In [61]:
```
# Useful starting lines
%matplotlib inline
import numpy as np
import matplotlib.pyplot as plt
%reload_ext autoreload
%autoreload 2
```

In [95]:
```
# Check the Python version
import sys
if sys.version.startswith("3."):
  print("You are running Python 3. Good job :)")
else:
  print("This notebook requires Python 3.\nIf you are using Google Colab, go to Runtime > Change runtime type and choose Pytho
```

```
You are running Python 3. Good job :)
```

In [ ]:
```
# Load the data
```

In [96]:
```
import datetime
from helpers import *

height, weight, gender = load_data(sub_sample=False, add_outlier=False)
x, mean_x, std_x = standardize(height)
b, A = build_model_data(x, weight)
```

In [97]:
```
print('Number of samples n = ', b.shape[0])
print('Dimension of each sample d = ', A.shape[1])
```

```
Number of samples n =  10000
Dimension of each sample d =  2
```

Least Squares Estimation

Least squares estimation is one of the fundamental machine learning algorithms. Given an $n \times d$ matrix A and a $n \times 1$ vector b, the goal is to find a vector $x \in \mathbb{R}^d$ which minimizes the objective function

$$f(x) = \frac{1}{2n}\sum_{i=1}^{n}(a_i^\top x - b_i)^2 = \frac{1}{2n}\|Ax - b\|^2$$

In this exercise, we will try to fit x using Least Squares Estimation.

One can see the function is L smooth with $L = \frac{1}{n}\|A^T A\| = \frac{1}{n}\|A\|^2$.

Computing the Objective Function

Fill in the `calculate_objective` function below:

In [53]:
```
def calculate_objective(Axmb):
    """Calculate the mean squared error for vector Axmb = Ax - b."""
    # ***************************************************
    # INSERT YOUR CODE HERE
    # TODO: compute mean squared error
    # ***************************************************
    # 1. square all the elements
    squared = Axmb ** 2
    # 2. calculate the sum of square
    sum_squared = np.sum(squared)
    # 3. eliminate the nuumber of element
    mse = sum_squared / len(Axmb)
    return mse
```

Compute smoothness constant L

To compute the spectral norm of A you can use np.linalg.norm(A, 2)

In [54]:
```
def calculate_L(b, A):
    """Calculate the smoothness constant for f"""
    # ***************************************************
    # INSERT YOUR CODE HERE
    # TODO: compute ||A.T*A||
```

```
    # ***************************************************
    ATA = A.T @ A
    # ***************************************************
    # INSERT YOUR CODE HERE
    # TODO: compute L = smoothness constant of f
    # ***************************************************
    L = np.linalg,norm(ATA, 2)
    return L
```

Gradient Descent

Please fill in the functions `compute_gradient` below:

In [55]:
```
def compute_gradient(b, A, x):
    """Compute the gradient."""
    # ***************************************************
    # INSERT YOUR CODE HERE
    # TODO: compute gradient and objective
    # ***************************************************
    Axmb = A @ x - b
    grad = A.T @ Axmb
    return grad, Axmb
```

Please fill in the functions `gradient_descent` below:

In [56]:
```
def gradient_descent(b, A, initial_x, max_iters, gamma):
    """Gradient descent algorithm."""
    # Define parameters to store x and objective func. values
    xs = [initial_x]
    objectives = []
    x = initial_x
    for n_iter in range(max_iters):
        # ***************************************************
        # INSERT YOUR CODE HERE
        # TODO: compute gradient and objective function
        # ***************************************************
        grad, Axmb = compute_gradient(b, A, x)
```

```
        obj = 0.5 * np.sum(Axmb ** 2)
        # ***************************************************
        # INSERT YOUR CODE HERE
        # TODO: update x by a gradient descent step
        # ***************************************************
        x = x - gamma * grad
        # store x and objective function value
        xs.append(x)
        objectives.append(obj)
        print("Gradient Descent({bi}/{ti}): objective={l}".format(
              bi=n_iter, ti=max_iters - 1, l=obj))

    return objectives, xs
```

Test your gradient descent function with a naive step size through gradient descent demo shown below:

