19ECS741 Machine Learning

O. V. Ramana Murthy

Course Outcome 3

Develop regression models for predictive tasks.

Contents

1. Gradient Descent,
2. Linear, Quadratic and Polynomial Regression,
3. SVR, Kernel Methods
4. Regularization

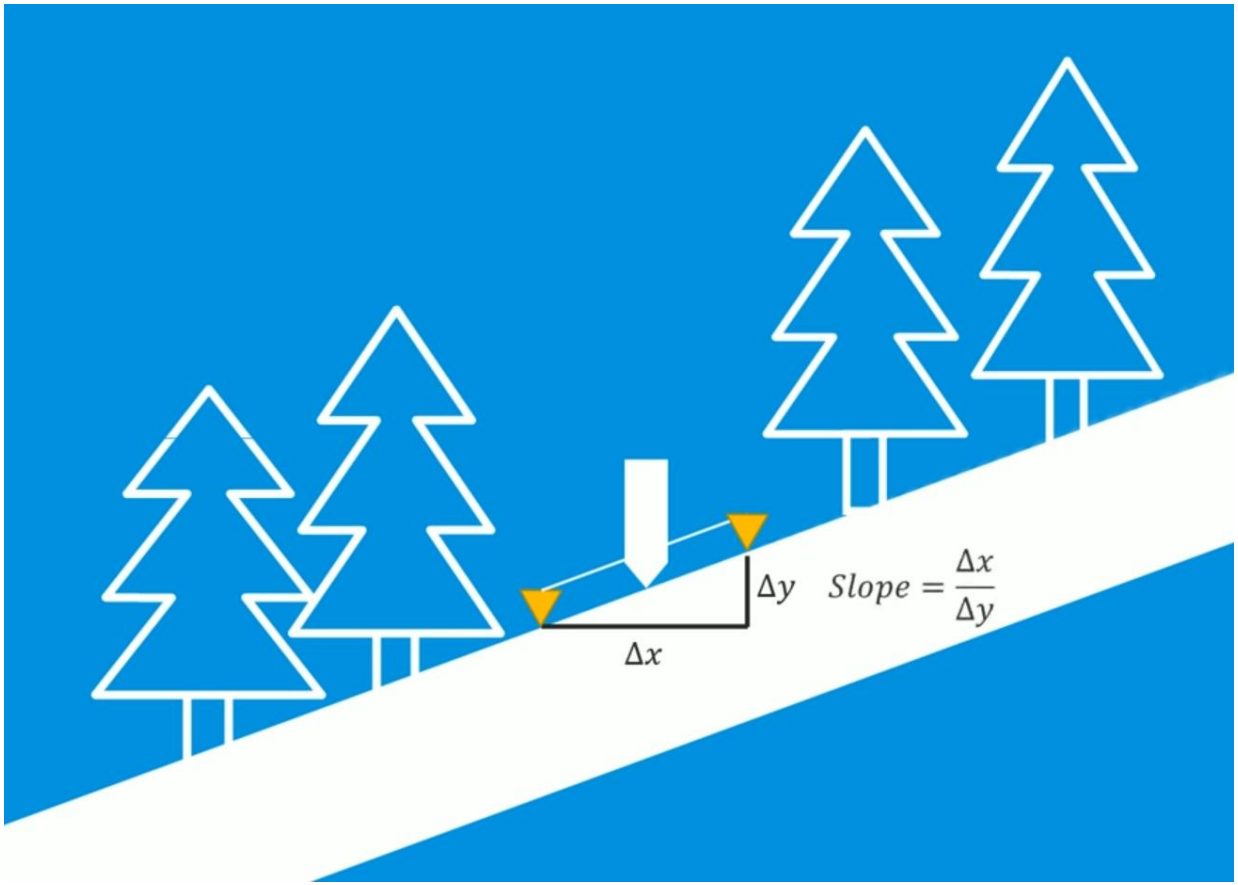
Reference

Chapters 3, 4, 5, 6

Aurelion Geron, Hands-on Machine Learning with Scikit-Learn, Keras, and Tensor Flow: Concepts, Tools and Techniques to build Intelligent Systems, 3/e, O'Reilly Media, 2022.

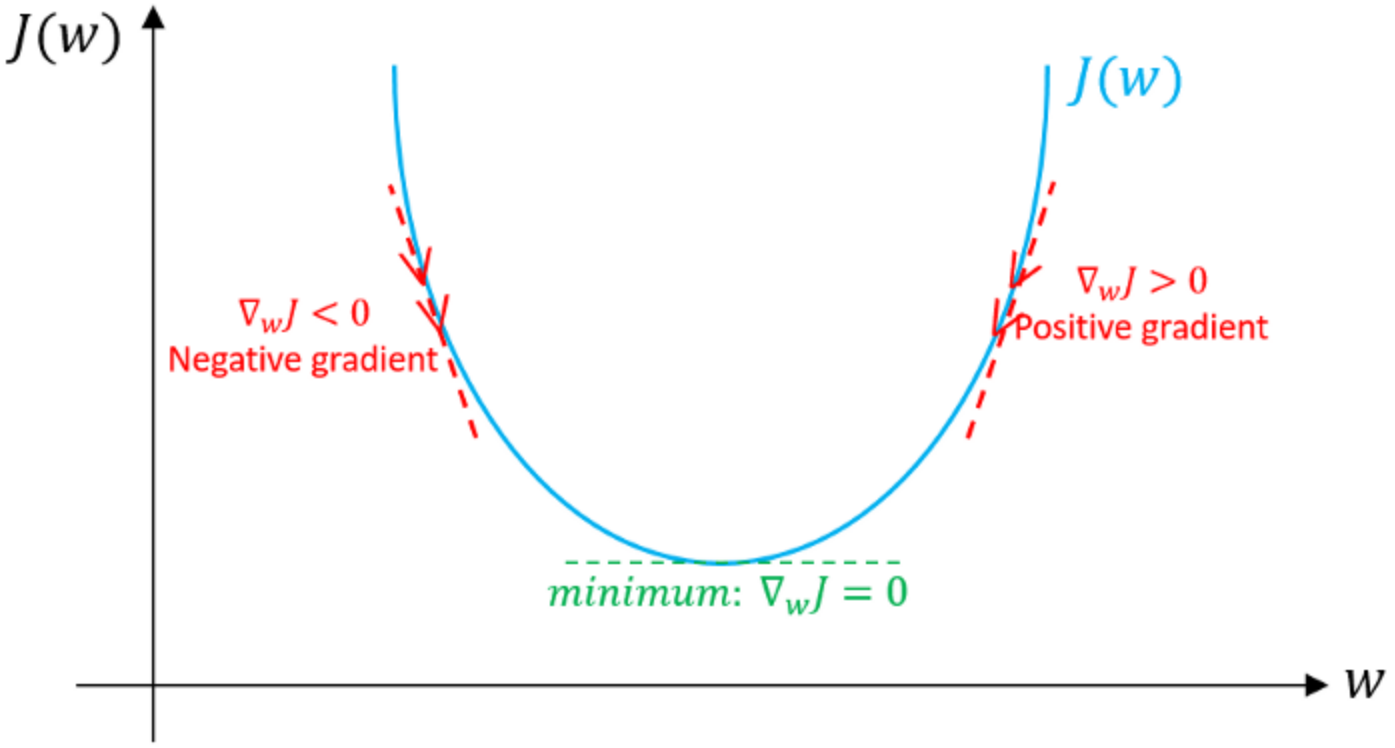
Motivation

- Gradient represents how STEEP a slope is.



Motivation

- Gradient represents how STEEP a slope is.
- Uphill is positive; downhill is negative.
- One direction – Derivative. multi-direction – Gradient.



Gradient Descent

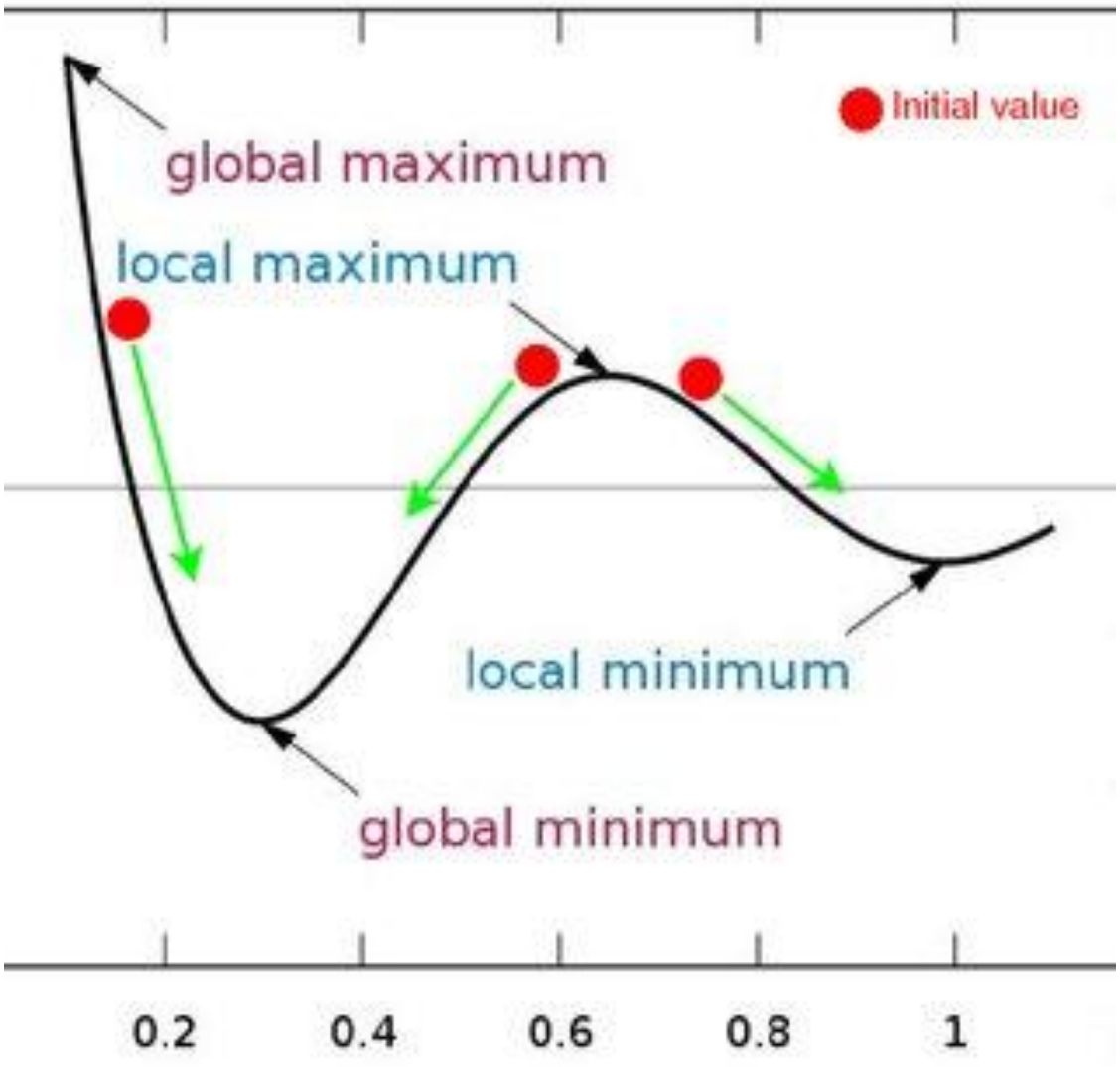
Gradient Descent is an optimization algorithm used to **minimize a loss function** by iteratively updating model parameters (like weights in linear regression)

