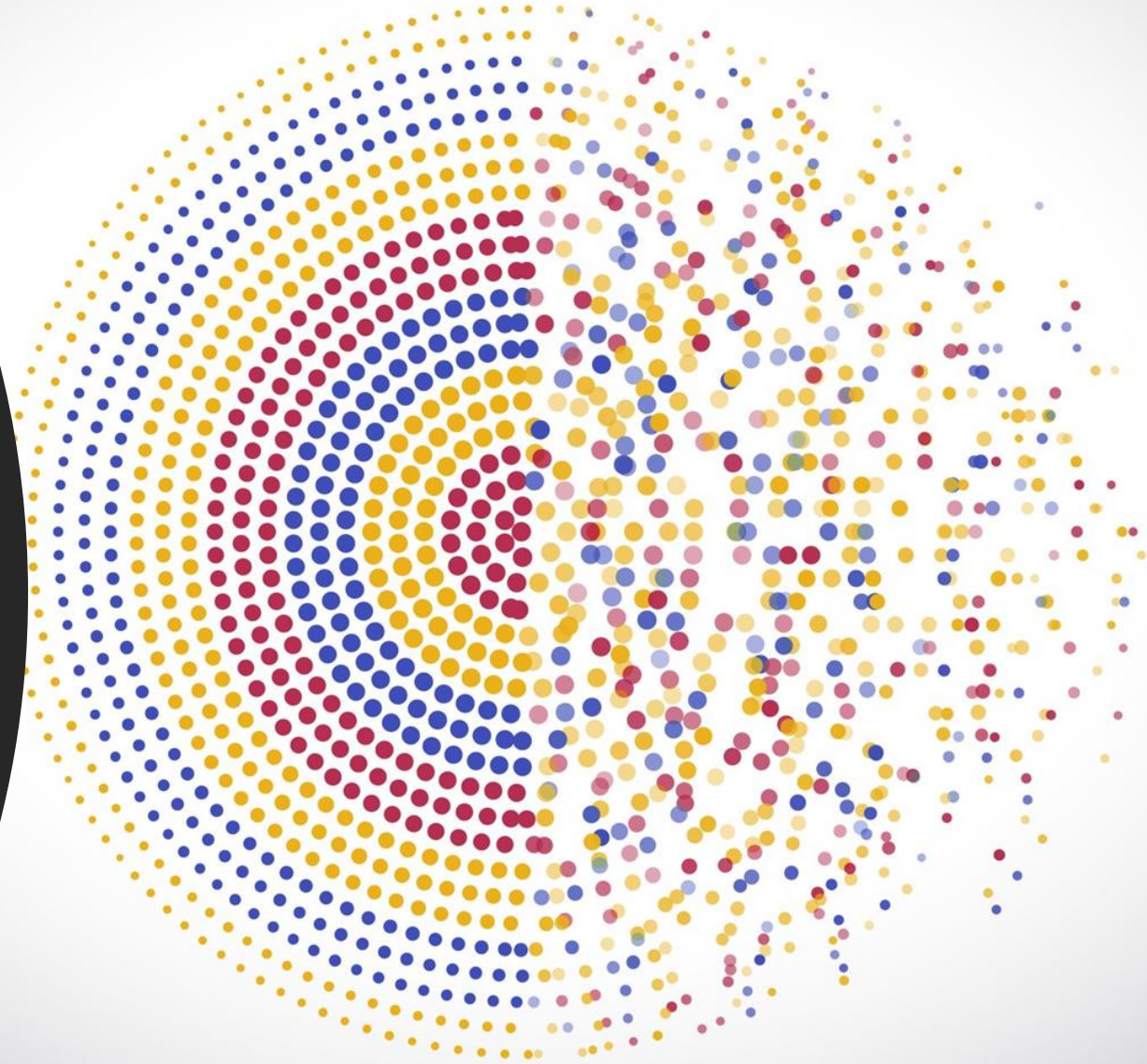
Supervised Learning

Dr. Mohamed AlHajri



Content

- Supervised Learning
- Linear Classification
    - Perceptron Algorithm
    - Linear Support Vector Machine (Linear SVM)
- Linear Regression
    - Least Square
    - Weighted Least Square
    - Ridge Regression
- Non-Linear Classification (Kernel)
    - Kernel Perceptron Algorithm
    - Kernel Support Vector Machine (Linear SVM)