In [57]:
```
# from gradient_descent import *
from plots import gradient_descent_visualization

# Define the parameters of the algorithm.
max_iters = 50

gamma = 0.1

# Initialization
x_initial = np.zeros(A.shape[1])

# Start gradient descent.
start_time = datetime.datetime.now()
gradient_objectives_naive, gradient_xs_naive = gradient_descent(b, A, x_initial, max_iters, gamma)
end_time = datetime.datetime.now()

# Print result
exection_time = (end_time - start_time).total_seconds()
print("Gradient Descent: execution time={t:.3f} seconds".format(t=exection_time))
```

```
Gradient Descent(0/49): objective=27922367.127591673
Gradient Descent(1/49): objective=27712999154772.4
Gradient Descent(2/49): objective=2.7657600715910955e+19
Gradient Descent(3/49): objective=2.7602313172080067e+25
Gradient Descent(4/49): objective=2.7547136148049077e+31
Gradient Descent(5/49): objective=2.749206942288922e+37
Gradient Descent(6/49): objective=2.7437112776112746e+43
Gradient Descent(7/49): objective=2.73822659876731e+49
Gradient Descent(8/49): objective=2.7327528837963527e+55
Gradient Descent(9/49): objective=2.7272901107816426e+61
Gradient Descent(10/49): objective=2.721838257850202e+67
Gradient Descent(11/49): objective=2.7163973031727706e+73
Gradient Descent(12/49): objective=2.7109672249637196e+79
Gradient Descent(13/49): objective=2.7055480014810067e+85
Gradient Descent(14/49): objective=2.700139611026039e+91
Gradient Descent(15/49): objective=2.6947420319435946e+97
Gradient Descent(16/49): objective=2.6893552426217576e+103
Gradient Descent(17/49): objective=2.6839792214917652e+109
Gradient Descent(18/49): objective=2.678613947028007e+115
Gradient Descent(19/49): objective=2.6732593977478876e+121
Gradient Descent(20/49): objective=2.667915552211779e+127
Gradient Descent(21/49): objective=2.662582389022902e+133
Gradient Descent(22/49): objective=2.6572598868272573e+139
Gradient Descent(23/49): objective=2.6519480243135004e+145
Gradient Descent(24/49): objective=2.6466467802128983e+151
Gradient Descent(25/49): objective=2.641356133299262e+157
Gradient Descent(26/49): objective=2.6360760623887817e+163
Gradient Descent(27/49): objective=2.630806546340078e+169
Gradient Descent(28/49): objective=2.6255475640539503e+175
Gradient Descent(29/49): objective=2.6202990944734056e+181
Gradient Descent(30/49): objective=2.6150611165835803e+187
Gradient Descent(31/49): objective=2.6098336094115208e+193
Gradient Descent(32/49): objective=2.6046165520263096e+199
Gradient Descent(33/49): objective=2.5994099235387985e+205
Gradient Descent(34/49): objective=2.594213703101647e+211
Gradient Descent(35/49): objective=2.58902786990915e+217
Gradient Descent(36/49): objective=2.583852403197197e+223
Gradient Descent(37/49): objective=2.578687282243221e+229
Gradient Descent(38/49): objective=2.5735324863659783e+235
Gradient Descent(39/49): objective=2.568387994925744e+241
Gradient Descent(40/49): objective=2.5632537873238695e+247
```

```
Gradient Descent(41/49): objective=2.558129843002995e+253
Gradient Descent(42/49): objective=2.553016141446836e+259
Gradient Descent(43/49): objective=2.547912662180085e+265
Gradient Descent(44/49): objective=2.542819384768383e+271
Gradient Descent(45/49): objective=2.5377362888182327e+277
Gradient Descent(46/49): objective=2.532663353976882e+283
Gradient Descent(47/49): objective=2.5276005599322827e+289
Gradient Descent(48/49): objective=2.5225478864129914e+295
Gradient Descent(49/49): objective=2.5175053131880417e+301
Gradient Descent: execution time=0.002 seconds
```

