EE69210: Machine Learning for Signal Processing Laboratory
Department of Electrical Engineering, Indian Institute of Technology, Kharagpur

Regression and Classification

Anirvan Krishna | Roll no. 21EE38002

Keywords: Support Vector Machines, Least Square Regression, Convex Optimization

Grading Rubric

	Tick the best applicable per row			Points
	Below Expectations	Lacking in Some	Meets all Expectations	
Completeness of the report				
Organization of the report (5 pts)				
Quality of figures (5 pts)				
SVM-1 (20 pts)				
SVM-2 (20 pts)				
Regression-1 (40 pts)				
Regression-2 (20 pts)				
			TOTAL (100 pts)	

1. Linear classification with support vector machines (SVMs)

Consider the support vector machine (SVM) classification problem. Labeled dataset $\{\mathbf{x}_i, y_i\}_{i\in[N]}$ where each $\mathbf{x}_i \in \mathbb{R}^n$ is associated with a label $y_i \in \{1,-1\}$ such that $y_i = 1$ if $\mathbf{x}_i \in \mathscr{A}$ and $y_i = -1$ if $\mathbf{x}_i \in \mathscr{B}$ is given.

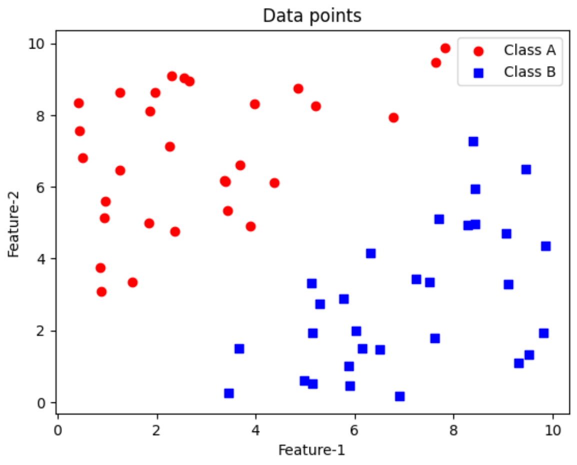


Fig. 1. Representation of data points in linear classification case

1.1 Formulation of primal problem

Given a dataset, $(\mathbf{X},\mathbf{y})$ where $\mathbf{X} \in \mathbb{R}^{n\times d}$ and $\mathbf{y} \in \{-1,1\}^n$, the primal SVM optimization problem can be formulated as:

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

subject to:

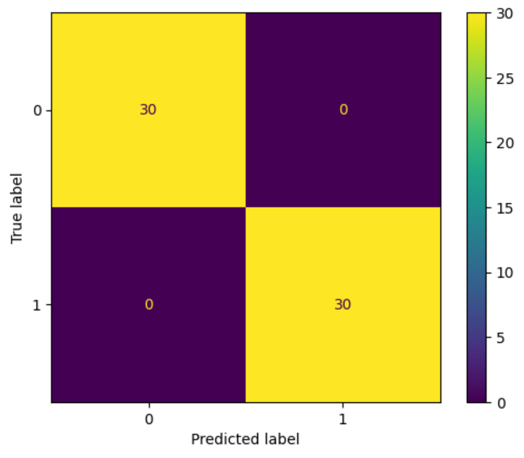
$$y_i(\mathbf{w}^\top \mathbf{x}_i + b) \geq 1-\xi_i, \quad \xi_i \geq 0, \quad i = 1,\ldots,n \tag{2}$$

where:

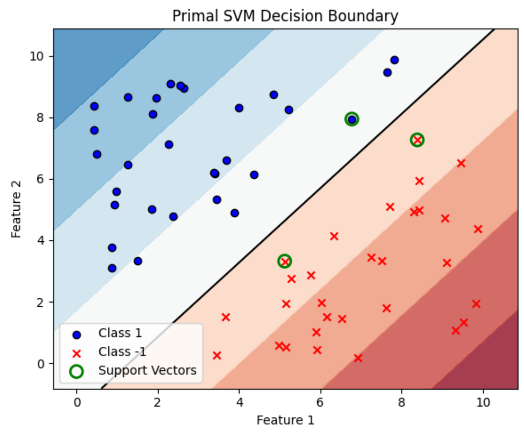
- $\mathbf{w}$ is the weight vector.
- b is the bias term.
- ξ_i are the slack variables that allow for some misclassification in the case of non-linearly separable data.
- $C$is the regularization parameter that controls the trade-off between maximizing the margin and minimizing the classification error.