1. The algorithm computes the gradient (slope) of the loss function ($L(\theta)$) with respect to the parameters θ.
2. It updates parameters in the opposite direction of the gradient to reduce the error.

$$\theta \leftarrow \theta - \alpha \nabla_\theta L(\theta)$$

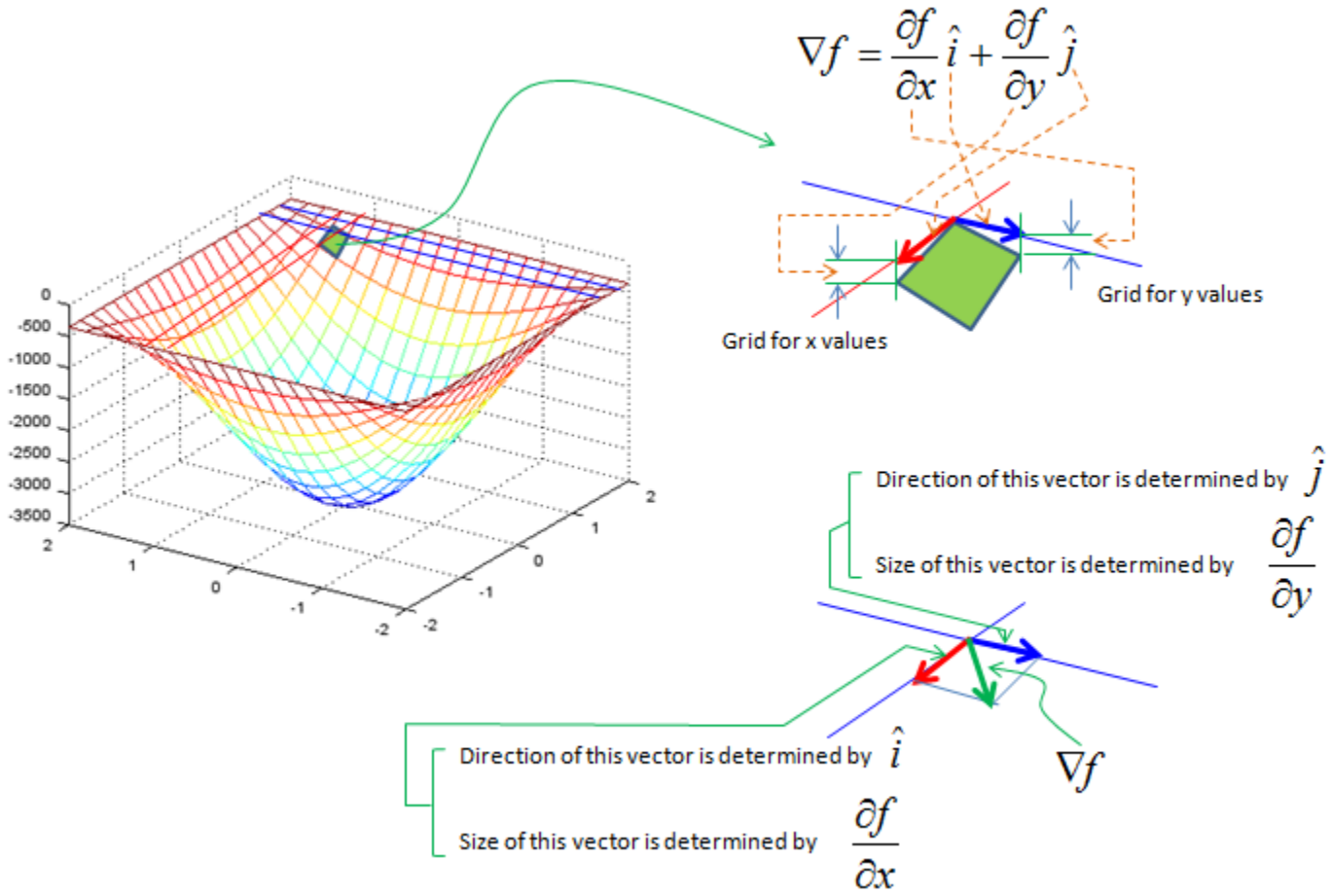
α is learning rate range is [0–1]

Local vs Global Solution



Motivation

- One direction – Derivative. multi-direction – Gradient.



Source: http://www.sharetechnote.com/html/Calculus_Gradient.html

Gradient Descent

Gradient Descent is an optimization algorithm used to **minimize a loss function** by iteratively updating model parameters (like weights in linear regression)

1. The algorithm computes the gradient (slope) of the loss function ($L(\theta)$) with respect to the parameters θ.
2. It updates parameters in the opposite direction of the gradient to reduce the error.

$$\theta \leftarrow \theta - \alpha \nabla_\theta L(\theta)$$

α is learning rate range is [0–1]

Linear Regression

Linear regression is one of the simplest and most widely used **supervised learning** algorithms for predicting a **continuous numeric output** based on one or more input features.

It assumes a **linear relationship** between the input features

$$y = w^T x + b$$

w: weights (slopes)

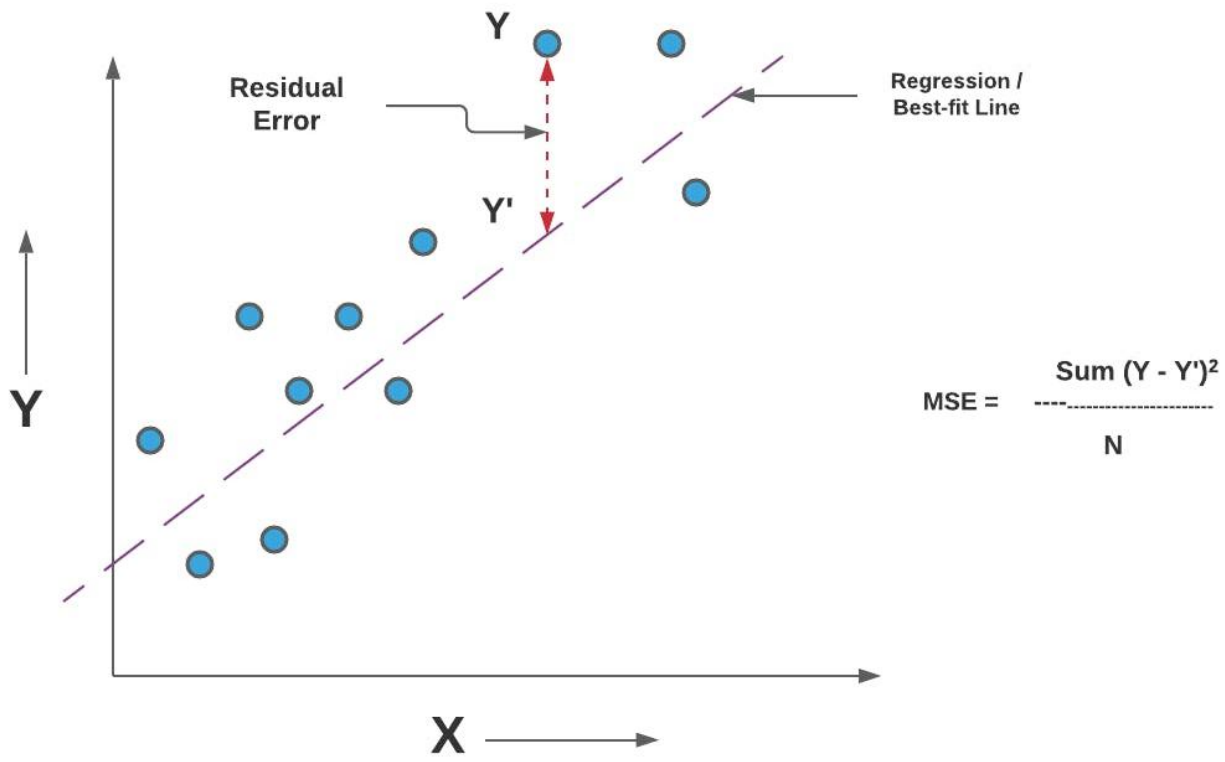
b: bias (intercept)

x: input vector

y: predicted output

Linear Regression

Goal: Find the values of *w* and *b* that **minimize the difference** between the predicted and actual values, typically using the **Mean Squared Error (MSE)**



Algorithm to Update the Weights

Given : Input features: $X \in R^{n \times d}$;Output values: $y \in R^{n}$

1. **Initialize weights and bias**
2. **Repeat for T epochs:**

For each sample $x^{(i)}, y^{(i)}$:

a. Compute prediction $\hat{y}^{(i)} = w^T x^{(i)} + b$

b. Compute error $e^{(i)} = \hat{y}^{(i)} - y^{(i)}$

$$L(w, b) = \frac{1}{n} \sum_{i=1}^{n} \left(\hat{y}^{(i)} - y^{(i)} \right)^2$$

c. Compute gradients $\frac{\partial L}{\partial w} = \frac{1}{n} \sum_{i=1}^{n} e^{(i)} x^{(i)}$; $\frac{\partial L}{\partial b} = \frac{1}{n} \sum_{i=1}^{n} e^{(i)}$