Time Visualization

In [81]:
```
from ipywidgets import interact, IntSlider
from grid_search import *


def plot_figure(n_iter):
    # Generate grid data for visualization (parameters to be swept and best combination)
    grid_x0, grid_x1 = generate_w(num_intervals=10)
    grid_objectives = grid_search(b, A, grid_x0, grid_x1)
    obj_star, x0_star, x1_star = get_best_parameters(grid_x0, grid_x1, grid_objectives)

    fig = gradient_descent_visualization(
        gradient_objectives, gradient_xs, grid_objectives, grid_x0, grid_x1, mean_x, std_x, height, weight, n_iter)
    fig.set_size_inches(20.0, 4.0)
    display(fig)

interact(plot_figure, n_iter=IntSlider(min=1, max=len(gradient_xs_naive)))
```

```
interactive(children=(IntSlider(value=1, description='n_iter', max=51, min=1), Output()), _dom_classes=('widge…
```

Out[81]:
```
<function __main__.plot_figure(n_iter)>
```

Try doing gradient descent with a better learning rate

In [76]:
```
# Define the parameters of the algorithm.
max_iters = 50

# ***************************************************
# INSERT YOUR CODE HERE
```

```
# TODO: a better learning rate using the smoothness of f
# ***************************************************
L = np.linalg.norm(A.T @ A, 2)
gamma = 1.0 / L

# Initialization
x_initial = np.zeros(A.shape[1])

# Start gradient descent.
start_time = datetime.datetime.now()
gradient_objectives, gradient_xs = gradient_descent(b, A, x_initial, max_iters, gamma)
end_time = datetime.datetime.now()

# Print result
exection_time = (end_time - start_time).total_seconds()
print("Gradient Descent: execution time={t:.3f} seconds".format(t=exection_time))
```

```
Gradient Descent(0/49): objective=27922367.127591673
Gradient Descent(1/49): objective=153858.87868829397
Gradient Descent(2/49): objective=153858.878688294
Gradient Descent(3/49): objective=153858.878688294
Gradient Descent(4/49): objective=153858.878688294
Gradient Descent(5/49): objective=153858.878688294
Gradient Descent(6/49): objective=153858.878688294
Gradient Descent(7/49): objective=153858.878688294
Gradient Descent(8/49): objective=153858.878688294
Gradient Descent(9/49): objective=153858.878688294
Gradient Descent(10/49): objective=153858.878688294
Gradient Descent(11/49): objective=153858.878688294
Gradient Descent(12/49): objective=153858.878688294
Gradient Descent(13/49): objective=153858.878688294
Gradient Descent(14/49): objective=153858.878688294
Gradient Descent(15/49): objective=153858.878688294
Gradient Descent(16/49): objective=153858.878688294
Gradient Descent(17/49): objective=153858.878688294
Gradient Descent(18/49): objective=153858.878688294
Gradient Descent(19/49): objective=153858.878688294
Gradient Descent(20/49): objective=153858.878688294
Gradient Descent(21/49): objective=153858.878688294
Gradient Descent(22/49): objective=153858.878688294
Gradient Descent(23/49): objective=153858.878688294
Gradient Descent(24/49): objective=153858.878688294
Gradient Descent(25/49): objective=153858.878688294
Gradient Descent(26/49): objective=153858.878688294
Gradient Descent(27/49): objective=153858.878688294
Gradient Descent(28/49): objective=153858.878688294
Gradient Descent(29/49): objective=153858.878688294
Gradient Descent(30/49): objective=153858.878688294
Gradient Descent(31/49): objective=153858.878688294
Gradient Descent(32/49): objective=153858.878688294
Gradient Descent(33/49): objective=153858.878688294
Gradient Descent(34/49): objective=153858.878688294
Gradient Descent(35/49): objective=153858.878688294
Gradient Descent(36/49): objective=153858.878688294
Gradient Descent(37/49): objective=153858.878688294
Gradient Descent(38/49): objective=153858.878688294
Gradient Descent(39/49): objective=153858.878688294
Gradient Descent(40/49): objective=153858.878688294
```

```
Gradient Descent(41/49): objective=153858.878688294
Gradient Descent(42/49): objective=153858.878688294
Gradient Descent(43/49): objective=153858.878688294
Gradient Descent(44/49): objective=153858.878688294
Gradient Descent(45/49): objective=153858.878688294
Gradient Descent(46/49): objective=153858.878688294
Gradient Descent(47/49): objective=153858.878688294
Gradient Descent(48/49): objective=153858.878688294
Gradient Descent(49/49): objective=153858.878688294
Gradient Descent: execution time=0.003 seconds
```