The computed weights, bias, and support vectors are:

$$\begin{aligned} \text{Weights:} \quad & [-0.923, 0.763] \\ \text{Bias:} \quad & 1.198 \\ \text{Support Vectors:} \quad & \begin{bmatrix} 6.775 & 7.939 \\ 5.124 & 3.318 \\ 8.383 & 7.262 \end{bmatrix} \end{aligned}$$



(a) Confusion matrix



(b) Decision boundary

Fig. 2. Confusion matrix and decision boundaries for primal problem solution of linear classification

Class	Precision	Recall	F1-score	Support
-1.0	1.00	1.00	1.00	30
1.0	1.00	1.00	1.00	30
Accuracy			1.00	60
Macro avg	1.00	1.00	1.00	60
Weighted avg	1.00	1.00	1.00	60

Table 1. Classification report for the primal SVM model.

1.2 Formulation of dual problem

In this section, we will implement the Dual SVM formulation using convex optimization techniques. The dual SVM aims to find the optimal hyperplane that separates the data points of two classes with the maximum margin by solving the dual optimization problem.

1.2.1 Objective Function

Given a dataset $(\mathbf{X}, \mathbf{y})$ where $\mathbf{X} \in \mathbb{R}^{n \times d}$ and $\mathbf{y} \in \{-1, 1\}^n$, the dual SVM optimization problem can be formulated as:

$$\max_{\alpha} \quad \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} \alpha_i \alpha_j y_i y_j \mathbf{x}_i^\top \mathbf{x}_j \tag{3}$$

subject to:

$$\sum_{i=1}^{n} \alpha_i y_i = 0, \quad 0 \leq \alpha_i \leq C, \quad i = 1, \ldots, n \tag{4}$$

where:

- α are the Lagrange multipliers.
- C is the regularization parameter that controls the trade-off between maximizing the margin and minimizing the classification error.

1.2.2 Decision Function

The decision function for a new data point $\mathbf{x}$ is given by:

$$f(\mathbf{x}) = \sum_{i=1}^{n} \alpha_i y_i \mathbf{x}_i^\top \mathbf{x} + b \tag{5}$$

where b is the bias term, which can be computed using the support vectors.

1.2.3 Computing Primal Weight and Bias

Once we have the optimal Lagrange multipliers α from the dual problem, we can compute the primal weight vector $\mathbf{w}$ and the bias term b as follows:

- Primal Weight Vector $\mathbf{w}$:

$$\mathbf{w} = \sum_{i=1}^{n} \alpha_i y_i \mathbf{x}_i \tag{6}$$

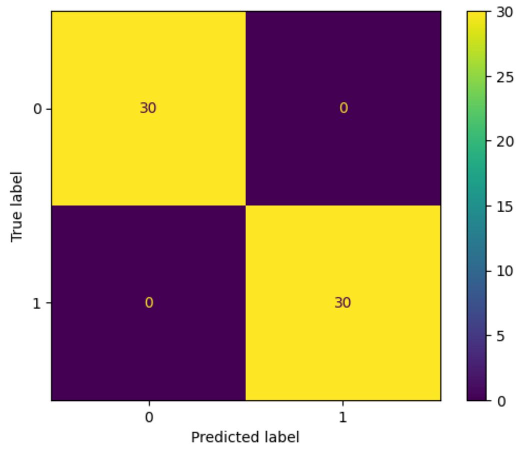
- Bias Term b:

$$b = y_i - \sum_{j=1}^{n} \alpha_j y_j \mathbf{x}_j^\top \mathbf{x}_i \tag{7}$$

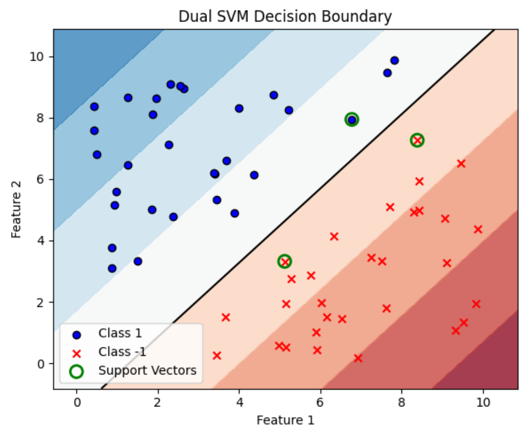
For numerical stability, it is common to average the bias term over all support vectors:

$$b = \frac{1}{|\mathscr{S}|} \sum_{i \in \mathscr{S}} \left(y_i - \sum_{j=1}^{n} \alpha_j y_j \mathbf{x}_j^\top \mathbf{x}_i \right) \tag{8}$$

where $\mathscr{S}$ is the set of support vectors.



(a) Confusion matrix



(b) Decision boundary

Fig. 3. Confusion matrix and decision boundaries for dual problem solution of linear classification

Class	**Precision**	**Recall**	**F1-score**	**Support**
-1.0	1.00	1.00	1.00	30
1.0	1.00	1.00	1.00	30
Accuracy			1.00	60
Macro avg	1.00	1.00	1.00	60
Weighted avg	1.00	1.00	1.00	60

Table 2. Classification report for the dual SVM model

We compare the weights and bias values obtained from the primal and dual SVM formulations:

$$\begin{aligned} \text{Primal Weights:} &\quad [-0.92288282, 0.76271838] \\ \text{Dual Weights:} &\quad [-0.92288126, 0.76271797] \\ \text{Primal Bias:} &\quad 1.1976144708333099 \\ \text{Dual Bias:} &\quad 1.1976076276102534 \end{aligned}$$

We compute the difference in weights and bias values.

$$\begin{aligned} \text{Weight Difference Norm:} &\quad 1.61 \times 10^{-6} \\ \text{Bias Difference Norm:} &\quad 6.84 \times 10^{-6} \end{aligned}$$

This confirms that the solutions obtained from both formulations are nearly identical, validating the correctness of the dual optimization approach.

2. Non-Linear Classification with Kernelized SVMs

Kernelized Support Vector Machines (SVMs) extend the concept of SVMs to non-linear classification problems by using kernel functions. These functions implicitly map the input features into a higher-dimensional space where a linear separation is possible.

2.1 Objective Function

The dual SVM optimization problem with a kernel function $K(\mathbf{x}_i, \mathbf{x}_j)$ is given by:

$$\max_{\alpha} \quad \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} \alpha_i \alpha_j y_i y_j K(\mathbf{x}_i, \mathbf{x}_j) \tag{9}$$

subject to:

$$\sum_{i=1}^{n} \alpha_i y_i = 0, \quad 0 \leq \alpha_i \leq C, \quad i = 1, \ldots, n \tag{10}$$

2.2 Kernel Functions

Common kernel functions include:

- **Linear Kernel**: $K(\mathbf{x}_i, \mathbf{x}_j) = \mathbf{x}_i^\top \mathbf{x}_j$
- **Gaussian (RBF) Kernel**: $K(\mathbf{x}_i, \mathbf{x}_j) = \exp\left(-\frac{\|\mathbf{x}_i - \mathbf{x}_j\|^2}{2\sigma^2}\right)$