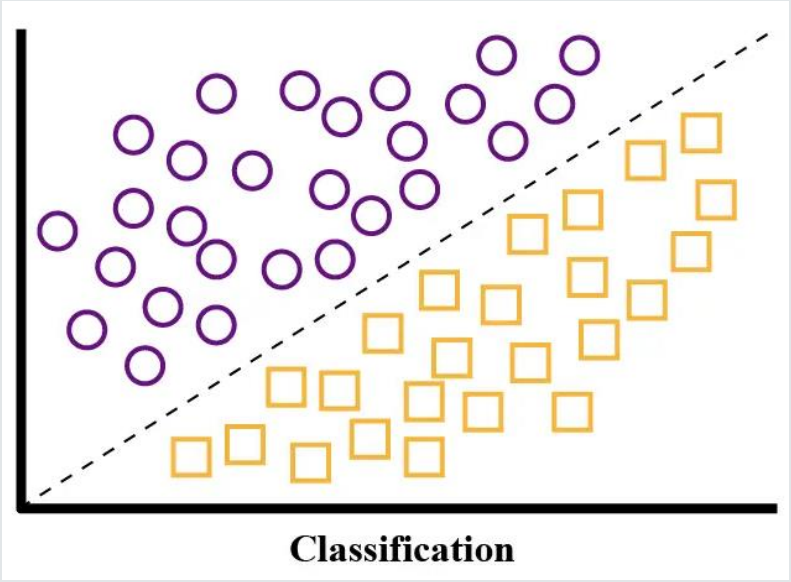
Content

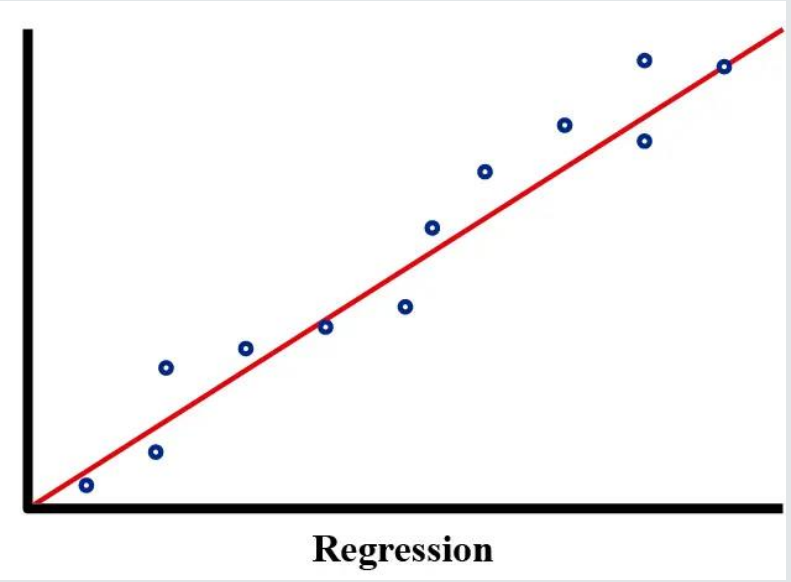
- Non-Linear Regression
    - Non-Linear Least Square
- K-nearest neighbor
- Decision Trees
- Feedforward Neural Network
- Convolutional Neural Network
- Recurrent Neural Network

Lecture 6 (Linear methods)

Supervised Learning

- Supervised learning involves learning a mapping function $f: X \rightarrow Y$ from input features $X \in \mathbb{R}^n$ to output labels Y, where:
  - Classification: Y is discrete (e.g., $Y \in \{1,2,\cdots,k\}$).
  - Regression: Y is continuous (e.g., $Y \in \mathbb{R}$).





https://panamahitek.com/en/what-is-the-difference-between-regression-and-classification-in-machine-learning/

Linear Classification

- **Linear separability** refers to the ability to separate two classes of data points in a feature space using a single straight hyperplane. Mathematically, a dataset is said to be linearly separable if there exists a hyperplane such that all data points belonging to one class are on one side of the hyperplane, while all data points belonging to the other class are on the opposite side.
- In $\mathbb{R}^n$, the hyperplane can be defined as:

$$w.x + b = 0$$

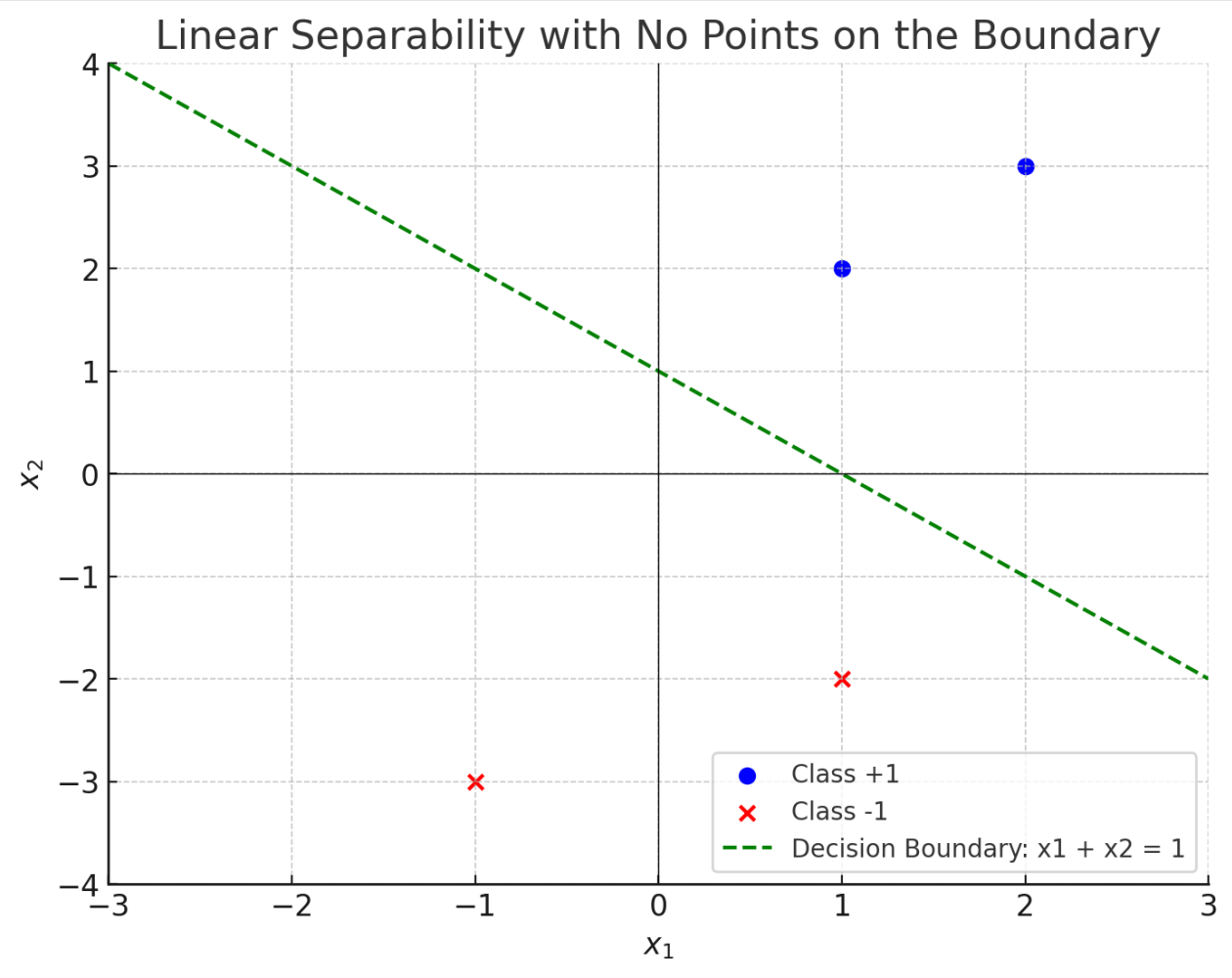
Where:

- w is the weight vector (defining the orientation of the hyperplane),
- x is the input feature vector, and
- b is the bias term (defining the position of the hyperplane).