Time visualization with a better learning rate

In [77]:
```python
def plot_figure(n_iter):
    # Generate grid data for visualization (parameters to be swept and best combination)
    grid_x0, grid_x1 = generate_w(num_intervals=10)
    grid_objectives = grid_search(b, A, grid_x0, grid_x1)
    obj_star, x0_star, x1_star = get_best_parameters(grid_x0, grid_x1, grid_objectives)

    fig = gradient_descent_visualization(
        gradient_objectives, gradient_xs, grid_objectives, grid_x0, grid_x1, mean_x, std_x, height, weight, n_iter)
    fig.set_size_inches(10.0, 6.0)
    display(fig)

interact(plot_figure, n_iter=IntSlider(min=1, max=len(gradient_xs)))
```

```
interactive(children=(IntSlider(value=1, description='n_iter', max=51, min=1), Output()), _dom_classes=('widge…
```

Out[77]:
```
<function __main__.plot_figure(n_iter)>
```

Loading more complex data

The data is taken from https://archive.ics.uci.edu/ml/datasets/Concrete+Compressive+Strength

In [82]:
```python
data = np.loadtxt("Concrete_Data.csv",delimiter=",")

A = data[:,:-1]
```

```
b = data[:,-1]
A, mean_A, std_A = standardize(A)
```

In [83]:
```
print('Number of samples n = ', b.shape[0])
print('Dimension of each sample d = ', A.shape[1])
```

```
Number of samples n =  1030
Dimension of each sample d =  8
```

Running gradient descent

Assuming bounded gradients

Assume we are moving in a bounded region $\|x\| \leq 25$ containing all iterates (and we assume $\|x - x^\star\| \leq 25$ as well, for simplicity). Then by $\nabla f(x) = \frac{1}{n} A^\top (Ax - b)$, one can see that f is Lipschitz over that bounded region, with Lipschitz constant $\|\nabla f(x)\| \leq \frac{1}{n}(\|A^\top A\| \|x\| + \|A^\top b\|)$

In [90]:
```
# ***************************************************
# INSERT YOUR CODE HERE
# TODO: Compute the bound on the gradient norm
# ***************************************************
norm_ATA = np.linalg.norm(A.T @ A, 2)
norm_ATb = np.linalg.norm(A.T @ b, 2)
n = b.shape[0]
grad_norm_bound = (norm_ATA * 25 + norm_ATb) / n
```

