```python
import numpy as np
import data
import matplotlib.pyplot as plt

def re_lu(x):
    return np.maximum(0, x)


def softmax(x):
    exp_x_shifted = np.exp(x - np.max(x))
    exp_sums = np.sum(exp_x_shifted, axis=1).reshape((-1, 1))
    probs = exp_x_shifted / exp_sums
    return probs


def fcann2_train(X, Y_, shape=(-1, 3), max_iter=100_000, eta=0.05, lamda=0.001, print_frequency=100):
    N = len(X)
    n = len(X[0])
    C = np.max(Y_) + 1
    hidden = shape[1]

    W1 = np.random.randn(n, hidden)
    b1 = np.zeros((1, hidden))
    W2 = np.random.randn(hidden, C)
    b2 = np.zeros((1, C))

    Y_one_hot = data.class_to_onehot(Y_)

    for i in range(max_iter):
        # FORWARD
        # 1st layer
        s1 = X @ W1 + b1
        h1 = re_lu(s1)
        # second layer
        s2 = h1 @ W2 + b2
        h2 = re_lu(s2)
        # loss
        P = softmax(h2)

        if i % print_frequency == 0:
            loss = -1.0 / N * np.sum(np.log(P) * Y_one_hot)
            print(f"iteration {i} loss = {loss}")

        # BACKWARD
        dL_ds2 = P - Y_one_hot
        dL_dW2 = h1.T @ dL_ds2
        dL_db2 = np.sum(dL_ds2, axis=0)

        dL_ds1 = dL_ds2 @ W2.T
        dL_ds1[s1 <= 0] = 0
        dL_dW1 = dL_ds1.T @ X
        dL_db1 = np.sum(dL_ds1, axis=0)
        # ADJUSTING WEIGHTS AND BIASES
        W1 = W1 * (1 - eta * lamda) - 1 / N * eta * dL_dW1.T
        b1 -= 1 / N * eta * dL_db1
        W2 = W2 * (1 - eta * lamda) - 1 / N * eta * dL_dW2
        b2 -= 1 / N * eta * dL_db2

    return W1, b1, W2, b2


def fcann2_classify(X, W1, b1, W2, b2):
    s1 = X @ W1 + b1
    h1 = re_lu(s1)
    s2 = h1 @ W2 + b2
    return softmax(s2)
```

```
def decfun(W1, b1, W2, b2):
    return lambda X: fcann2_classify(X, W1, b1, W2, b2)[:, 1]


if __name__ == "__main__":
    np.random.seed(100)
    X, Y_ = data.sample_gmm_2d(6, 2, 10)
    W1, b1, W2, b2 = fcann2_train(X, Y_, (-1, 5), 10_000, 0.05, 0.001, 500)
    probs = fcann2_classify(X, W1, b1, W2, b2)
    Y = np.argmax(probs, axis=1)
    bounding_box = (np.min(X, axis=0), np.max(X, axis=0))
    data.graph_surface(decfun(W1, b1, W2, b2), bounding_box, offset=0.5)
    data.graph_data(X, Y_, Y)
    plt.show()
```