For a dataset to be linearly separable, there must exist a weight vector w and a bias b such that:

$$y_i(w \cdot x_i + b) > 0 \; \forall i$$

Linear Classification



- Hyperplane

$$w_1 x_1 + w_2 x_2 + b = 0$$

$$x_1 + x_2 - 1 = 0$$

- Datapoints

$$(1,1)\ ;\ w.x + b = 2 > 0\ (+1)$$

$$(2,2)\ ;\ w.x + b = 4 > 0\ (+1)$$

$$(1,-2)\ ;\ w.x + b = -3 < 0\ (-1)$$

$$(-1,-3)\ ;\ w.x + b = -5 < 0\ (-1)$$

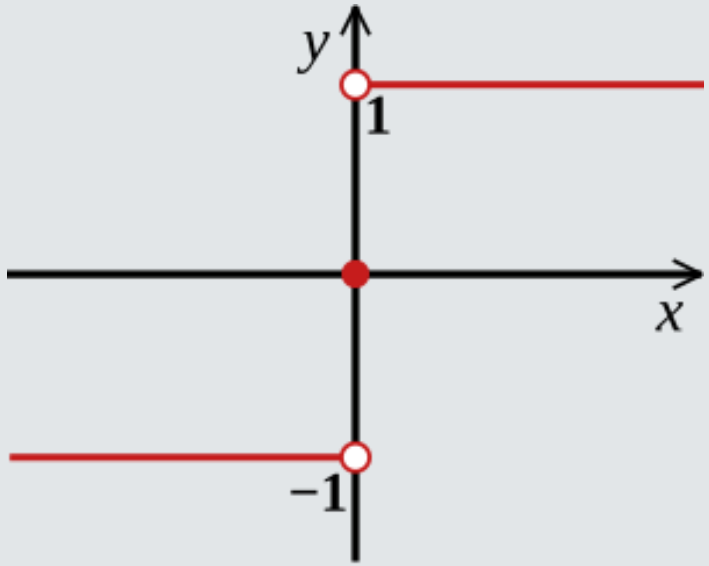
Linear Classification

- **Perceptron Algorithm:** foundational algorithm for binary classification, aiming to find a linear decision boundary that separates two classes. It updates its weights iteratively based on misclassifications.

$$f(x) = sign(w.x + b)$$

Where

- $w \in \mathbb{R}^n$ is the weight vector
- $b \in \mathbb{R}$ is the bias term



Linear Classification

- **Perceptron Algorithm Update Rule:** When a data point (x_i, y_i) is misclassified, update the weights and bias:

$$w \leftarrow w + \eta y_i x_i$$
$$b \leftarrow b + \eta y_i$$

where η is the learning rate

How did we derive this update?

Linear Classification

- **Gradient Descent Interpretation:** The perceptron update can be viewed as a form of gradient descent on the hinge loss:

$$L(w, b) = \max(0, -y_i(w \cdot x_i + b))$$

- **Partial Derivatives:**

$$\frac{\partial L}{\partial w} = \begin{cases} -y_i x_i \; ; if \; y_i(w.x_i + b) \leq 0 \\ 0 \; ; otherwise \end{cases}$$

$$\frac{\partial L}{\partial b} = \begin{cases} -y_i \; ; if \; y_i(w.x_i + b) \leq 0 \\ 0 \; ; otherwise \end{cases}$$

- **Weight Update as Gradient Step**:

$$w \leftarrow w \; - \eta \frac{\partial L}{\partial w} = w + \eta y_i x_i$$

$$b \leftarrow \; b \; - \eta \frac{\partial L}{\partial b} = b + \eta y_i$$

Linear Classification

- **Pseudo Code:**
- Inputs:

X: Training data matrix of shape (m, n) ; y: Labels vector of shape (m,), with values in {-1, +1}

η: Initial learning rate ; MaxIter: Maximum number of iterations ; LearningRateSchedule

- w = zeros(n); b = 0

For t = 1 to MaxIter:

For each (x_i, y_i) in (X, y):

if y_i * (w · x_i + b) <= 0:

w = w + η * y_i * x_i

b = b + η * y_i

η = LearningRateSchedule(η, t)

Output: weight vector w, bias b

Linear Classification

- **Numerical Example:** Consider 2D dataset

Sample	x_1	x_2	y
1	2	3	+1
2	1	1	-1
3	2	1	-1
4	3	3	+1
5	5	5	+1

$(2,3); w = [2,3], b = 1$
$(1,1); w = [1,2], b = 0$
$(2,1); w = [-1,1], b = -1$
$(3,3); w = [2,4], b = 0$
$(5,5)$; no update
$(2,3)$; no update
$(1,1); w = [1,3], b = -1$

$(2,1); w = [-1,2], b = -2$(Answer)

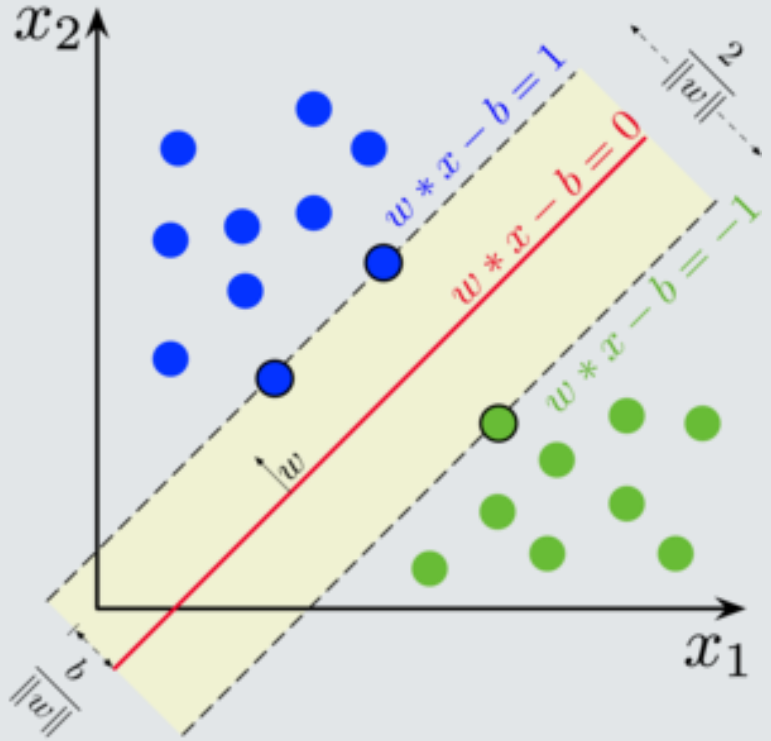
Is it unique?