Fill in the learning rate assuming bounded gradients

In [92]:
```
max_iters = 50

# ***************************************************
# INSERT YOUR CODE HERE
# TODO: Compute learning rate based on bounded gradient
# ***************************************************
gamma =  1.0 / grad_norm_bound
```

```python
# Initialization
x_initial = np.zeros(A.shape[1])

# Start gradient descent.
start_time = datetime.datetime.now()
bd_gradient_objectives, bd_gradient_xs = gradient_descent(b, A, x_initial, max_iters, gamma)
end_time = datetime.datetime.now()


# Print result
exection_time = (end_time - start_time).total_seconds()
print("Gradient Descent: execution time={t:.3f} seconds".format(t=exection_time))

# Averaging the iterates as is the case for bounded gradients case
bd_gradient_objectives_averaged = []
for i in range(len(bd_gradient_xs)):
    if i > 0:
        bd_gradient_xs[i] = (i * bd_gradient_xs[i-1] + bd_gradient_xs[i])/(i + 1)
    grad, err = compute_gradient(b, A, bd_gradient_xs[i])
    obj = calculate_objective(err)
    bd_gradient_objectives_averaged.append(obj)
```

```
Gradient Descent(0/49): objective=804294.6597
Gradient Descent(1/49): objective=22182316.13626735
Gradient Descent(2/49): objective=6781412750.83074
Gradient Descent(3/49): objective=2485261680516.3525
Gradient Descent(4/49): objective=1108146630365384.0
Gradient Descent(5/49): objective=6.537956716248343e+17
Gradient Descent(6/49): objective=5.085467340605488e+20
Gradient Descent(7/49): objective=4.698707106504167e+23
Gradient Descent(8/49): objective=4.6872266444377457e+26
Gradient Descent(9/49): objective=4.81220386521648e+29
Gradient Descent(10/49): objective=4.990553690310146e+32
Gradient Descent(11/49): objective=5.193399076861745e+35
Gradient Descent(12/49): objective=5.410834089681546e+38
Gradient Descent(13/49): objective=5.639621176816034e+41
Gradient Descent(14/49): objective=5.878880084790928e+44
Gradient Descent(15/49): objective=6.128573286219311e+47
Gradient Descent(16/49): objective=6.388972897525534e+50
Gradient Descent(17/49): objective=6.660472934854817e+53
Gradient Descent(18/49): objective=6.943523375014137e+56
Gradient Descent(19/49): objective=7.2386072940494704e+59
Gradient Descent(20/49): objective=7.546233292928493e+62
Gradient Descent(21/49): objective=7.866933377761103e+65
Gradient Descent(22/49): objective=8.201262807732291e+68
Gradient Descent(23/49): objective=8.54980067171551e+71
Gradient Descent(24/49): objective=8.913150752793004e+74
Gradient Descent(25/49): objective=9.291942524183193e+77
Gradient Descent(26/49): objective=9.686832221623424e+80
Gradient Descent(27/49): objective=1.0098503973672407e+84
Gradient Descent(28/49): objective=1.0527670984528167e+87
Gradient Descent(29/49): objective=1.097507676874488e+90
Gradient Descent(30/49): objective=1.1441496439018957e+93
Gradient Descent(31/49): objective=1.1927738048921672e+96
Gradient Descent(32/49): objective=1.2434643992776057e+99
Gradient Descent(33/49): objective=1.296309246505524e+102
Gradient Descent(34/49): objective=1.3513998981813633e+105
Gradient Descent(35/49): objective=1.408831796677994e+108
Gradient Descent(36/49): objective=1.4687044404857616e+111
Gradient Descent(37/49): objective=1.5311215565896487e+114
Gradient Descent(38/49): objective=1.596191280172167e+117
Gradient Descent(39/49): objective=1.6640263419532658e+120
Gradient Descent(40/49): objective=1.7347442634918465e+123
```

```
Gradient Descent(41/49): objective=1.8084675607871975e+126
Gradient Descent(42/49): objective=1.8853239565331308e+129
Gradient Descent(43/49): objective=1.9654466013924755e+132
Gradient Descent(44/49): objective=2.0489743046753393e+135
Gradient Descent(45/49): objective=2.1360517748207414e+138
Gradient Descent(46/49): objective=2.226829870098249e+141
Gradient Descent(47/49): objective=2.3214658599639647e+144
Gradient Descent(48/49): objective=2.420123697523623e+147
Gradient Descent(49/49): objective=2.5229743035748695e+150
Gradient Descent: execution time=0.001 seconds
```

Gradient descent using smoothness

Fill in the learning rate using smoothness of the function

In [93]:
```python
max_iters = 50


# ***************************************************
# INSERT YOUR CODE HERE
# TODO: a better learning rate using the smoothness of f
# ***************************************************
L = np.linalg.norm(A.T @ A, 2)
gamma = 1.0 / L

# Initialization
x_initial = np.zeros(A.shape[1])

# Start gradient descent.
start_time = datetime.datetime.now()
gradient_objectives, gradient_xs = gradient_descent(b, A, x_initial, max_iters, gamma)
end_time = datetime.datetime.now()

# Print result
exection_time = (end_time - start_time).total_seconds()
print("Gradient Descent: execution time={t:.3f} seconds".format(t=exection_time))
```