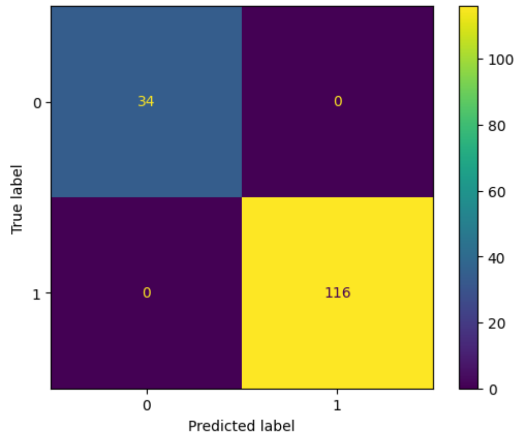
2.3 Decision Function

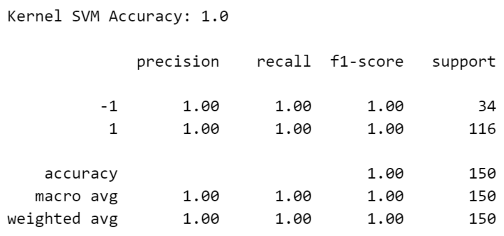
$$f(\mathbf{x}) = \sum_{i=1}^{n} \alpha_i y_i K(\mathbf{x}_i, \mathbf{x}) + b \tag{11}$$

The bias term b is computed similarly as in the linear case, using support vectors.

$$b = \frac{1}{|\mathscr{S}|} \sum_{i \in \mathscr{S}} \left(y_i - \sum_{j=1}^{n} \alpha_j y_j K(\mathbf{x}_j, \mathbf{x}_i) \right) \tag{12}$$

where $\mathscr{S}$ is the set of support vectors.





```
Kernel SVM Accuracy: 1.0

              precision    recall  f1-score   support

          -1       1.00      1.00      1.00        34
           1       1.00      1.00      1.00       116

    accuracy                           1.00       150
   macro avg       1.00      1.00      1.00       150
weighted avg       1.00      1.00      1.00       150
```

Fig. 5. Classification Report for Kernel SVMs

Fig. 4. (a) Confusion Matrix

‘

3. Least-Squares Polynomial Regression

3.1 Feature Map Definition

Given a dataset with 100 samples $\{(x_i, y_i)\}_{i=1}^{N}$, where:

- $\mathbf{x_i} \in \mathbb{R}^3$
- $y_i \in \mathbb{R}$
- y_i is a polynomial of degree at most 2 in x

We define a feature map $\phi(x)$ that includes all polynomial terms up to degree 2:

$$\phi(x) = \left[1, x_1^{(1)}, x_1^{(2)}, x_1^{(3)}, (x_1^{(1)})^2, (x_1^{(2)})^2, (x_1^{(3)})^2, x_1^{(1)} x_1^{(2)}, x_1^{(1)} x_1^{(3)}, x_1^{(2)} x_1^{(3)}\right]^\top \tag{13}$$

Feature Dimension: $\phi(x) \in \mathbb{R}^{10}$.

3.2 Least Squares Problem Formulation

Assume the polynomial function:

$$y_i = \mathbf{w}^\top \phi(\mathbf{x_i}) + \varepsilon_i \tag{14}$$

where:

- $\mathbf{w} \in \mathbb{R}^{10}$ (decision variable)
- ε_i represents noise

To estimate $\mathbf{w}$, we minimize the least squares error:

$$\min_{\mathbf{w}} \sum_{i=1}^{N} \left(y_i - \phi(x_i)^\top \mathbf{w} \right)^2 \tag{15}$$

Matrix form:

$$\min_{\mathbf{w}} \|\mathbf{y} - \Phi\mathbf{w}\|^2 \tag{16}$$

where:

- $\Phi \in \mathbb{R}^{N\times 10}$ (design matrix with rows $\phi(x_i)^\top$)
- $\mathbf{y} \in \mathbb{R}^{N}$ (output vector)

Cost function:

$$J(\mathbf{w}) = \frac{1}{N}\sum_{i=1}^{N}(y_i - \phi(x_i)^\top \mathbf{w})^2 = \frac{1}{N}\|\mathbf{y} - \Phi\mathbf{w}\|^2 \tag{17}$$

The gradient is given by:

$$\nabla J(\mathbf{w}) = -\frac{2}{N}\Phi^\top (y - \Phi\mathbf{w}) \tag{18}$$

3.3 Solution from the Normal Equation

The optimal solution for $\mathbf{w}$ in the least squares sense can be obtained by solving the normal equation:

$$\Phi^\top \Phi \mathbf{w} = \Phi^\top \mathbf{y} \tag{19}$$

Solving for $\mathbf{w}$:

$$\mathbf{w} = (\Phi^\top \Phi)^{-1} \Phi^\top \mathbf{y} \tag{20}$$

provided that $\Phi^\top \Phi$ is invertible. This closed-form solution minimizes the squared error directly.