Linear Classification

- **Limitations of the Perceptron:**
- Linearly Separable Data
  - Requirement: The perceptron algorithm converges only if the data is linearly separable.
  - Non-linear Separability: Fails on datasets like XOR where no linear hyperplane can separate the classes (Infinite Iterations)
- No Margin Optimization
  - No Maximum Margin: The perceptron does not seek the hyperplane with the largest margin, potentially leading to poor generalization.

Linear Classification (SVM)

- **Support Vector Machine (SVM):** aims to find the maximum margin hyperplane, which separates two classes with the largest possible margin. This ensures better generalization to unseen data.



https://en.wikipedia.org/wiki/Support_vector_machine

Linear Classification (SVM)

- For linear classification, the decision boundary is:

$$w \cdot x + b = 0$$

- The margin is defined as the distance between the decision boundary and the closest data points from either class. The goal is to maximize this distance.
- For correctly classified points, we have:

$$y_i(w \cdot x_i + b) \geq 1;\ \forall i$$

- This inequality ensures that points from class +1lie on one side of the hyperplane, and points from class −1lie on the other side, separated by a margin of at least 1 unit.
- The **margin** is given by $\frac{2}{\|w\|}$, where $\| w \|$ is the Euclidean norm of the weight vector w. To maximize the margin, we minimize $\| w \|$, or equivalently, minimize $\frac{\|w\|^2}{2}$.

Linear Classification (SVM)

- For linear classification, the decision boundary is:

$$w \cdot x + b = 0$$

- The margin is defined as the distance between the decision boundary and the closest data points from either class. The goal is to maximize this distance.
- For correctly classified points, we have:

$$y_i(w \cdot x_i + b) \geq 1;\ \forall i$$

- This inequality ensures that points from class +1lie on one side of the hyperplane, and points from class −1lie on the other side, separated by a margin of at least 1 unit.
- The **margin** is given by $\frac{2}{\|w\|}$, where $\| w \|$ is the Euclidean norm of the weight vector w. To maximize the margin, we minimize $\| w \|$, or equivalently, minimize $\frac{\|w\|^2}{2}$.

Linear Classification (SVM)

- The optimization problem is formulated as:

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

subject to the constraints

$$y_i(w.x_i + b) \geq 1 \; ; \forall i$$

- We solve this constrained optimization problem using Lagrange multiplers. We introduce a lagrange multipler $\alpha_i \geq 0$ for each constraint $y_i(w \cdot x_i + b) \geq 1$, and the Lagrangian is defined as:

$$L(w, b, \alpha) = \frac{1}{2}||w||^2 - \sum_{i=1}^{n} \alpha_i (y_i(w \cdot x_i + b) - 1)$$

Linear Classification (SVM)

$$L(w, b, \alpha) = \frac{1}{2}||w||^2 - \sum_{i=1}^{n} \alpha_i (y_i (w \cdot x_i + b) - 1)$$

$$\frac{\partial L}{\partial w} = w - \sum_{i=1}^{n} \alpha_i y_i x_i \Rightarrow w = \sum_{i=1}^{n} \alpha_i y_i x_i$$

$$\frac{\partial L}{\partial b} = -\sum_{i=1}^{n} \alpha_i y_i \Rightarrow \sum_{i=1}^{n} \alpha_i y_i = 0$$

If we replace the w with $\sum_{i=1}^{n} \alpha_i y_i x_i$

The **optimization problem** becomes

$$\max_{\alpha} -\frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} \alpha_i \alpha_j y_i y_j (x_i x_j) + \sum_{i=1}^{n} \alpha_i$$

$$s.t. \sum_{i=1}^{n} \alpha_i \, y_i = 0, \alpha_i \geq 0 \; \forall i$$

Linear Classification (SVM) [Dual]

Extra Notes (Beyond the scope of the course):

To solve the following problem there are different optimization algorithms that can be used to solve it (Interior Point Methods, Sequential Minimal Optimization (SMO)). In addition

Why $\min||w||^2$?

Linear Classification (SVM)

- The margin is the **perpendicular distance** between the decision boundary (hyperplane) and the closest data points.
- The distance d from a point x_0 to the hyperplane can be calculated using the formula for the distance between a point and a plane:

$$d = \frac{|w.x_0 + b|}{||w||}$$

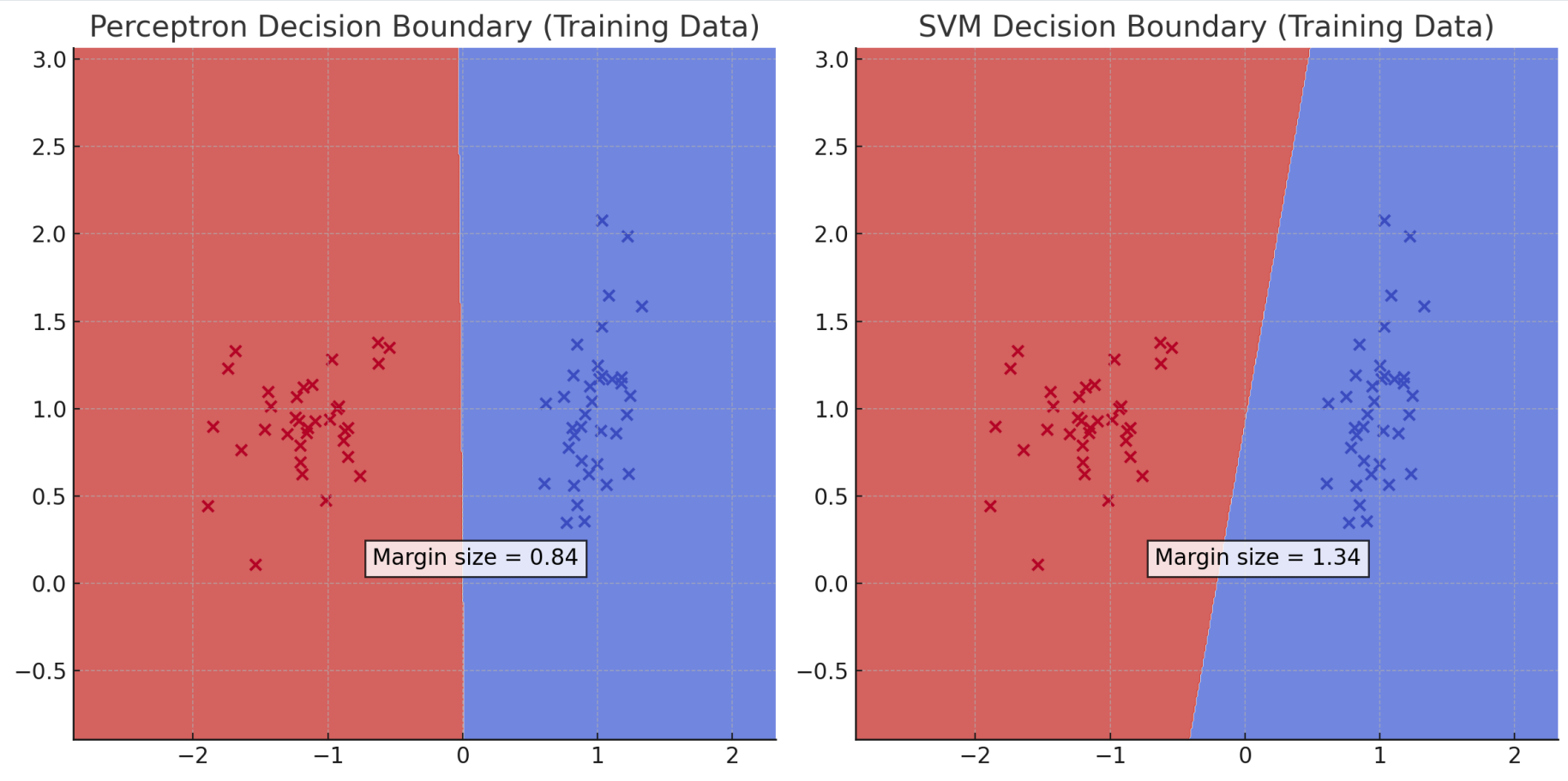
This formula gives the **perpendicular distance** from the point x_0 to the hyperplane.

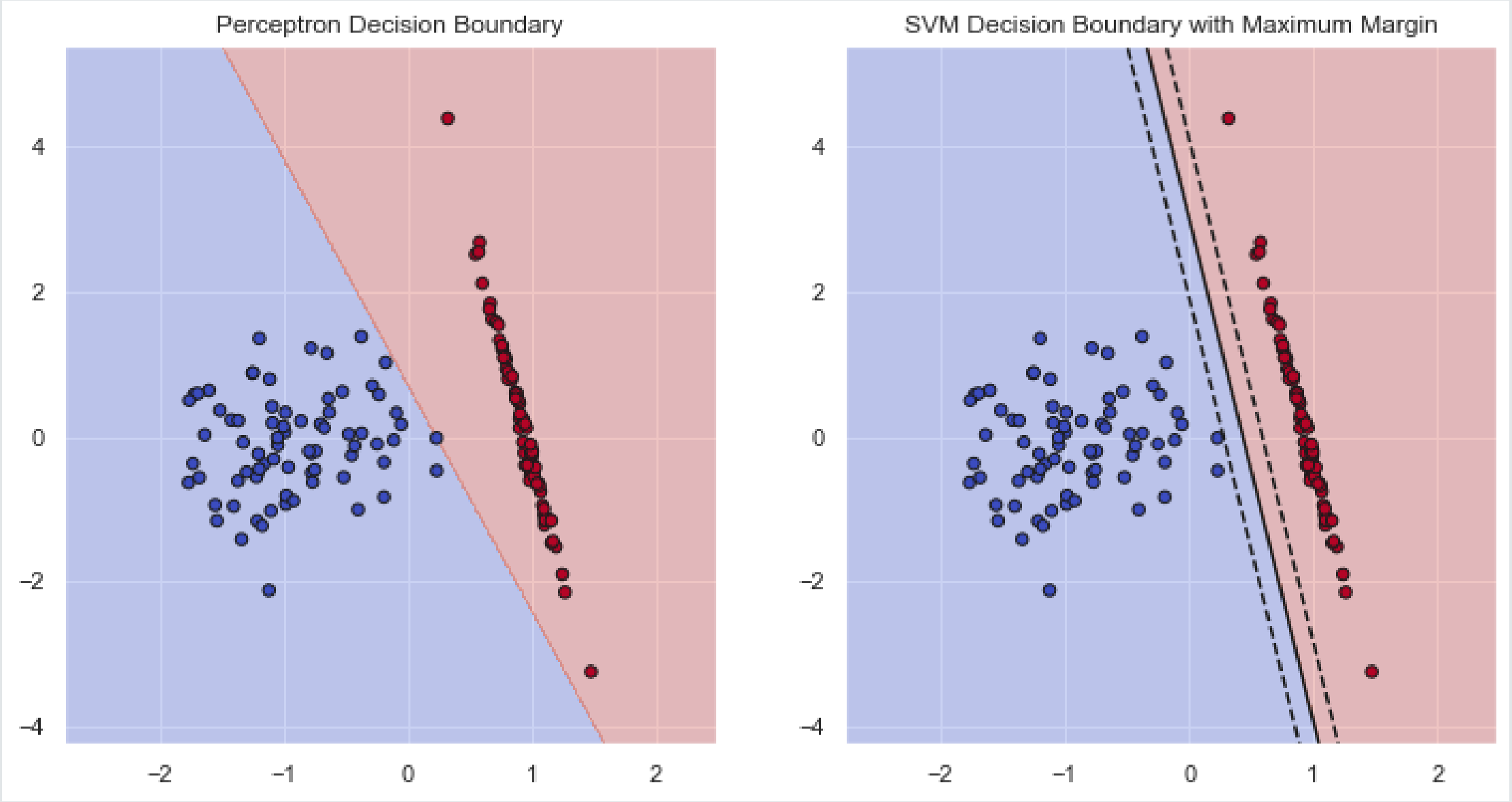
- The decision boundary $w.x + b = 0$
- The hyperplanes that pass through the support vectros, given by $w.x + b = \pm 1$
- The distance between these two hyperplanes is the margin. $\frac{2}{||w||}$

Thus, the margin is inversely proportional to $||w||$, meaning the larger $||w||$, the smaller the margin and vice versa. We square it since it would be easier to minimize since it is quadratic that is differentiable.