3. **Update the parameters**

$$w \leftarrow w - \alpha \frac{\partial L}{\partial w} ; b \leftarrow b - \alpha \frac{\partial L}{\partial b}$$

Example

$x^{(i)}$	$y^{(i)}$
1	2
2	3
3	4

Initial values:

$w=0$; $b=0$;

$\alpha=0.1$ (learning rate)

Step 1: Predictions

$y^1=w\cdot 1+b=0$; $y^2=w\cdot 2+b=0$; $y^3=w\cdot 3+b=0$

Step 2: Errors

$e^1=y^1-y^1=0-2=-2$; $e^2=0-3=-3$; $e^3=0-4=-4$

Example

3. Compute gradients

$$\frac{\partial L}{\partial w} = \frac{1}{n}\sum_{i=1}^{n} e^{(i)} x^{(i)} =$$

$$\frac{\partial L}{\partial b} = \frac{1}{n}\sum_{i=1}^{n} e^{(i)} =$$

4. Update Parameters

$$w \leftarrow w - \alpha \frac{\partial L}{\partial w}; =$$

$$b \leftarrow b - \alpha \frac{\partial L}{\partial b} =$$

Example

3. Compute gradients

$$\frac{\partial L}{\partial w} = \frac{1}{n}\sum_{i=1}^{n} e^{(i)} x^{(i)} = \frac{1}{3}[-2 \times 1 - 3 \times 2 - 4 \times 3] = -6.67$$

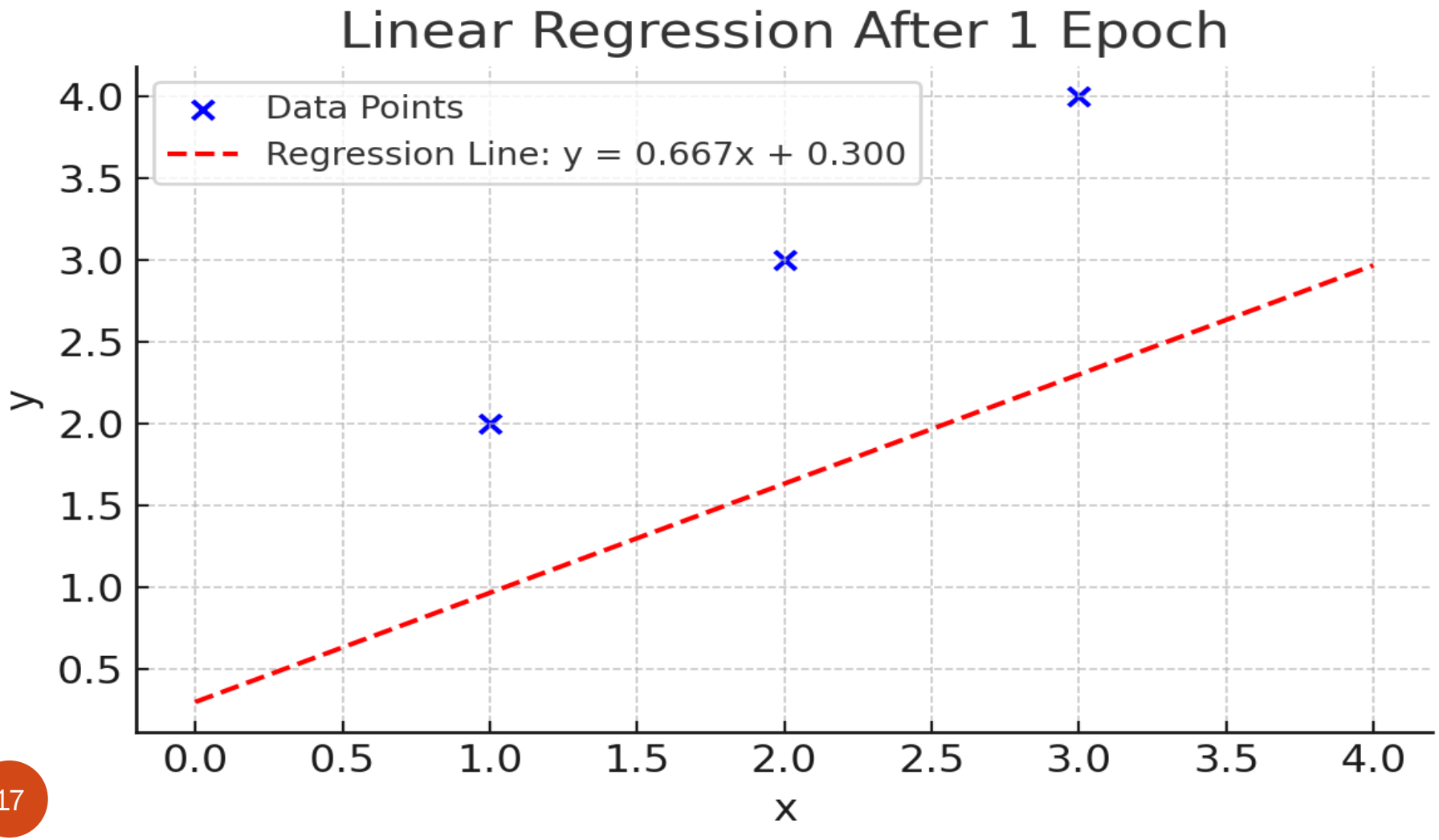
$$\frac{\partial L}{\partial b} = \frac{1}{n}\sum_{i=1}^{n} e^{(i)} = \frac{1}{3}[-2 - 3 - 4] = -3$$

4. Update Parameters

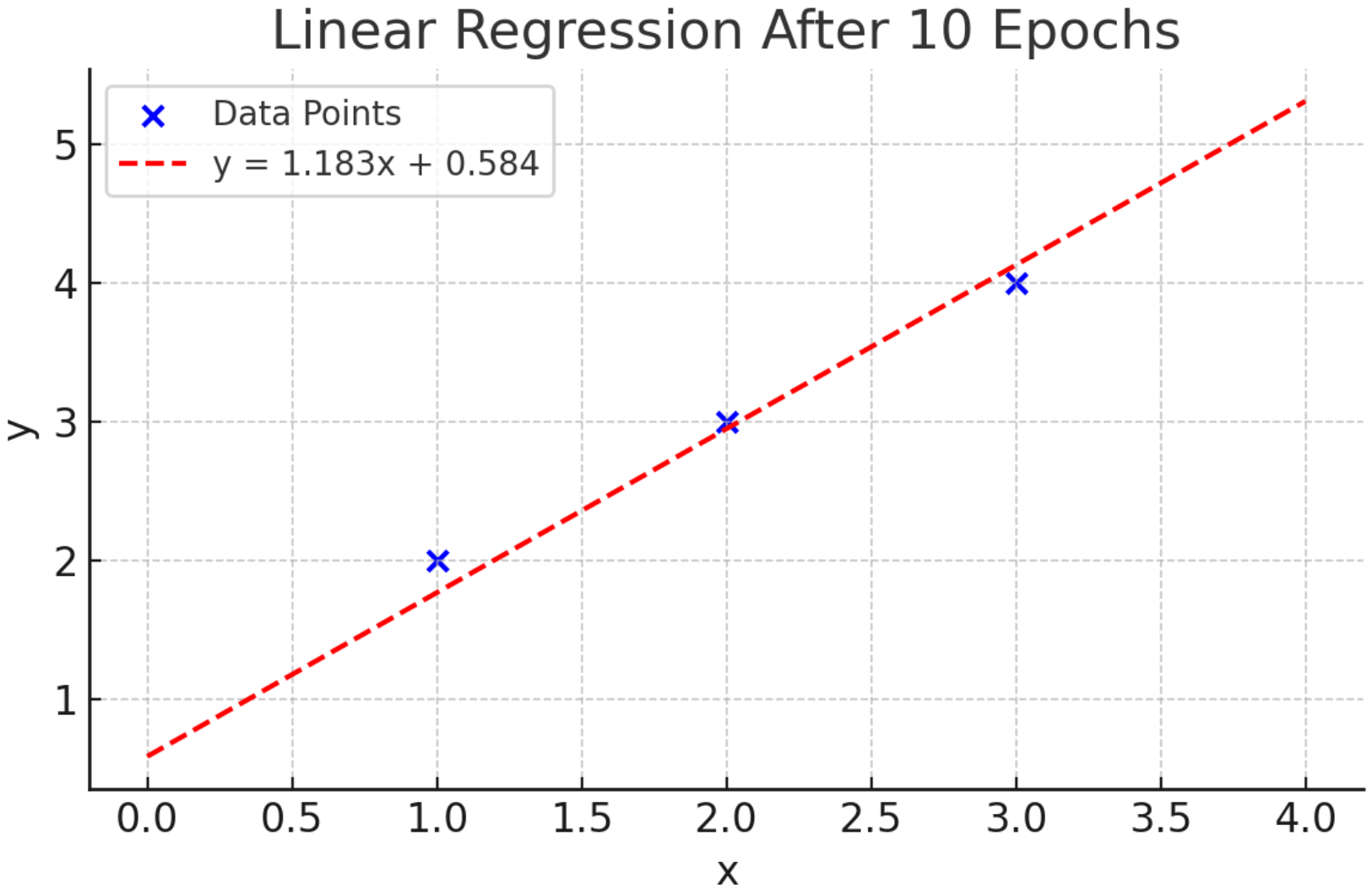
$$w \leftarrow w - \alpha \frac{\partial L}{\partial w}; = 0 - (-0.1)(-0.67) = 0.667$$

