

Neural Networks

This project implements a neural network framework from first principles, beginning with linear regression and extending to more complex architectures.

LINEAR REGRESSION

Neural networks begin with linear regression. Let $x \in \mathbb{R}^d$ be our input vector with dimension d , $W \in \mathbb{R}^{p \times d}$ a weight matrix mapping to p outputs, and $b \in \mathbb{R}^p$ a bias vector. The linear transformation computes:

$$\hat{y} = Wx + b$$

This computation forms the core of our linear layer. Learning occurs through gradient descent, which requires computing how changes in weights affect the loss. Let \mathcal{L} be our loss function, mean square error. The weight gradients are:

$$\nabla_W \mathcal{L} = \frac{\partial \mathcal{L}}{\partial \hat{y}} \frac{\partial \hat{y}}{\partial W}$$

For a linear layer, $\frac{\partial \hat{y}}{\partial W_{ij}} = x_j$ for output \hat{y}_i , which in matrix form is $\frac{\partial \hat{y}}{\partial W} = x^T$.

For MSE, defined as:

$$\mathcal{L}(y, \hat{y}) = \frac{1}{n} \sum_{i=0}^n (y_i - \hat{y}_i)^2$$

This gradient is:

$$\frac{\partial \mathcal{L}}{\partial \hat{y}} = -\frac{2}{n} (y - \hat{y})$$

Our implementation separates these concerns into distinct modules:

- A **Linear** module handling forward computation and gradient calculations ;
- A **MSELoss** module computing the loss and its gradient ;
- The base **Module** class defining interfaces for all layers, given by the handout.

The training process iteratively applies gradient descent:

Gradient descent for linear regression

Input: x examples, y labels, η , \mathcal{E}

Output: W, b

```

1  $W \leftarrow \mathbb{R}^{p \times d}, b \leftarrow \mathbb{R}^p$ 
2 For  $e \in \{1, \dots, \mathcal{E}\}$ 
3    $\hat{y} \leftarrow Wx + b$ 
4    $\mathcal{L} \leftarrow \frac{1}{n} \sum (y_i - \hat{y}_i)^2$ 
5    $\nabla_W \mathcal{L} \leftarrow -\frac{2}{n} (y - \hat{y}) x^T$ 
6    $\nabla_b \mathcal{L} \leftarrow -\frac{2}{n} (y - \hat{y})$ 
7    $W \leftarrow W - \eta \nabla_W \mathcal{L}$ 
8    $b \leftarrow b - \eta \nabla_b \mathcal{L}$ 
```

Results analysis. Fig. 1 shows our linear regression results. The loss curve displays the expected pattern: rapid initial decrease followed by gradual convergence. Within 200 epochs, most improvement occurs, with minimal gains thereafter. The bottom plots contrast the best and worst models. Both capture linear relationships but with different slopes. The data points cluster tightly around both lines, showing both models learned useful patterns performance differences. This variability stems from random initialization and dataset generation.

These results indicate our implementation correctly applies the gradient descent algorithm. The smooth convergence indicates proper gradient computation and parameter updates.

LINEAR AND NON-LINEAR CLASSIFICATION

Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua quaerat volup-

tatem. Ut enim aequae doleamus animo, cum corpore dolemus, fieri tamen permagna accessio potest, si aliquod aeternum et infinitum impendere malum nobis opinemur. Quod idem licet transferre in voluptatem, ut.

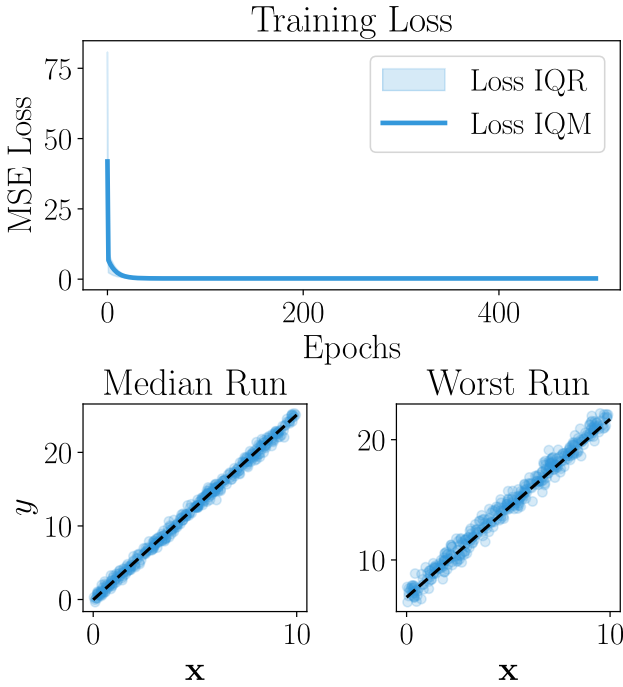


Fig. 1. – This figure displays training results from 100 linear regression trials ($n = 100$) with 200 samples per run ($\sigma = 200$), learning rate 0.01 ($\eta = 0.01$), and 1000 epochs ($\mathcal{E} = 1000$). The top plot shows the interquartile mean loss and Q1-Q3 range during training, while the bottom plots contrast the best and worst performing models from the ensemble.