Linear Classification (SVM)

- **Limitations of SVM:**
- Computational Complexity


Perceptron Decision Boundary (Training Data)
Margin size = 0.84
SVM Decision Boundary (Training Data)
Margin size = 1.34
3.0
2.5
2.0
1.5
1.0
0.5
0.0
−0.5
−2
−1
0
1
2



Experiment 1

Linear Regression

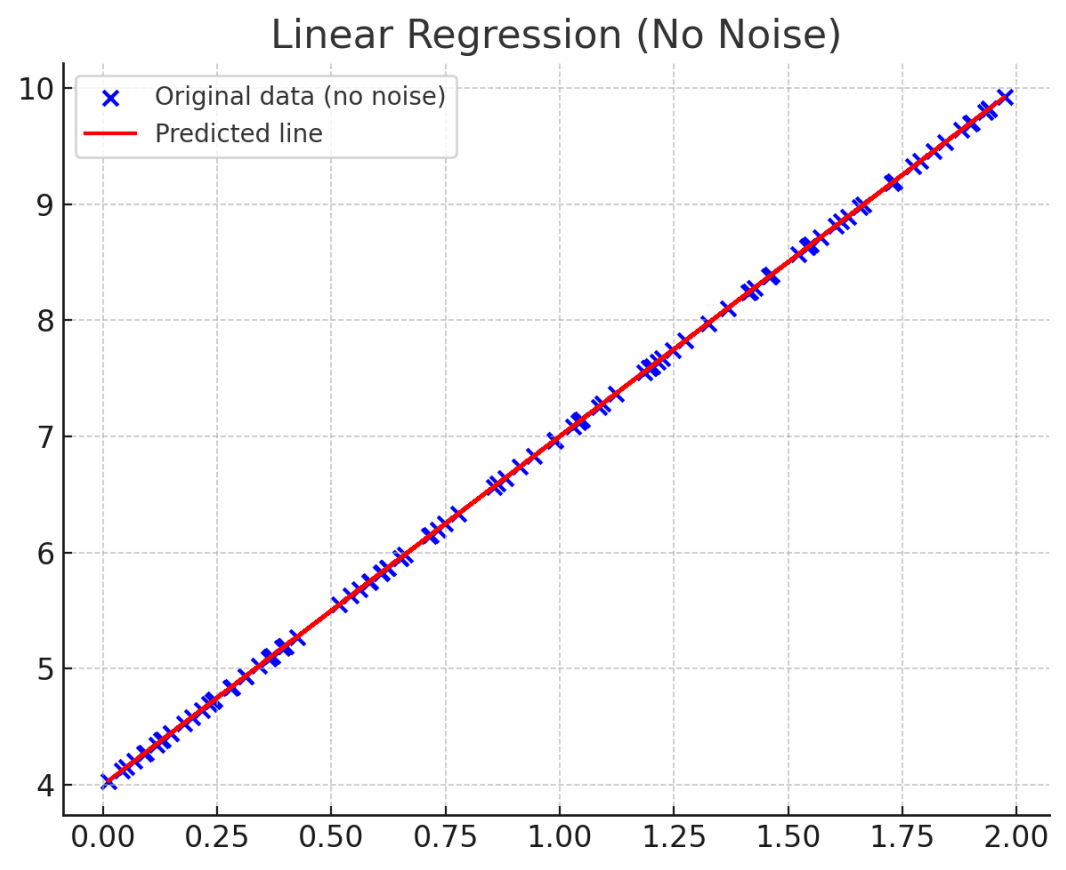
- Linear regression models the relationship between the input features $X \in R^{n \times p}$ and the target values $y \in \mathbb{R}^n$ as:

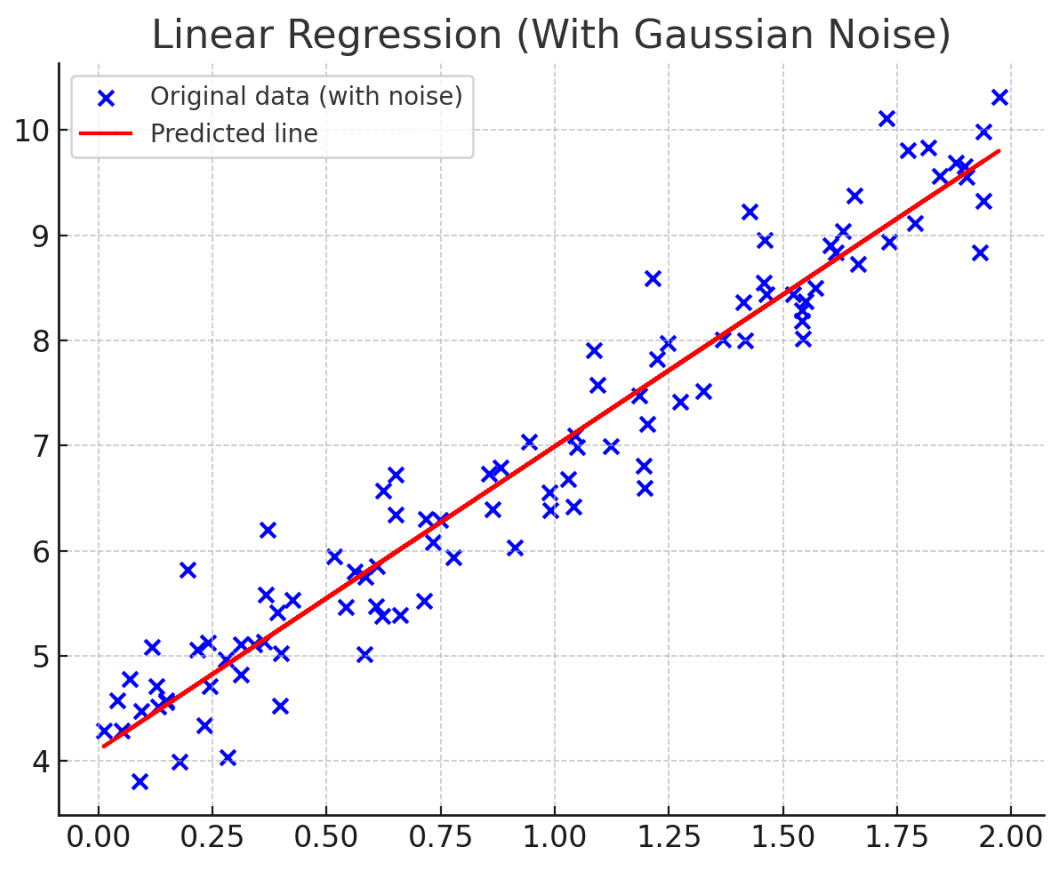
$$y = X\beta + \epsilon$$

- X is the design matrix of input features.
- β is the vector of coefficients.
- ϵ is the error term.
- To estimate β, we minimize the sum of squared residuals:

$$\min_{\beta} \left|\left|y - X\beta\right|\right|_2^2 = \min_{\beta} \sum_{i=1}^{n} (y_i - X_i\beta)^2$$