$$b \leftarrow b - \alpha \frac{\partial L}{\partial b} = 0 - (-0.1)(-3) = 0.3$$

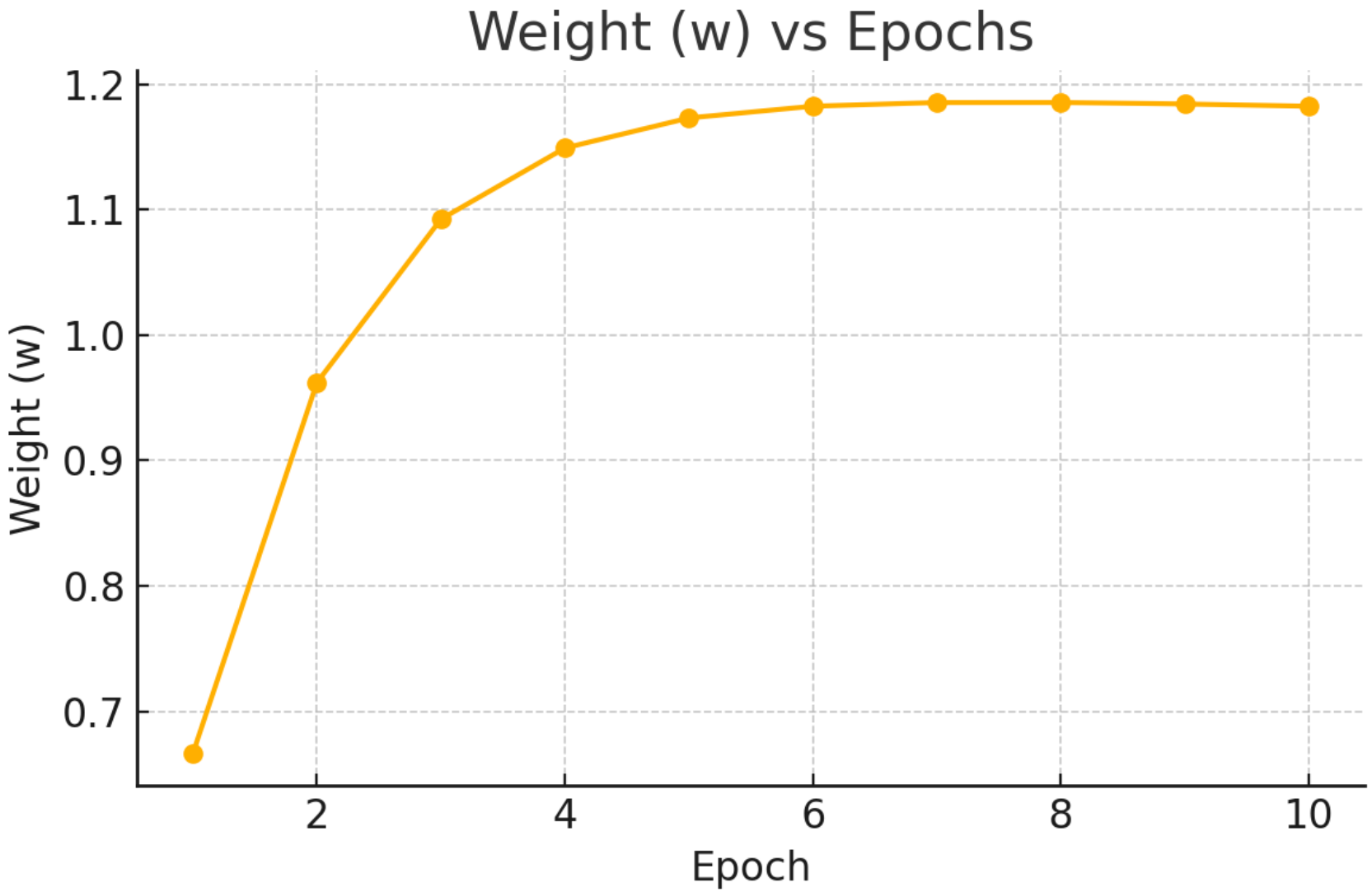
Example


Linear Regression After 1 Epoch
Data Points
Regression Line: y = 0.667x + 0.300
4.0
3.5
3.0
2.5
2.0
1.5
1.0
0.5
y
0.0
0.5
1.0
1.5
2.0
2.5
3.0
3.5
4.0
x

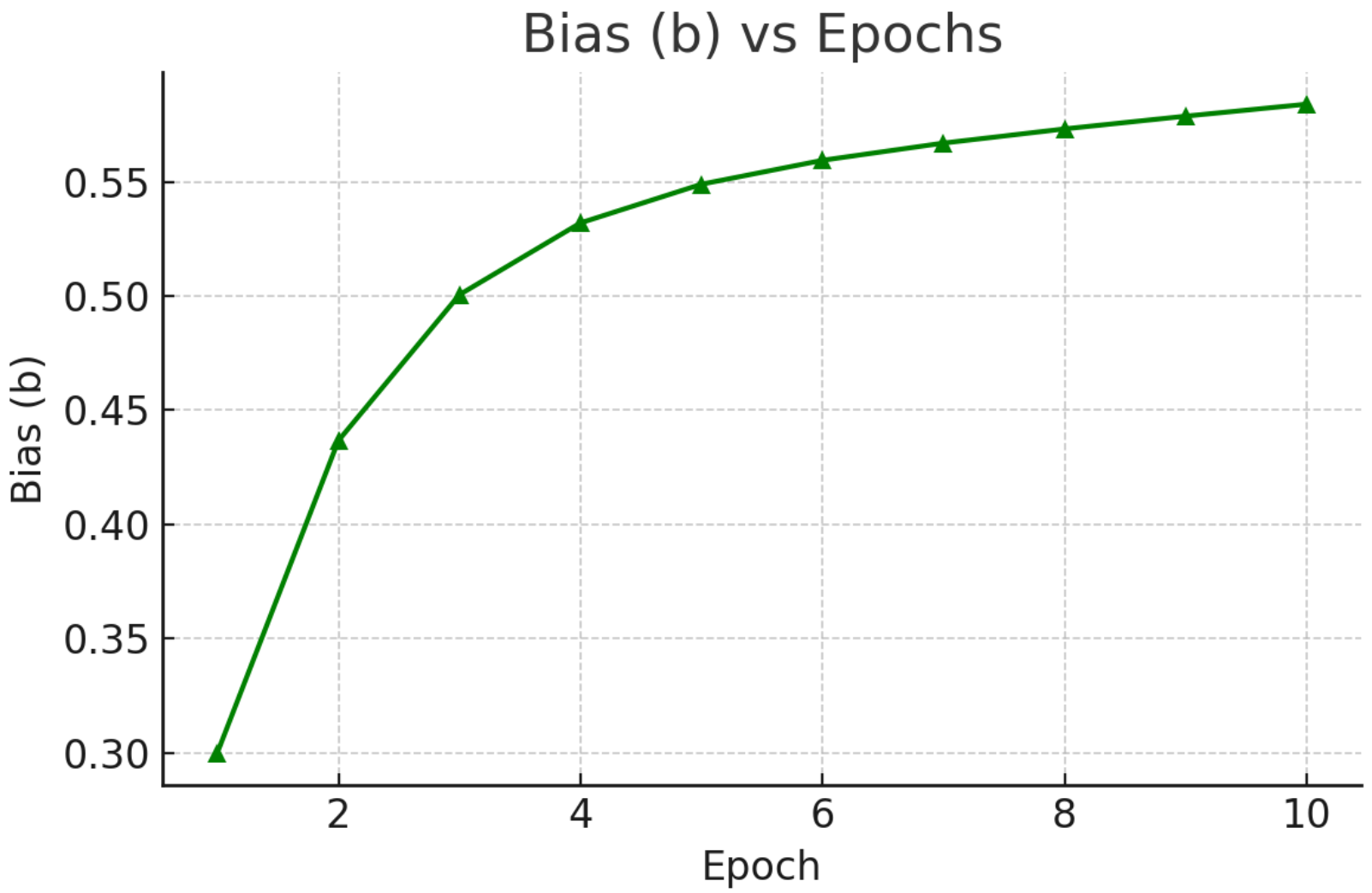
Example


Linear Regression After 10 Epochs
Data Points
y = 1.183x + 0.584
y
x
0.0
0.5
1.0
1.5
2.0
2.5
3.0
3.5
4.0
1
2
3
4
5