3.4 Optimization Algorithms

3.4.1 Gradient Descent (GD)

$$\mathbf{w}_{t+1} = \mathbf{w}_t - \eta \nabla J(\mathbf{w}_t) \tag{21}$$

3.4.2 Accelerated Gradient Descent (AGD) with Momentum

We use momentum to accelerate convergence:

$$\mathbf{v}_{t+1} = \beta \mathbf{v}_t - \eta \nabla J(\mathbf{w}_t) \tag{22}$$

$$\mathbf{w}_{t+1} = \mathbf{w}_t + \mathbf{v}_{t+1} \tag{23}$$

where:

- $\beta \in (0,1)$ is the momentum parameter (e.g., 0.9)
- η is the learning rate

3.4.3 Stochastic Gradient Descent (SGD)

Instead of computing the full gradient, we approximate it using a single sample:

$$\mathbf{w}_{t+1} = \mathbf{w}_t - \eta \nabla J_i(\mathbf{w}_t) \tag{24}$$

where:

$$\nabla J_i(\mathbf{w}_t) = -2\phi(x_i)(y_i - \phi(x_i)^\top \mathbf{w}_t) \tag{25}$$

for a randomly chosen i. This leads to faster updates but introduces variance in the gradient estimate.

3.5 Results and Observations

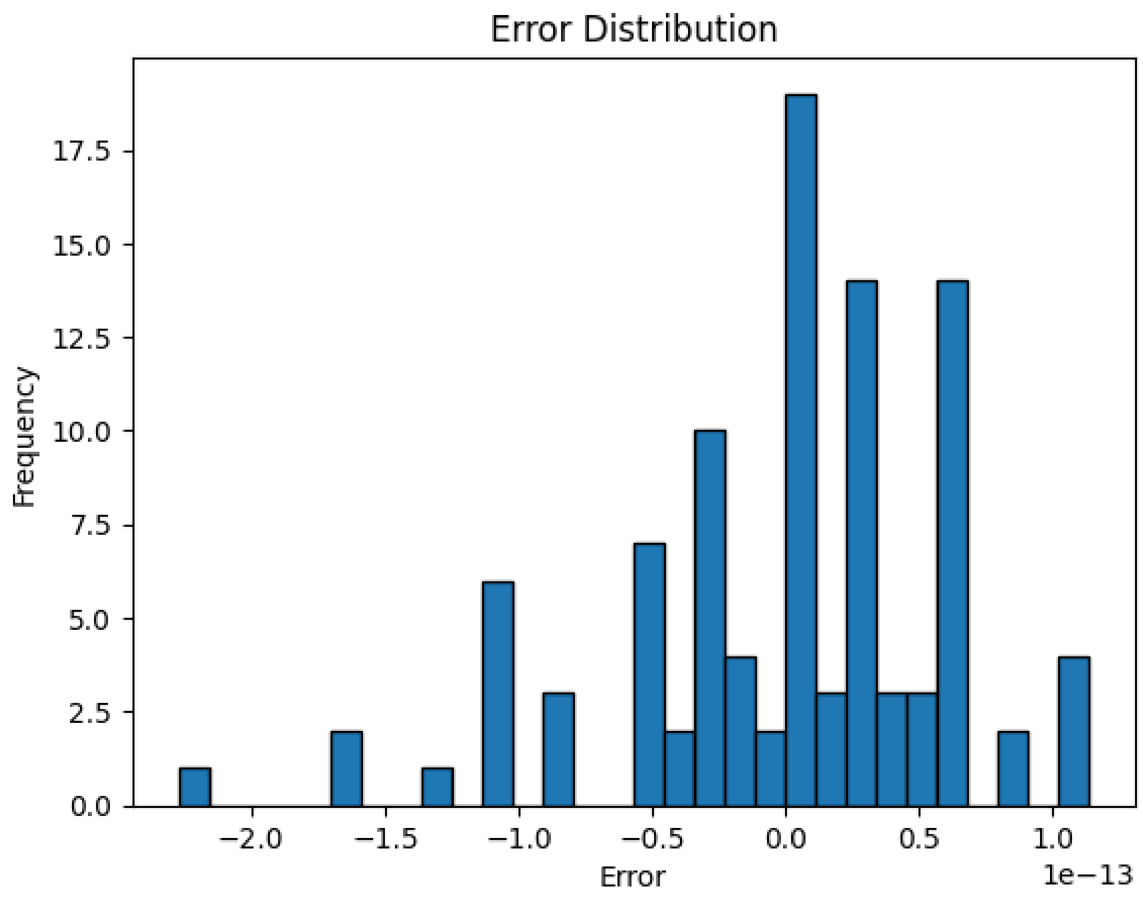


Fig. 6. Sample-wise error histogram for Least Square Regression solved using optimization solver

