Mean Squared Error (MSE)

Basic MSE Calculation

```
import numpy as np

def mse(predictions, targets):
    return np.mean((predictions - targets)**2)

y_pred = np.array([1.2, 2.3, 3.1])
y_true = np.array([1.0, 2.0, 3.0])

error = mse(y_pred, y_true)
print(f"MSE: {error:.4f}")

# Show individual errors
differences = y_pred - y_true
squared_errors = differences ** 2
print(f"\nDifferences: {differences}")
print(f"Squared errors: {squared_errors}")
print(f"Mean: {np.mean(squared_errors):.4f}")
```

MSE vs Other Metrics

```
from sklearn.metrics import (mean_squared_error,
                              mean_absolute_error,
                              r2_score)

y_true = np.array([3, -0.5, 2, 7])
y_pred = np.array([2.5, 0.0, 2, 8])

mse = mean_squared_error(y_true, y_pred)
rmse = np.sqrt(mse)
mae = mean_absolute_error(y_true, y_pred)
r2 = r2_score(y_true, y_pred)

print(f"MSE:  {mse:.4f}")
print(f"RMSE: {rmse:.4f}")
print(f"MAE:  {mae:.4f}")
print(f"R²:   {r2:.4f}")
```

MSE in Linear Regression

```
# Generate synthetic data
np.random.seed(42)
X = np.linspace(0, 10, 100)
y_true = 2 * X + 1 + np.random.randn(100) * 2

# Fit line
coefficients = np.polyfit(X, y_true, 1)
y_pred = np.polyval(coefficients, X)

# Calculate MSE
mse = np.mean((y_true - y_pred)**2)
rmse = np.sqrt(mse)

print(f"Fitted line: y = {coefficients[0]:.2f}x + {coefficients[1]:.2f}")
print(f"MSE: {mse:.4f}")
print(f"RMSE: {rmse:.4f}")
# Visualize
```

```
import matplotlib.pyplot as plt
plt.figure(figsize=(10, 6))
plt.scatter(X, y_true, alpha=0.5, label='Data')
plt.plot(X, y_pred, 'r-', linewidth=2, label='Fitted line')
plt.xlabel('X')
plt.ylabel('y')
plt.title(f'Linear Regression (MSE = {mse:.2f})')
plt.legend()
plt.grid(True, alpha=0.3)
plt.show()
```

Batch MSE Calculation

```
# Multiple predictions
predictions = np.array([[1.0, 2.0, 3.0],
                       [1.5, 2.5, 3.5],
                       [0.9, 1.8, 2.7]])

targets = np.array([[1.0, 2.0, 3.0],
                   [1.0, 2.0, 3.0],
                   [1.0, 2.0, 3.0]])

# MSE per sample
mse_per_sample = np.mean((predictions - targets)**2, axis=1)
print("MSE per sample:", mse_per_sample)

# Overall MSE
overall_mse = np.mean((predictions - targets)**2)
print(f"\nOverall MSE: {overall_mse:.4f}")

# MSE per feature
mse_per_feature = np.mean((predictions - targets)**2, axis=0)
print(f"MSE per feature: {mse_per_feature}")
```

MSE Gradient for Optimization

```
# MSE gradient for linear regression
def mse_gradient(X, y, weights):
    predictions = X @ weights
    errors = predictions - y
    gradient = 2 * X.T @ errors / len(y)
    return gradient

# Example
X = np.array([[1, 1], [1, 2], [1, 3], [1, 4]])
y = np.array([2, 4, 6, 8])
weights = np.array([0.0, 0.0])

gradient = mse_gradient(X, y, weights)
print("Gradient:", gradient)

# Gradient descent step
learning_rate = 0.01
weights = weights - learning_rate * gradient
print(f"Updated weights: {weights}")

# Calculate MSE
predictions = X @ weights
mse = np.mean((predictions - y)**2)
print(f"MSE after update: {mse:.4f}")
```