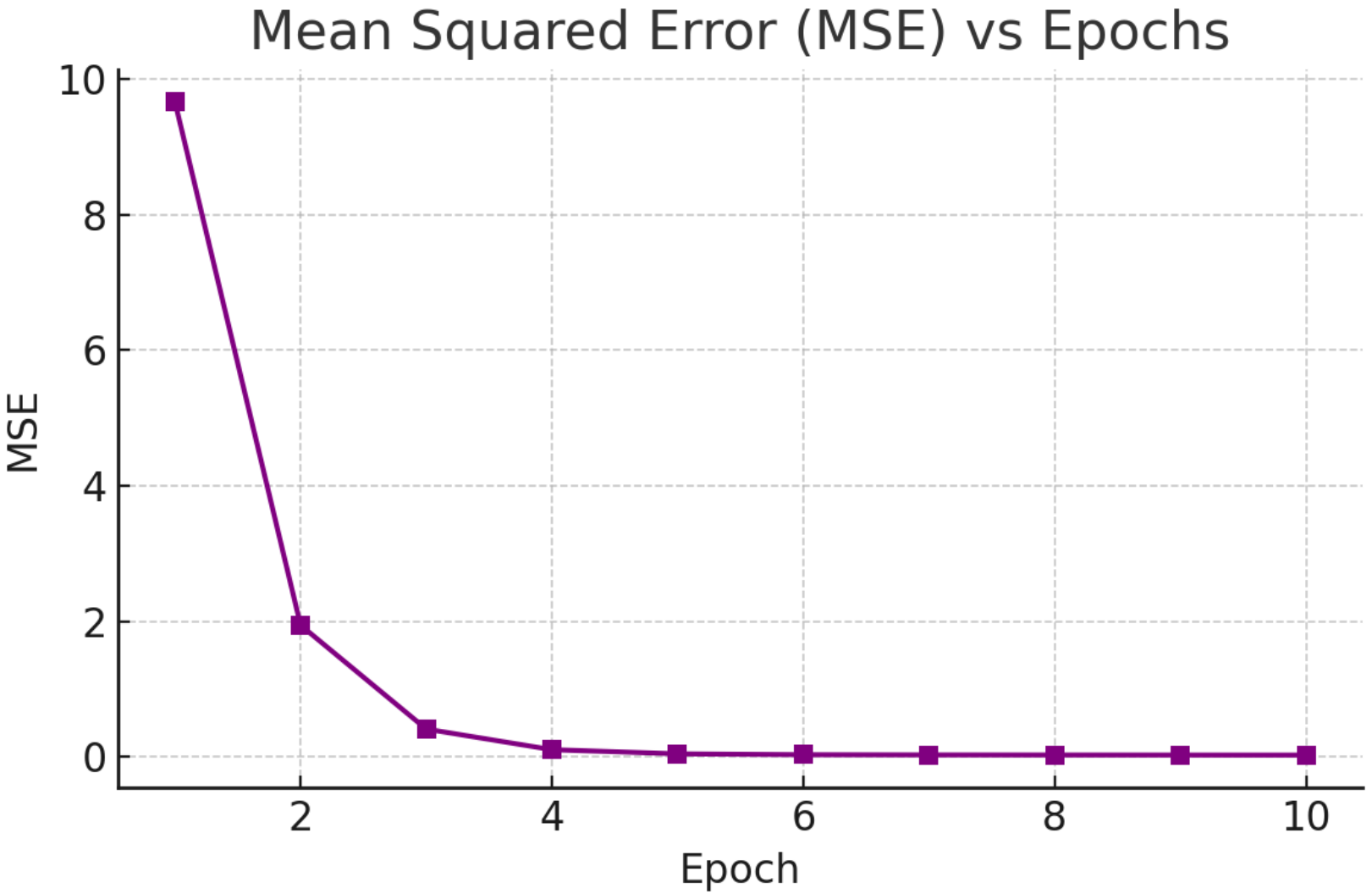
Learning curves



Learning curves



Learning curves



Quadratic Regression

Given : Input features: $X \in R^{n \times d}$;Output values: $y \in R^{n}$

1. **Initialize weights and bias**
2. **Repeat for T epochs:**

For each sample $x^{(i)}, y^{(i)}$:

a. Compute prediction $\hat{y}^{(i)} = a\left(x^{(i)}\right)^2 + b\left(x^{(i)}\right) + c$

b. Compute error $e^{(i)} = \hat{y}^{(i)} - y^{(i)}$

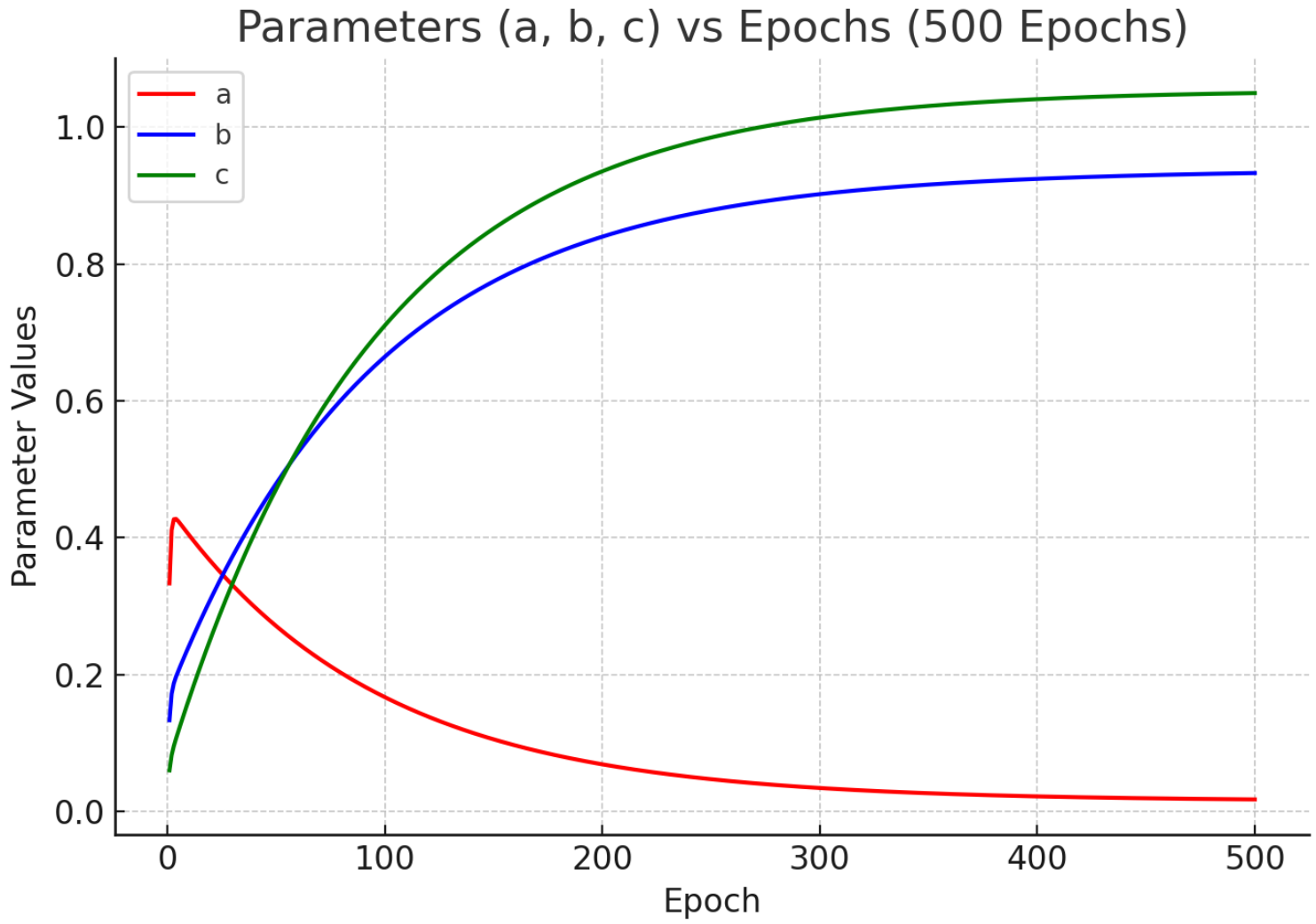
$$L(w, b) = \frac{1}{n}\sum_{i=1}^{n}\left(\hat{y}^{(i)} - y^{(i)}\right)^2$$

c. Compute gradients

$$\frac{\partial L}{\partial a} = \frac{1}{n}\sum_{i=1}^{n} e^{(i)}\left(x^{(i)}\right)^2; \frac{\partial L}{\partial b} = \frac{1}{n}\sum_{i=1}^{n} e^{(i)}\left(x^{(i)}\right), \frac{\partial L}{\partial c} = \frac{1}{n}\sum_{i=1}^{n} e^{(i)}$$

3. **Update the parameters** $a, b, c \leftarrow a, b, c - \alpha \frac{\partial L}{\partial a,b,c}$