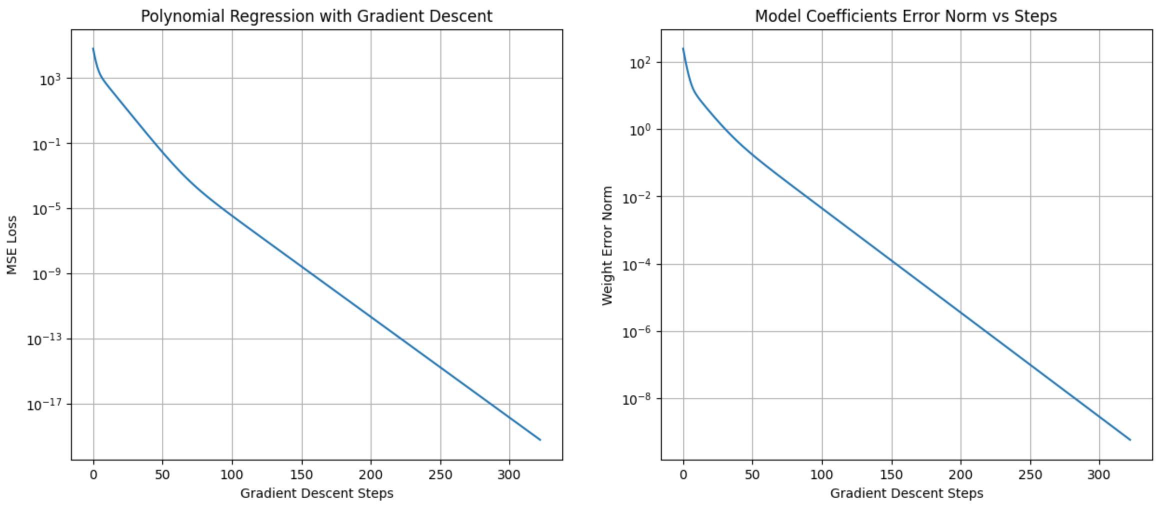


Fig. 7. Normal Gradient Descent Plots

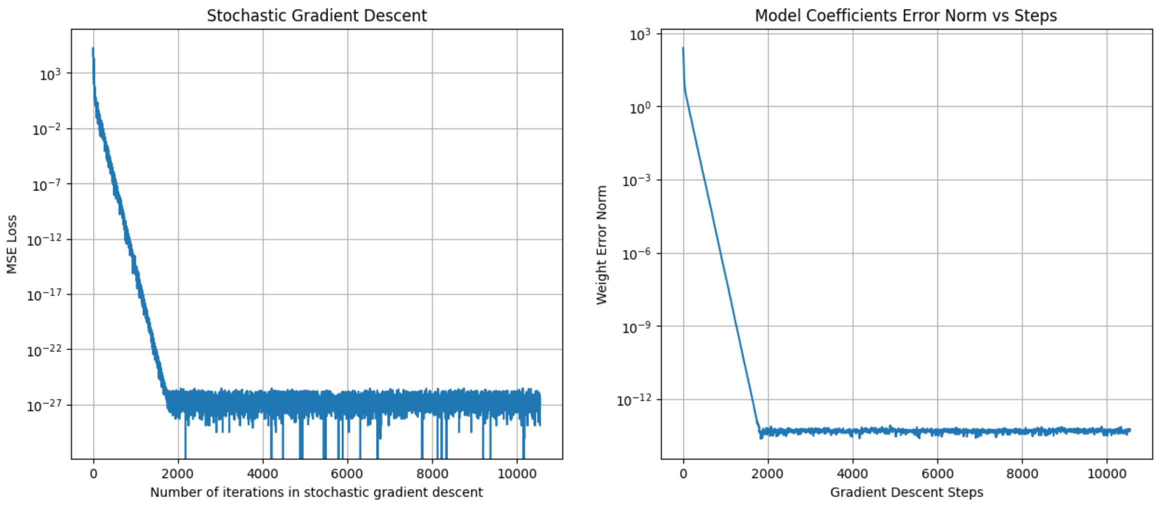


Fig. 8. Plots for Stochastic Gradient Descent

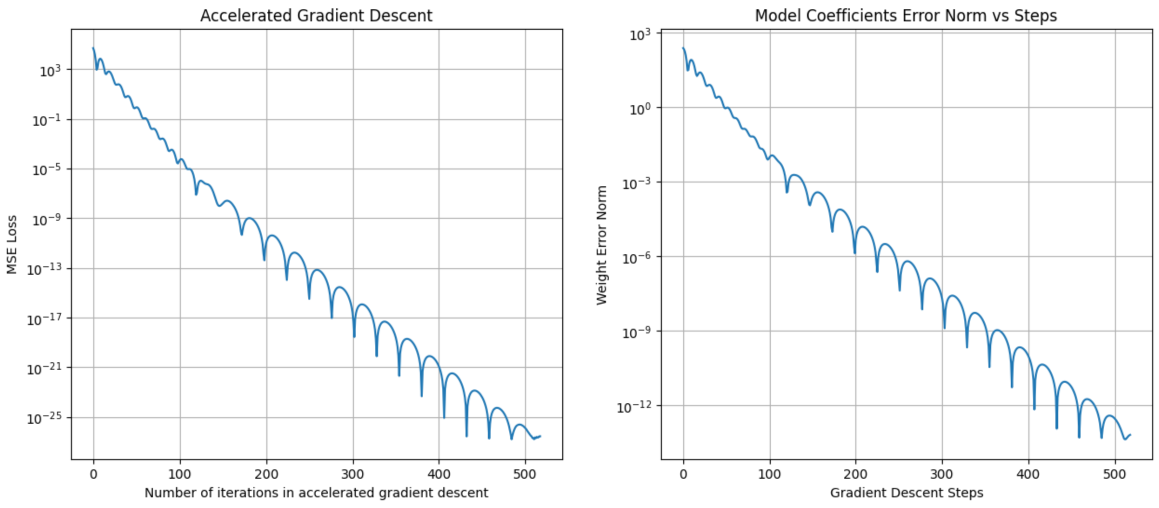


Fig. 9. Accelerated Gradient Descent

4. Distributed Least Squares Regression

4.1 Network Topology

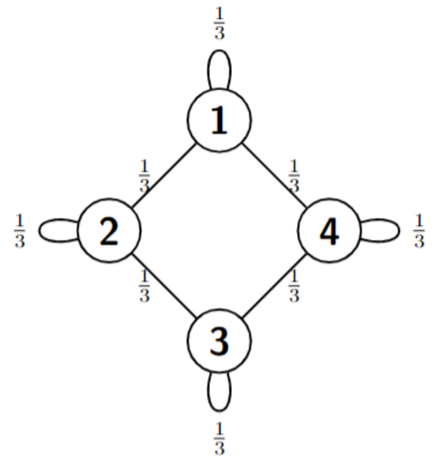


Fig. 10. Network topology for distributed least squares problem

- There are **four agents** $\{1,2,3,4\}$ communicating over a graph.
- The communication topology is defined by the given network graph (Figure 10).
- The weight a_{ij} between two neighboring agents is **1/3**.

