Softmax Regression

Classify data into multiples classes (e.g., 3 classes: 0, 1, 2)
we want a function that gives class probabilities.

1. Linear Model

For every class j:

$$\boxed{z_j = W_j^T \cdot x + b_j}$$

Where:

- x = input features (2D here)
- W_j = weights for class j
- b_j = bias for class j

This gives raw scores (Logits), not probabilities.

---

2. Softmax Functions (converts scores → probabilities)

$$\boxed{\text{Softmax}(z_j) = \frac{e^{z_j}}{\sum_{k=1}^{C} e^{z_k}}}$$

Where:

- C = number of classes

✓ Output of Softmax:

- Every class gets a probability
- Probabilities sum to 1

---

3. Loss Function: Cross Entropy

Measures how wrong model ~~predict~~ predictions are.
For each examples:

$$\boxed{L = -\sum_{j=1}^{C} y_j \log(\hat{y}_j)}$$

Where:

- y_j = true ~~label~~ label (one-hot)
- $\hat{y}_j$ = predicted probability

Total loss:

$$\boxed{J = \frac{1}{N}\sum_{i=1}^{N} L_i}$$

Lower loss → better model.

---

4. Gradient Descent (Optimization)

We update weights in the direction that reduces loss.

Gradients:

$$\boxed{\frac{\partial J}{\partial W} = \frac{1}{N} X^T(\hat{Y} - Y)}$$

$$\boxed{\frac{\partial J}{\partial b} = \frac{1}{N}\sum_{i=1}^{N}(\hat{Y} - Y)}$$

Update Rules:

$$\boxed{W := W - \alpha \frac{\partial J}{\partial W}} \qquad \boxed{b := b - \alpha \frac{\partial J}{\partial b}}$$

Where:

α = learning rate (small step size)

---

5. Prediction

Pick the class with ~~lowest~~ highest probability:

$$\boxed{\hat{y} = \arg\max_j (\hat{y}_j)}$$

✓ Gives final class label (0, 1, 2 . .)

· Full Concept in 4 Lines

Step	Action	Math Output
1.	Linear function	Raw scores z
2.	Softmax	Probabilities
3.	Cross entropy	Loss
4.	Gradient descent	Improve weights

· Intuition

Model draws decision boundaries between groups.
Softmax gives confidence for each class.
Training improves boundaries until most points are classified correctly.