Pseudocode:

Initialize inputs:

x_in, y_in, y1_in, z_in (input arrays for data).

Set initial parameters w = [-2, 3, 1] (initial guess for the weights).

Define constants: maxit = 1000000, epsilon = 1.e-3.

Define objective function:

Compute z = w[0] + w[1] * x_in + w[2] * z_in.

Calculate z1 = log(1 + exp(-z)) and z2 = log(1 + exp(z)).

Compute the value of the objective function:

objective_value = dot(y_in, z1) + dot(y1_in, z2).

Define gradient function:

Compute hi = 1 / (1 + exp(-w[0] - w[1] * x_in - w[2] * z_in)).

Calculate the difference yh = hi - y_in.

Return gradient as [sum(yh), dot(yh, x_in), dot(yh, z_in)].

Define line search function:

Set beta = 0.1, stepsize = 1, trial = 100, tau = 0.5.

For each iteration (up to trial times):

Compute fx1 = objective_function(x).

Compute fx2 = objective_function(x - stepsize * gradient).

Calculate condition c = -beta * stepsize * dot(gradient, gradient).

If fx2 - fx1 <= c, break loop (valid step size found).

Else, reduce stepsize by multiplying by tau.

Return the final step size.

Optimization loop:

For each iteration (up to maxit times):

Compute gradient = gradient_function(w).

Calculate norm of the gradient b = norm(gradient).

If b < epsilon, break loop (convergence achieved).

Perform line search to compute stepsize.

Update w = w - stepsize * gradient.

Print the iteration count and gradient norm.

After the loop:

Calculate the minimum objective function value:

minimum_value = objective_function(w).

Print the minimum value, location (w), and number of iterations (i).