Linear Regression





Linear Regression

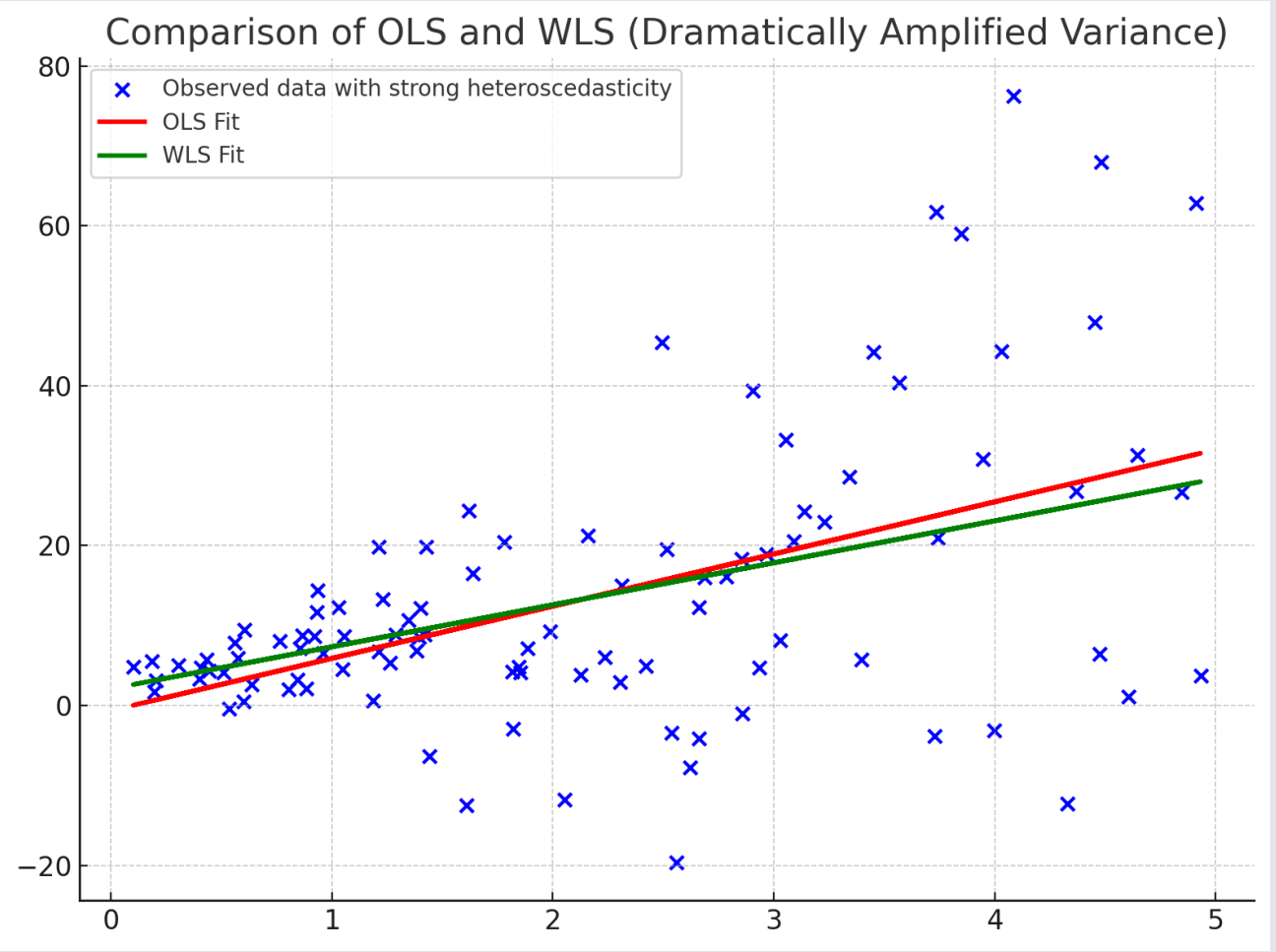
- Limitations:
- Linear relationship
- Constant Variance
- Singularity in $(X^T X)^{-1}$