Quadratic Regression



Polynomial Regression

- For degree n,

$$\hat{y} = w_n x^n + w_{n-1} x^{n-1} + \cdots + w_2 x^2 + w_1 x^1 + w_0$$

If $n = 1$; Linear regression

$n = 2$ Quadratic regression

$n = 3$ Cubic regression

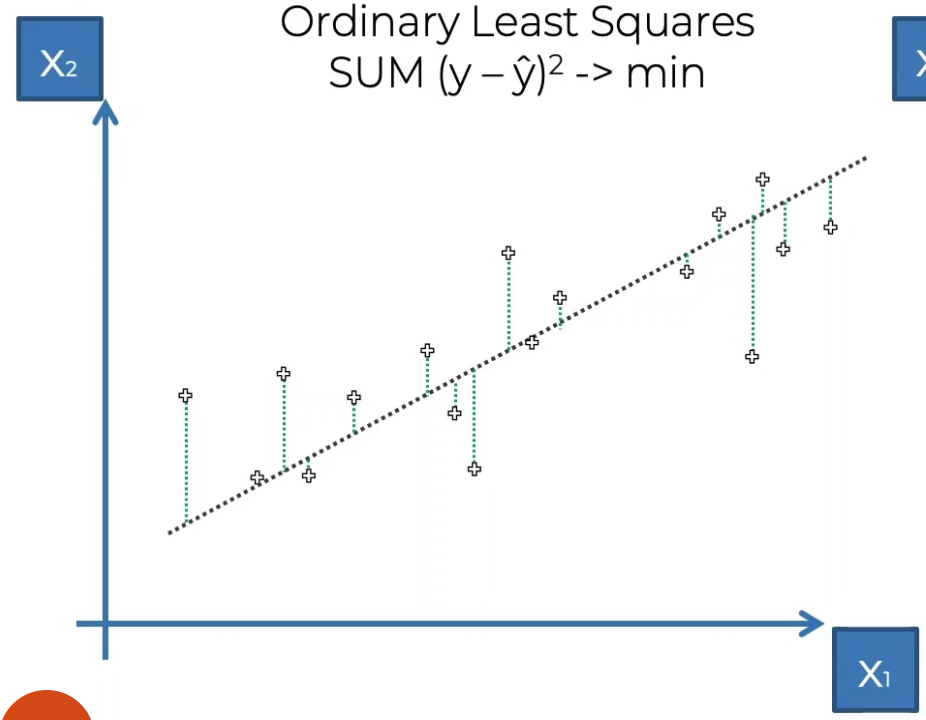
…

- Add Sigmoid function, then you use the probabilities for classification

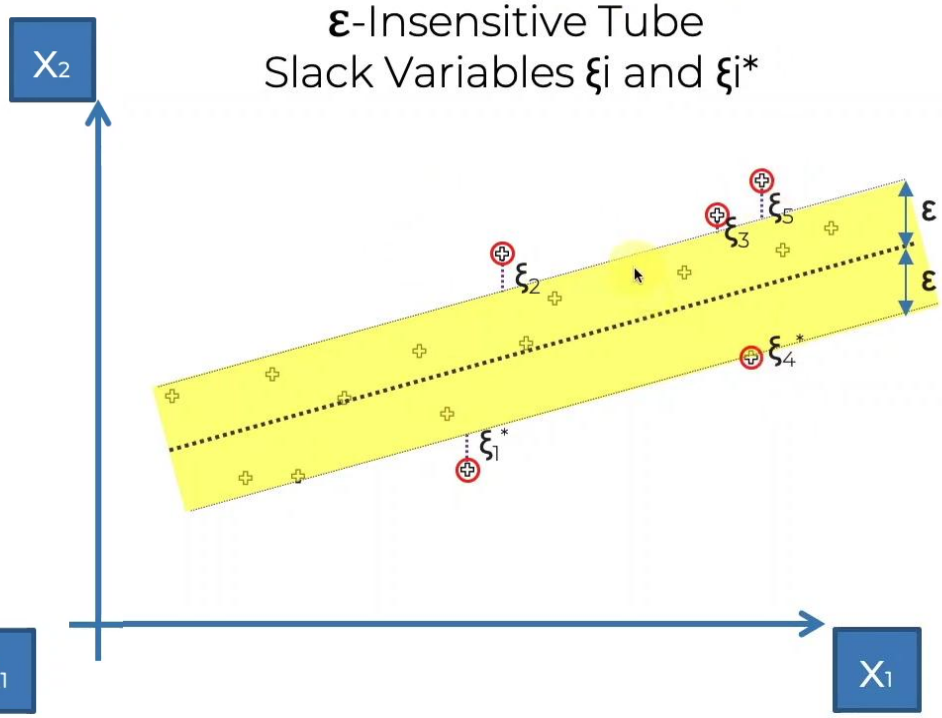
Support Vector Regression (SVR)

$$\frac{1}{2}\|w\|^2 + C\sum_{i=1}^{m}(\xi_i + \xi_i^*) \rightarrow min$$

Ordinary Least Squares
SUM $(y - \hat{y})^2$ -> min



ε-Insensitive Tube
Slack Variables **ξ**i and **ξ**i*

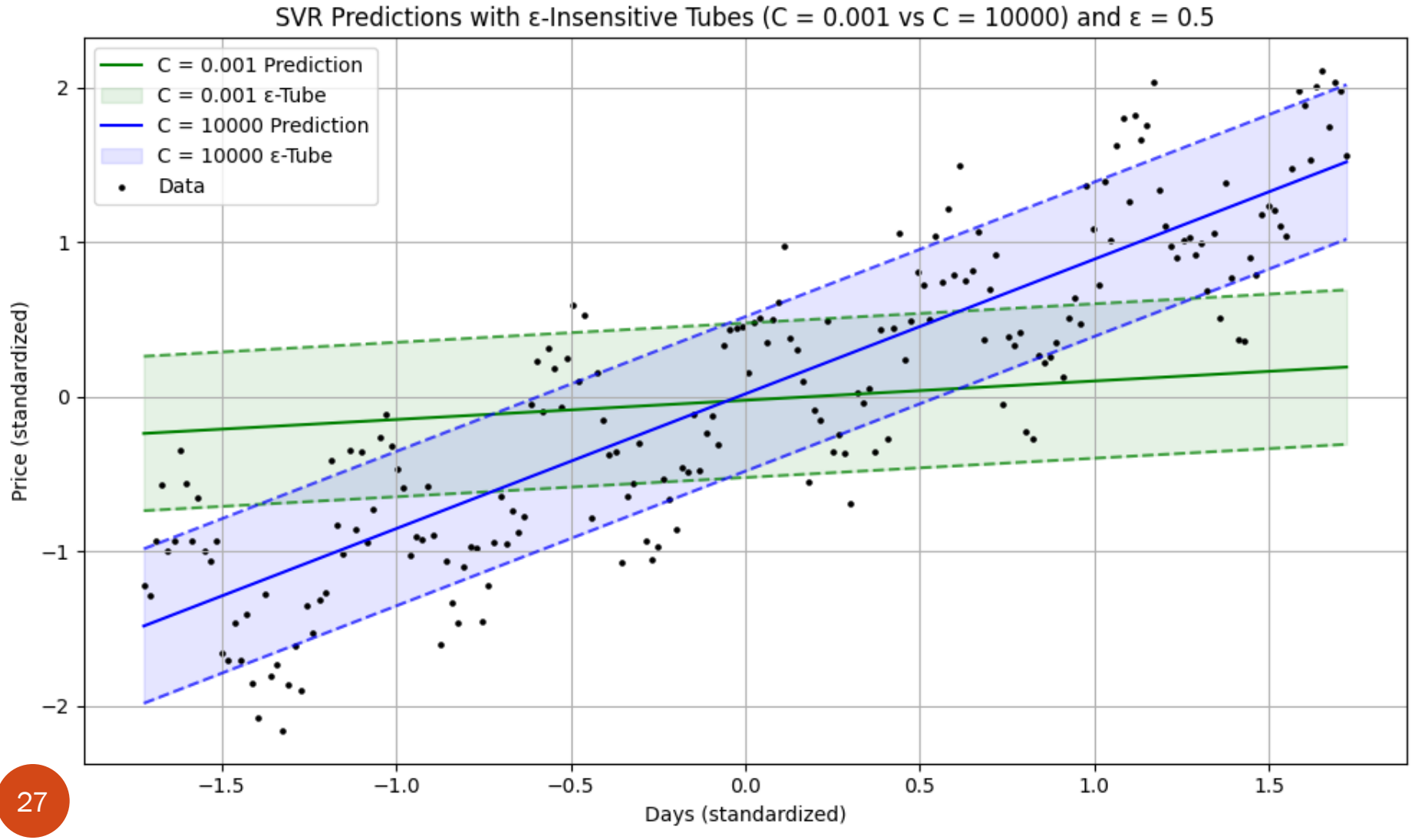


Support Vector Regression (SVR)

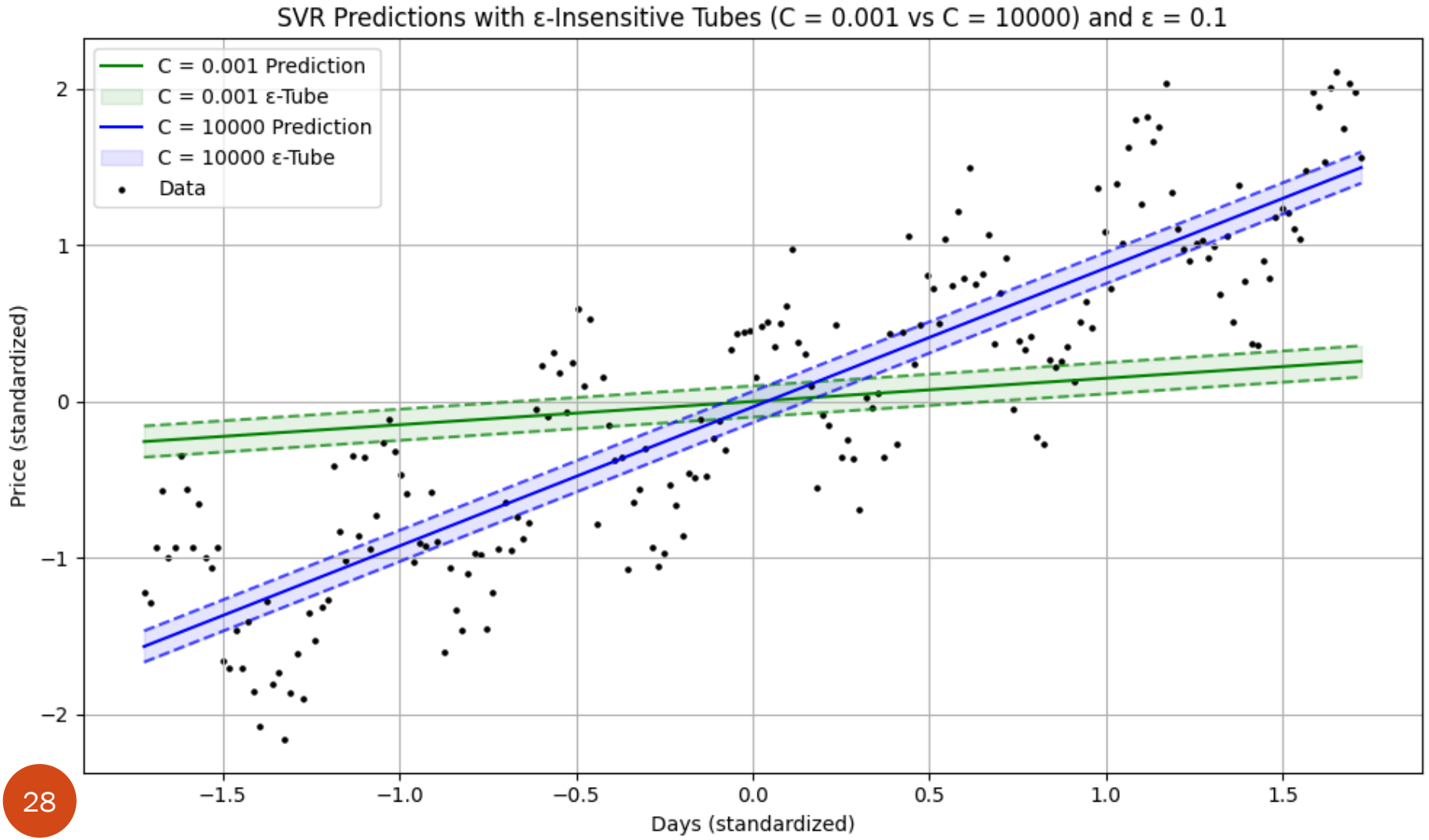
SVR (Support Vector Regression) tries to **fit a function within a tolerance margin (ε)** and **penalizes deviations beyond ε** — while keeping the function as flat as possible.