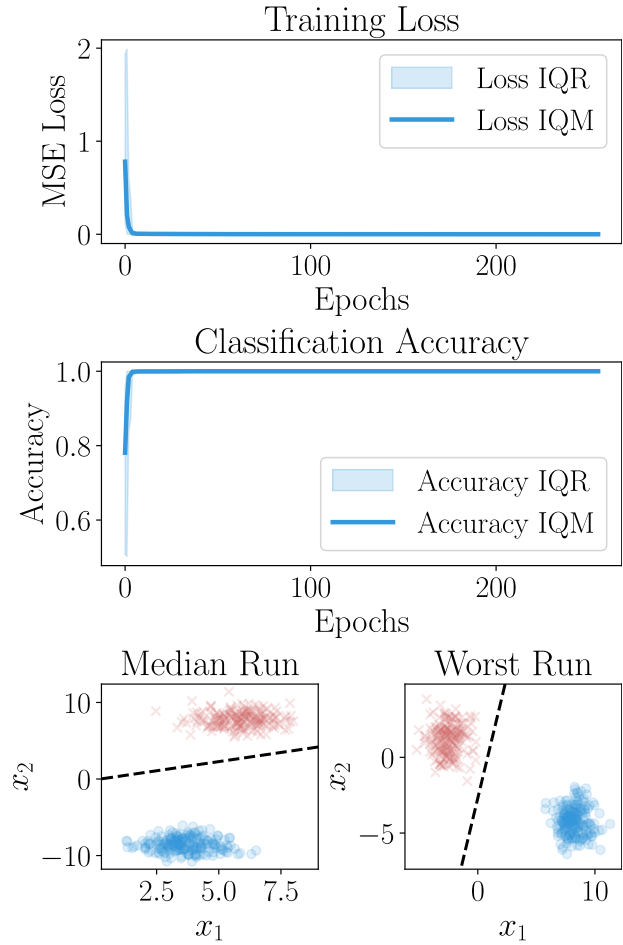


Fig. 2. –

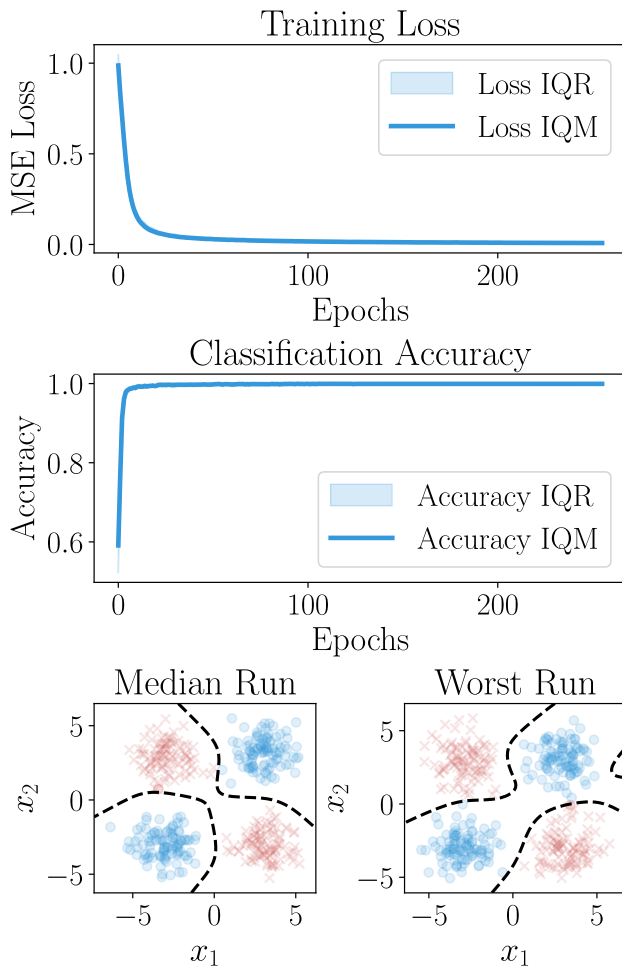


Fig. 3. –