```python
import numpy as np
import matplotlib.pyplot as plt
from sklearn.datasets import fetch_openml
from sklearn.preprocessing import StandardScaler
from sklearn.model_selection import train_test_split


def ADMM(X_train, y_train, lambda_, rho, max_iter=100, tol=1e-4):
    d = X_train.shape[1]
    w, z, y = np.zeros(d), np.zeros(d), np.zeros(d)
    obj_values, w_z_diffs = [], []

    for iter_count in range(max_iter):
        # w-update with gradient descent
        for _ in range(40):
            y_pred = X_train @ w
            neg_pred = -y_train * y_pred
            sig_vals = sig_func(neg_pred)
            term1 = -(X_train.T @ (y_train * sig_vals)) / X_train.shape[0]
            term2 = rho * (w - z + y / rho)
            gradient = term1 + term2
            w = w - 0.1 * gradient

        z = (rho * w + y) / (rho + lambda_)
        y = y + rho * (w - z)

        current_loss = log_func(w, X_train, y_train)
        regularization = (lambda_/2) * np.linalg.norm(z)**2
        obj_values.append(current_loss + regularization)

        primal_residual = np.linalg.norm(w - z)
        w_z_diffs.append(primal_residual)

        if primal_residual < tol:
            break

    return w, obj_values, w_z_diffs

def log_func(w, X, y):
    return np.mean(np.log(1 + np.exp(-y * (X @ w))))

def sig_func(x):
    return 1 / (1 + np.exp(-x))

mnist_data = fetch_openml('mnist_784', version=1, as_frame=False)
features, labels = mnist_data.data, mnist_data.target.astype(int)

digit1, digit2 = 3, 8
selected_indices = (labels == digit1) | (labels == digit2)
features = features[selected_indices]
labels = labels[selected_indices]

binary_labels = np.where(labels == digit1, -1, 1)

scaler = StandardScaler()
scaled_features = scaler.fit_transform(features)

X_train, X_test, y_train, y_test = train_test_split(
    scaled_features, binary_labels, test_size=0.2, random_state=42
)

lambda_ = 0.1
rho = 1.0
model_weights, convergence_vals, residuals = ADMM(X_train, y_train, lambda_, rho)

fig = plt.figure(figsize=(12, 5))

ax1 = fig.add_subplot(1, 2, 1)
ax1.plot(convergence_vals)
ax1.set_yscale('log')
ax1.set_xlabel('Iteration')
ax1.set_ylabel('F(w, z)')
ax1.set_title('Objective Function')

ax2 = fig.add_subplot(1, 2, 2)
ax2.plot(residuals)
ax2.set_yscale('log')
ax2.set_xlabel('Iteration')
ax2.set_ylabel('||w - z||')
ax2.set_title('Convergence of w and z')

plt.tight_layout()
plt.show()

train_predictions = np.sign(X_train @ model_weights)
test_predictions = np.sign(X_test @ model_weights)

train_error = np.mean(train_predictions != y_train)
test_error = np.mean(test_predictions != y_test)

print(f"Training error: {train_error:.4f}")
print(f"Test error: {test_error:.4f}")
```