Symbol	Role
W	Defines model (slope of regression)
b	Bias/intercept
ϵ	Tolerance margin (no error counted here)
ξ_i	Error **above** the ε-tube
ξ_{i*}	Error **below** the ε-tube
C	Penalty for violating ε-tube

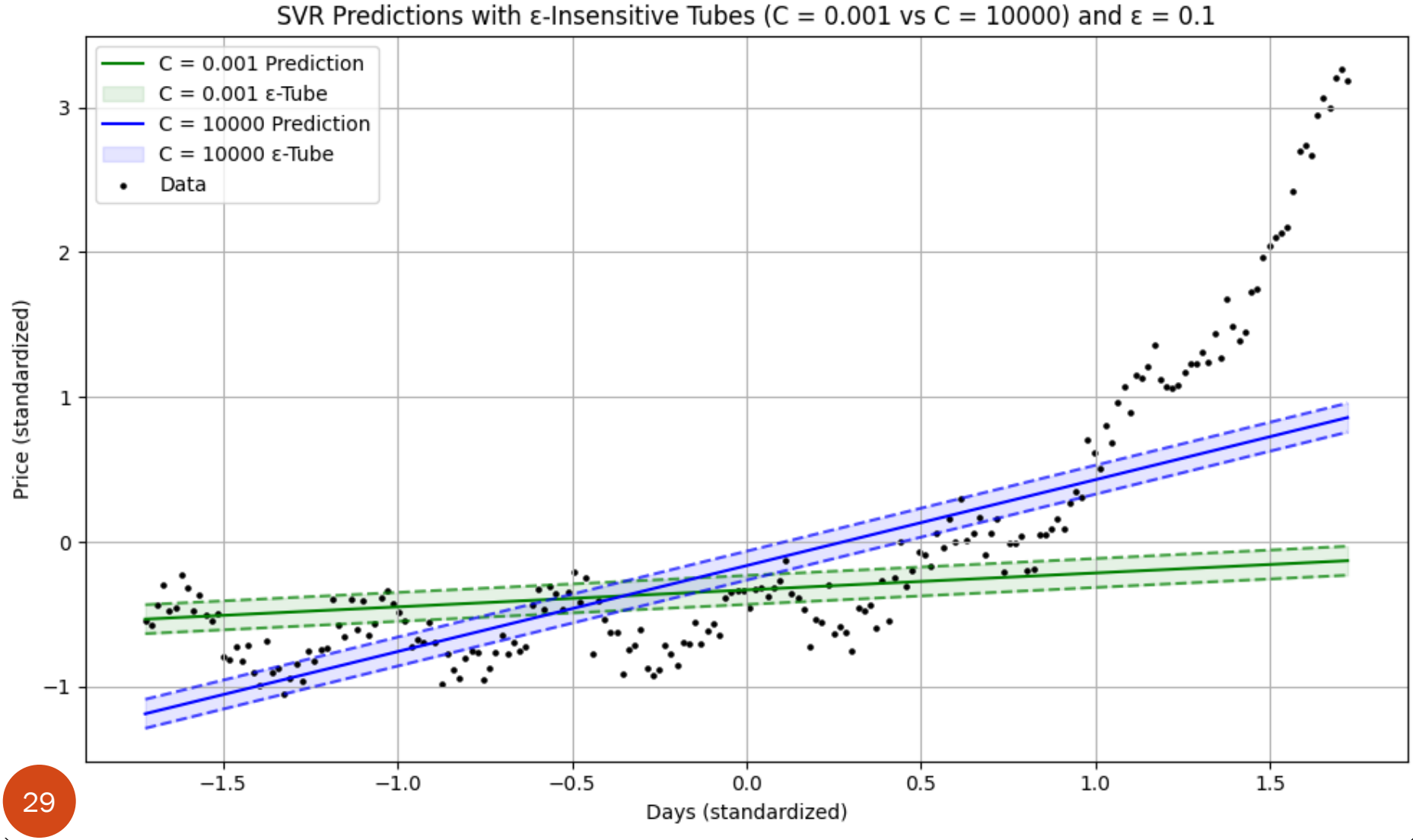
SVR: Effect of Epsilon



SVR: Effect of C



SVR


SVR Predictions with ε-Insensitive Tubes (C = 0.001 vs C = 10000) and ε = 0.1
C = 0.001 Prediction
C = 0.001 ε-Tube
C = 10000 Prediction
C = 10000 ε-Tube
Data
Price (standardized)
Days (standardized)
3
2
1
0
−1
−1.5
−1.0
−0.5
0.0
0.5
1.0
1.5

Kernel Method: Training

1. Collect the input data (e.g., days) and corresponding target values (e.g., stock prices).
2. Standardize both the input features and target values to ensure consistent scaling.
3. Create synthetic features:

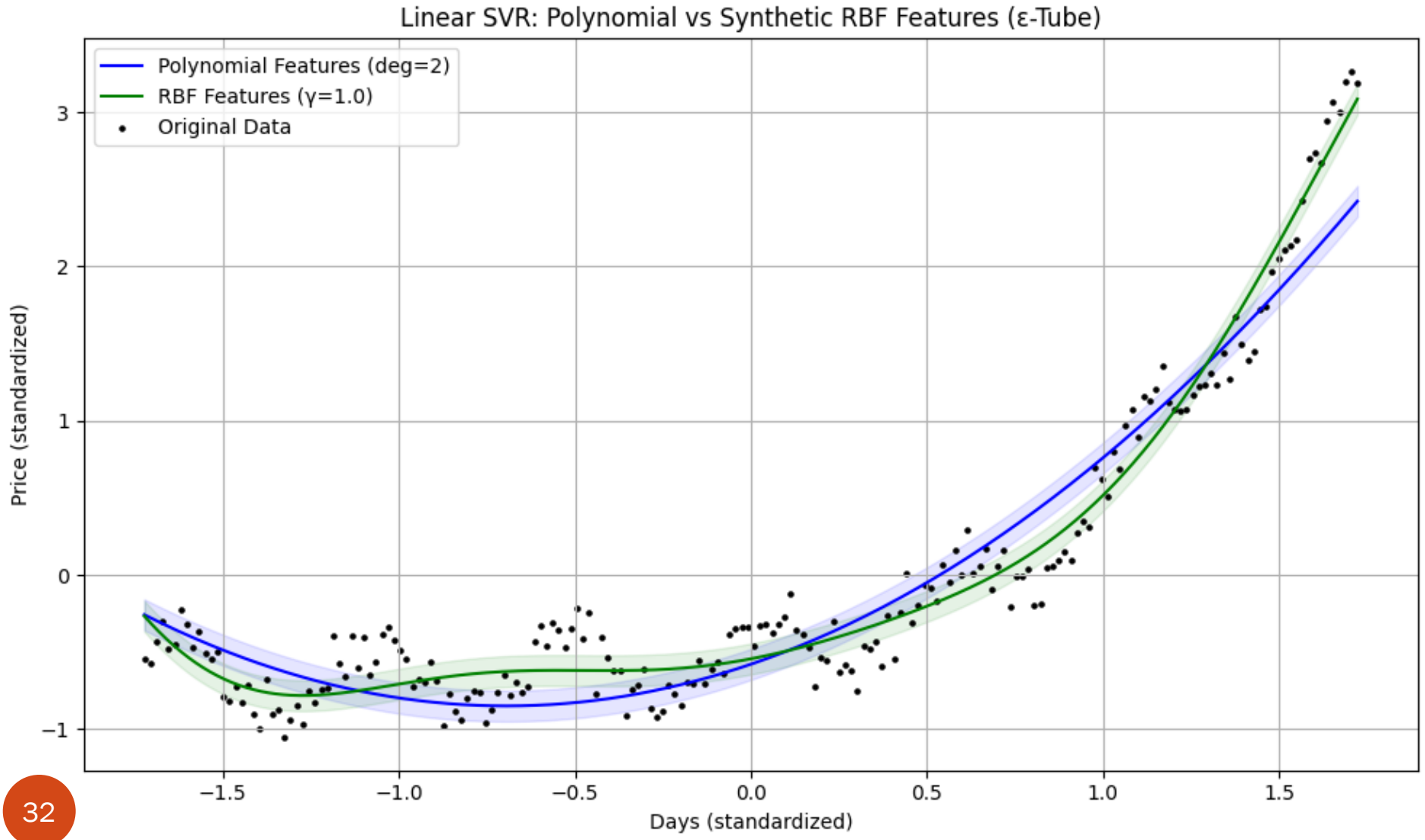
- For polynomial SVR, generate second-degree polynomial features from the standardized input.
- For RBF-based SVR, compute the RBF similarity between each input and all training inputs using a fixed gamma value.

4. Train a linear SVR model on each set of synthetic features using the same hyperparameters (e.g., C and ε).

Kernel Method: Testing

1. When a new test input is received, standardize it using the same scaler from training.
2. Transform the test input using the same feature generator used during training:
3. Use the same polynomial transformation or RBF similarity method.
4. Apply the trained SVR model to the transformed test input to obtain the predicted output.
5. Inverse transform the prediction back to the original scale if needed.