Linear Regression

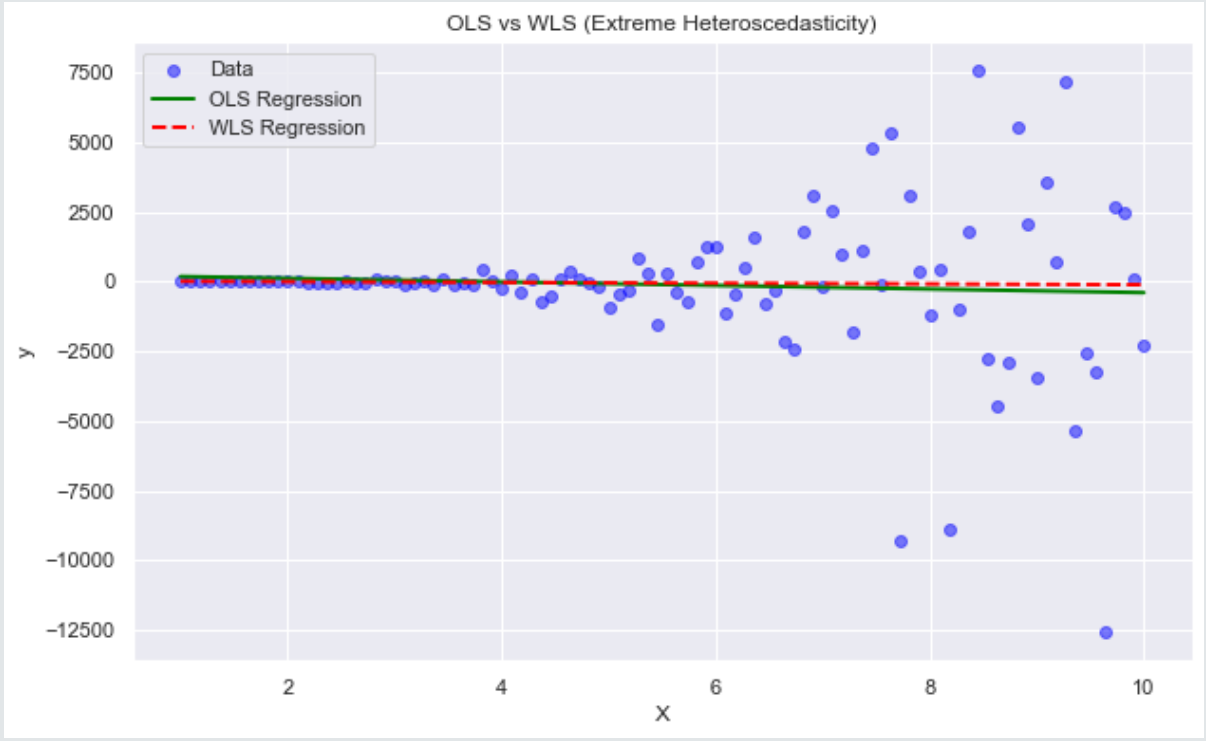
- **Weighted Least Square (WLS)**
- In WLS, we assume that the observations have **non-constant variance**, meaning the errors associated with different observations are not the same. The goal is to account for this heteroscedasticity by giving different weights to different observations.
- The idea is to minimize a weighted sum of the squared residuals, where observations with **higher variance** are given **less weight**, and observations with **lower variance** are given **more weight**.

Linear Regression

- **Weighted Least Square**



Linear Regression



Experiment 2

Linear Regression

- **Ridge Regression**
- In **Ordinary Least Squares (OLS)**, we aim to minimize the residual sum of squares:

$$\min_{\beta} \left|\left|y - X\beta\right|\right|_2^2$$

- However, when the matrix XX^T is **singular** or **nearly singular** (often caused by **multicollinearity**), the inverse $(X^TX)^{-1}$doesn't exist or becomes unstable. This instability makes OLS unreliable for certain datasets.
- To overcome this, **Ridge Regression** introduces a regularization term that penalizes large values of β, effectively **shrinking** the coefficients and stabilizing the solution. This regularization is key to handling multicollinearity and singularity issues.

$$\min_{\beta} \left|\left|y - X\beta\right|\right|_2^2 + \lambda \left|\left|\beta\right|\right|_2^2$$

- $\lambda \geq 0$ is the **regularization parameter**. It controls the amount of regularization applied:
  - $\boldsymbol{\lambda = 0}$ *gives the OLS solution,*
  - $\boldsymbol{\lambda > 0}$ *penalizes large coefficients, reducing overfitting and addressing multicollinearity.*

Linear Regression (Extra Reading)

- In **Lasso Regression,** introduces an L1 regularization term, which penalizes the absolute value of the coefficients. The objective function is:

$$\min_{\beta} \left|\left|y - X\beta\right|\right|_2^2 + \lambda \left|\left|\beta\right|\right|_1$$

Pros	Cons
Feature Selection: Lasso performs automatic feature selection by driving some coefficients to exactly zero, making the model sparse and more interpretable.	**Instability with Correlated Features**: Lasso tends to arbitrarily select one feature from a group of highly correlated variables, which may result in unstable solutions.
Good for High-Dimensional Data: Lasso works well when there are many features, especially when some are irrelevant, as it can exclude irrelevant features.	**Bias**: Because of the L1 penalty, Lasso may shrink important variables more than necessary, introducing bias into the estimates.
Improved Interpretability: By selecting a subset of features, Lasso creates a simpler, more interpretable model.	**No Closed-Form Solution**: Lasso requires iterative algorithms (like Coordinate Descent, or IRLS) to compute the solution, which can be computationally expensive for very large datasets.

Linear Regression (Extra Reading)

- **Elastic Net Regression,** is a hybrid of Ridge and Lasso that combines both L1 and L2 penalties. The objective function is:

$$\min_{\beta} \left|\left|y - X\beta\right|\right|_2^2 + \lambda_1 \left|\left|\beta\right|\right|_2^2 + \lambda_2 \left|\left|\beta\right|\right|_1$$

Pros	Cons
Combines the Strengths of Ridge and Lasso: Elastic Net combines the feature selection ability of Lasso with the stability of Ridge, making it useful when dealing with highly correlated features.	**More Complex Tuning**: Elastic Net requires tuning two hyperparameters (λ_1 and λ_2), which adds complexity to the model selection process.
Stability with Correlated Features: Unlike Lasso, Elastic Net doesn't arbitrarily pick one feature from a group of correlated features; instead, it tends to include all or none of the correlated features.	**Can Over-Select Features**: In some cases, Elastic Net may include too many features, especially when λ_2 (the L2 penalty) dominates.
Flexible Regularization: The model can be tuned to prioritize either the L1 or L2 penalty, offering flexibility based on the dataset.	**Computationally Intensive**: Similar to Lasso, Elastic Net does not have a closed-form solution and requires iterative algorithms, which can be expensive for large datasets.

Linear Regression (Extra Reading)

	Ridge	Lasso	Elastic Net
Penalty	L2 (Sum of squared coefficients)	L1 (Sum of absolute coefficients)	L1 + L2 (Combination of both)
Feature Selection	No	Yes (Sparse solution)	Yes (Incorporates groups of correlated features)
Handling Multicollinearity	Reduces multicollinearity, but keeps all features	May arbitrarily select one feature from a correlated group	Handles correlated features better than Lasso
Bias-Variance Tradeoff	Lower bias, but all coefficients shrink	More bias, as L1 penalty shrinks important variables	Flexible; can balance bias-variance tradeoff
Interpretability	Lower	Higher (fewer features)	Moderate (depends on tuning)
Computational Complexity	Lower (closed-form solution)	Higher (iterative solution)	Higher (iterative solution with two penalties)