4.2 Dataset Distribution

Each agent holds a subset of the dataset:

- **Agent 1**: First 20 data points
- **Agent 2**: Next 25 data points
- **Agent 3**: Next 30 data points
- **Agent 4**: Final 25 data points

4.3 Objective Function

The agents collectively minimize the least squares loss:

$$\min_{\mathbf{w}} f(\mathbf{w}) = \sum_{i=1}^{4} f_i(\mathbf{w}) \tag{26}$$

where the local loss function for agent i is:

$$f_i(\mathbf{w}) = \frac{1}{|\mathscr{D}_i|} \sum_{(\mathbf{x}_j, y_j) \in \mathscr{D}_i} \|y_j - \mathbf{x}_j^\top \mathbf{w}\|^2 \tag{27}$$

Here, $\mathbf{w}$ is the global regression parameter to be learned.

4.4 Distributed Gradient Descent Algorithm

Each agent updates its local model using a combination of **local gradient descent** and a **consensus averaging step**.

Algorithm Steps:

(1) Initialize Local Models: Each agent initializes its local model $\mathbf{w}_{i,0}$ randomly.

(2) Set Step Size: The learning rate is defined as:

$$\eta_t = \frac{1}{2\beta\sqrt{t}} \tag{28}$$

where β is the smoothness parameter.

(3) Local Gradient Descent Update:

$$\mathbf{w}_{i,t+1} = \mathbf{w}_{i,t} - \eta_t \nabla f_i(\mathbf{w}_{i,t}) \tag{29}$$

(4) Consensus Step:

$$\mathbf{w}_{i,t+1} := \mathbf{w}_{i,t+1} + \sum_{j\in\mathcal{N}(i)} a_{ij}(\mathbf{w}_{j,t} - \mathbf{w}_{i,t}) \tag{30}$$

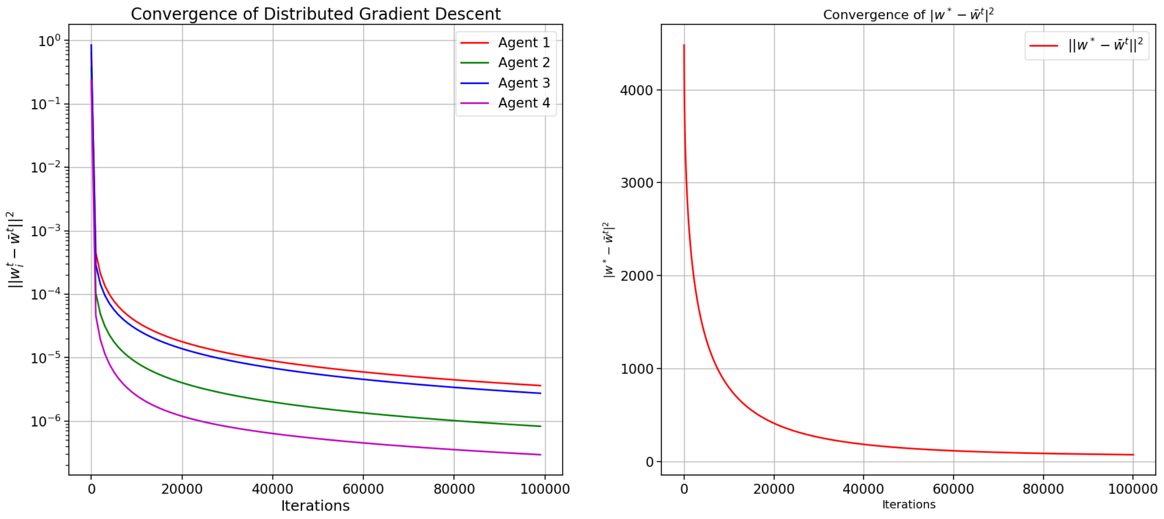


Fig. 11. Plots for distributed least squares regression