SVR: Synthetic features



Polynomial Features

For a single standardized input feature x, the degree-2 polynomial features are $\varphi(x) = [x, x^2]$

Linear SVR on degree-2 polynomial features operates in a 2D feature space. i.e., Axis 1: x (linear term)Axis 2: x^2(quadratic term)

$$w_1 x + w_2 x^2 + b$$

If there are multiple input features, it includes all combinations like $\varphi(x) = [\dots, x_1^2, x_1 x_2, x_1 x_3, \dots]$

Linear SVR operates in that *n* feature space

$$\dots + w_q x^2 + w_p x_1 x_2 + w_r x_1 x_3 \dots + b$$

RBF Features

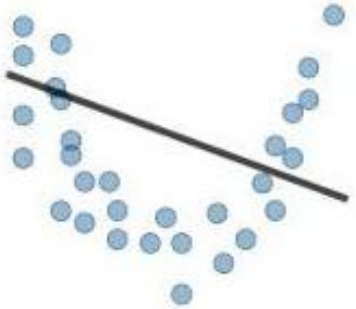
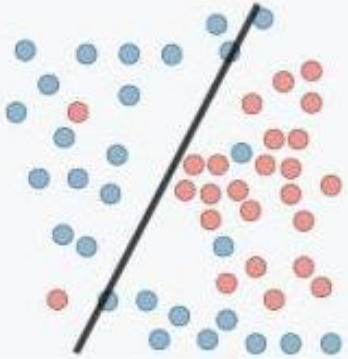
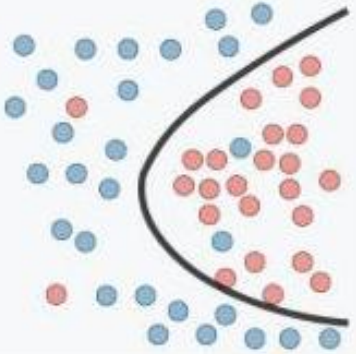
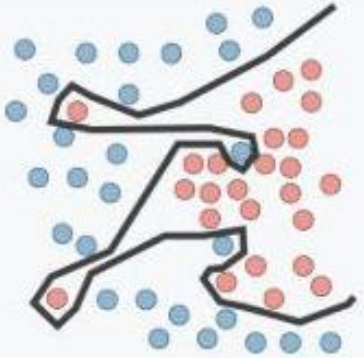
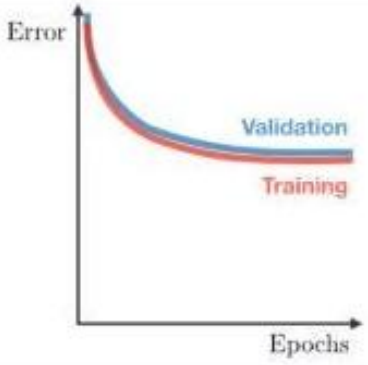
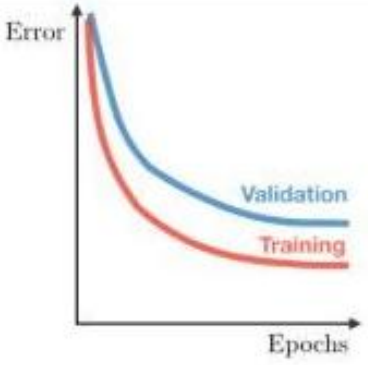
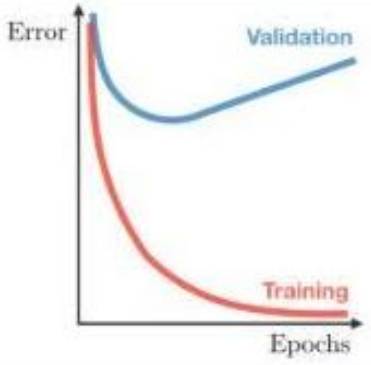
Given an input x, and other samples, $x_1, x_2, \ldots x_n$ the **RBF feature vector** is:

$$\varphi(x) = \left[e^{-\gamma\|x-x_1\|^2}, e^{-\gamma\|x-x_2\|^2}, \ldots, e^{-\gamma\|x-x_n\|^2}\right]$$

So, if you have 200 training points, then each $\phi(x)$ is a 200-dimensional vector. The SVR then learns a linear function in this 200-dimensional feature space.

Even if the sample x is e.g., 3D vector, the Euclidean distance manages the complexities.

A test input: x=[1,2,3] A training point: x_j=[4,0,2] γ=1.
Then: $\|x-x_j\|^2=(1-4)^2+(2-0)^2+(3-2)^2=9+4+1=14$
$\phi_j(x)=\exp(-1\cdot 14)=e^{-14}\approx 8.3\times 10^{-7}$

	Underfitting	Just right	Overfitting
Symptoms	• High training error • Training error close to test error • High bias	• Training error slightly lower than test error	• Very low training error • Training error much lower than test error • High variance
Regression illustration			
Classification illustration			
Deep learning illustration			

L1 (Lasso Logistic Regression)

Recall the Model

$$\hat{y} = w_n x^n + w_{n-1} x^{n-1} + \cdots + w_2 x^2 + w_1 x^1 + w_0$$

- Adds penalty on **absolute value** of coefficients

New Loss

$$J = Loss + \lambda \sum_{i=0}^{n} |w_i|$$

- Encourages sparsity: drives some $w_j = 0$
- Performs feature selection

L1 Regularization: Example

x1 (Signal)	x2 (Noise)	Pass (1/0)
2	100	0
4	90	0
6	105	1
8	95	1
10	85	1

- L1 may drive the weight for x_2 to zero
- Effectively removes irrelevant features

L2 (Ridge Logistic Regression)

Recall the Model

$$\hat{y} = w_n x^n + w_{n-1} x^{n-1} + \cdots + w_2 x^2 + w_1 x^1 + w_0$$

- Adds penalty on **Squared magnitude** of coefficients

New Loss

$$J = Loss + \lambda \sum_{i=0}^{n} w_i^2$$

- Keeps all features, but shrinks their influence
- Helps with multicollinearity

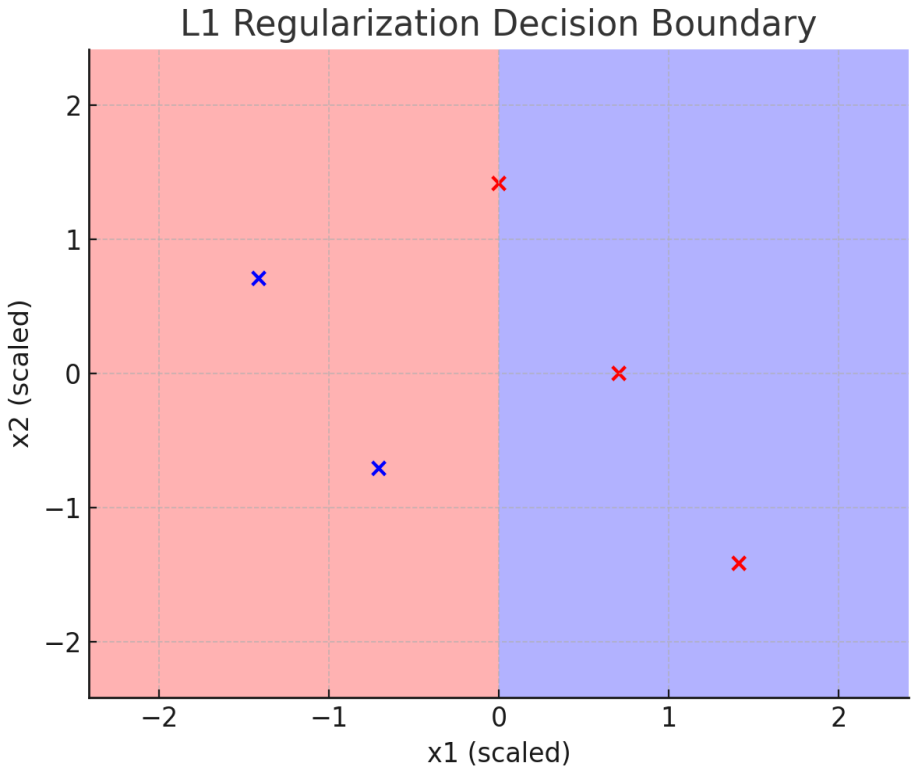
L2 Regression: Example

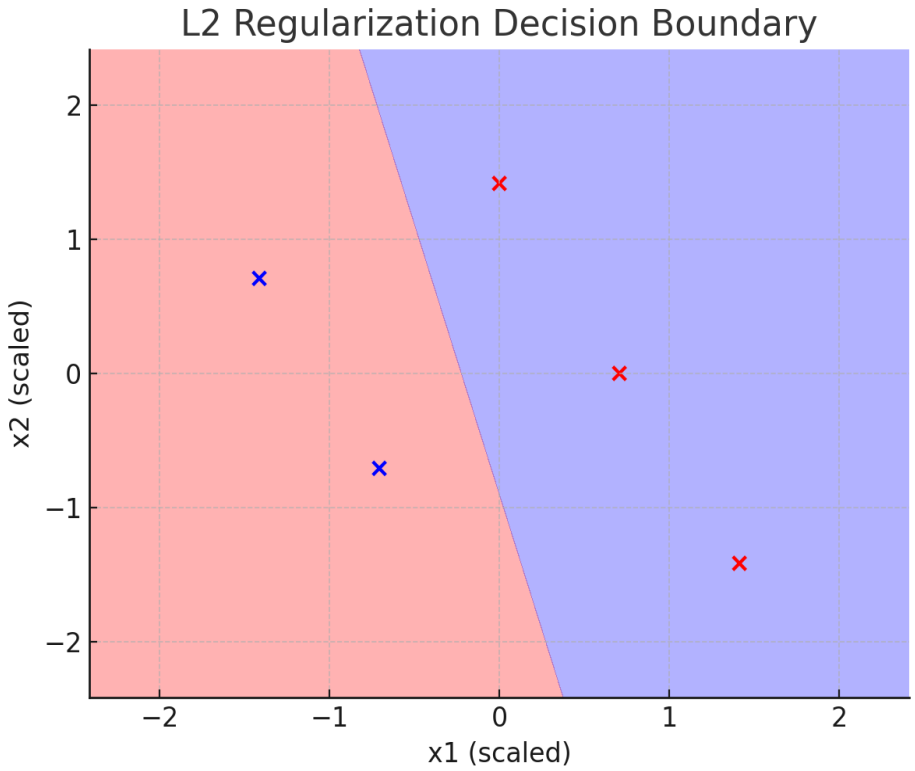
x1 (Signal)	x2 (Noise)	Pass (1/0)
2	100	0
4	90	0
6	105	1
8	95	1
10	85	1

- The penalty shrinks coefficients but does not force them to 0
- So, even the noise feature x_2 may get a small non-zero weight

[2.1, 0.15]

L1 vs L2





L1 tends to align the boundary mostly along informative feature (feature selection)

L2 blends all features by shrinking irrelevant ones but keeping them resulting in a more nuanced boundary

Regularization

Regularization techniques are used to prevent overfitting by adding additional information or constraints to the model. These techniques help to ensure that the model generalizes well to new, unseen data.

- **L1 Regularization (Lasso)**: Adds the absolute value of the magnitude of coefficients as a penalty term to the loss function. Encourages sparsity.
- **L2 Regularization (Ridge)**: Adds the squared magnitude of coefficients as a penalty term. Encourages smaller coefficients.
- **Early Stopping**: Stops training when the model's performance on a validation set starts to degrade.

Thank You All Very Much