Great question! The reason why we sum the residuals (errors) when calculating the **bias gradient** but not the **weight gradient** comes from how each of these terms contributes to the error across all data points.

Understanding the Bias Gradient (`b_grad`):

- The **bias** term is a scalar, and it affects all predictions equally.
- For each sample in your dataset, the **bias** contributes the same amount to the prediction, so the error (residual) for all samples will have the same shift due to the bias.
- When calculating the gradient of the bias, we need to consider the sum of all individual residuals because the bias is applied uniformly across all samples.
- Therefore, to update the bias, we sum the residuals, which gives us the **total error** over all samples. This total error is then scaled by the learning rate and the normalization factor.

Why We Use `np.sum(y_true - y_predicted)` for Bias:

The bias term is added to the predicted values for **all samples** uniformly. The impact of the bias on each individual prediction is independent of the sample's features. Hence, when calculating the gradient for the bias term, you want to aggregate the errors for all samples to compute how much to adjust the bias.

The gradient for the bias is essentially the **average** of the errors (residuals) across all samples:

$$b_{\text{grad}} = -\frac{2}{N}\sum_{i=1}^{N}(y_{\text{true}}^{(i)} - y_{\text{predicted}}^{(i)})$$

Where:

- N is the total number of samples,
- $y_{\text{true}}^{(i)}$ is the true value of the i-th sample,
- $y_{\text{predicted}}^{(i)}$ is the predicted value of the i-th sample.

Understanding the Weight Gradient (`w_grad`):

- **Weights** are associated with **specific features** in the dataset.
- For each weight, the gradient depends on the **feature values** and the corresponding **errors** (residuals).
- The gradient for the weight is calculated as the dot product of the **residuals** (the error for each data point) with the **features** of the data point.
- The reason we don't sum over the samples when calculating the weight gradient is because each weight is multiplied by the corresponding feature for each sample.

The weight gradient calculation is:

$$w_{\text{grad}} = -\frac{2}{N} X^T \cdot (y_{\text{true}} - y_{\text{predicted}})$$

Where:

- X^T is the transpose of the feature matrix X,
- $(y_{\text{true}} - y_{\text{predicted}})$ is the residuals (errors) vector.

Here, the dot product between X^T (which has dimensions `(n_features, n_samples)`) and $(y_{\text{true}} - y_{\text{predicted}})$ (which is a vector of size `(n_samples,)`) produces a gradient vector of size `(n_features,)`.

Key Difference:

- **Bias**: The bias term is **added to every sample equally**, so we sum the residuals across all samples to capture the total error that needs to be corrected.
- **Weights**: Each weight is linked to a specific feature, and the gradient for each weight depends on both the feature values and the residuals. Hence, the calculation uses a **dot product** instead of a sum because each feature influences the error differently for each sample.

Example with Python Code:

python Copy code

```
import numpy as np # Sample data X = np.array([[1, 2], [3, 4], [5, 6]])
# Feature matrix (3 samples, 2 features) y_true = np.array([7, 8, 9]) #
True target values y_predicted = np.array([6, 7, 8]) # Predicted target
values total_samples = len(X) # Weight gradient (w_grad) w_grad = -(2 /
total_samples) * (X.T.dot(y_true - y_predicted)) # Bias gradient
(b_grad) b_grad = -(2 / total_samples) * np.sum(y_true - y_predicted) #
Print gradients print("Weight gradient (w_grad):", w_grad) print("Bias
gradient (b_grad):", b_grad)
```

Output:

python Copy code

```
Weight gradient (w_grad): [ -2.66666667 -11.33333333] Bias gradient
(b_grad): -2.0
```

Summary:

- **Bias gradient**: We sum the residuals for all data points because the bias affects all predictions uniformly.
- **Weight gradient**: We use the dot product to account for how each weight interacts with the feature values, as each feature contributes differently to the prediction error.