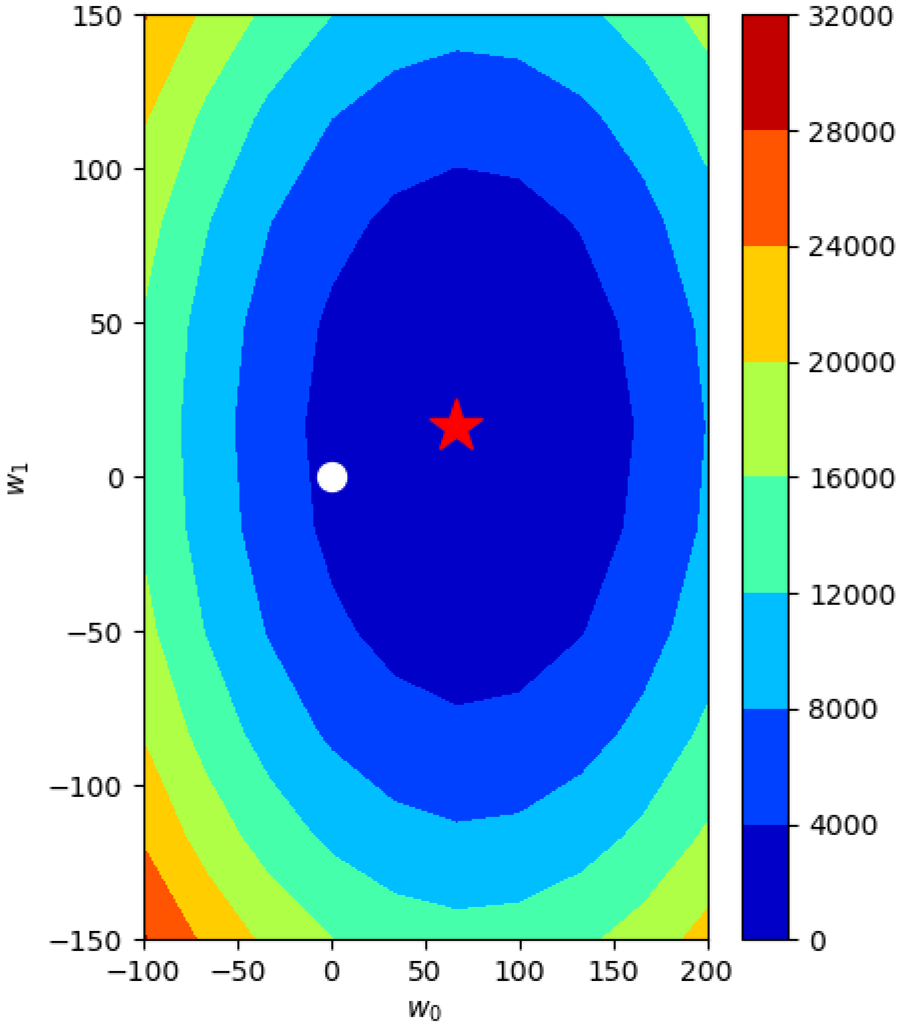
```
Gradient Descent(0/49): objective=804294.6597
Gradient Descent(1/49): objective=743150.4137068497
Gradient Descent(2/49): objective=727987.8891844836
Gradient Descent(3/49): objective=723230.9391829739
Gradient Descent(4/49): objective=721329.1577396914
Gradient Descent(5/49): objective=720374.5101028575
Gradient Descent(6/49): objective=719791.309291003
Gradient Descent(7/49): objective=719378.0034189869
Gradient Descent(8/49): objective=719055.5456951152
Gradient Descent(9/49): objective=718789.6632368966
Gradient Descent(10/49): objective=718563.7332546096
Gradient Descent(11/49): objective=718368.5515697026
Gradient Descent(12/49): objective=718198.2734231742
Gradient Descent(13/49): objective=718048.7351753768
Gradient Descent(14/49): objective=717916.7265493354
Gradient Descent(15/49): objective=717799.6529447329
Gradient Descent(16/49): objective=717695.3627793654
Gradient Descent(17/49): objective=717602.047734058
Gradient Descent(18/49): objective=717518.1774173777
Gradient Descent(19/49): objective=717442.4519295725
Gradient Descent(20/49): objective=717373.7649275991
Gradient Descent(21/49): objective=717311.1736497871
Gradient Descent(22/49): objective=717253.8740324158
Gradient Descent(23/49): objective=717201.1798081738
Gradient Descent(24/49): objective=717152.504843353
Gradient Descent(25/49): objective=717107.3481664611
Gradient Descent(26/49): objective=717065.2812586302
Gradient Descent(27/49): objective=717025.9372556831
Gradient Descent(28/49): objective=716989.0017706066
Gradient Descent(29/49): objective=716954.2050915522
Gradient Descent(30/49): objective=716921.3155483499
Gradient Descent(31/49): objective=716890.1338720422
Gradient Descent(32/49): objective=716860.4883984638
Gradient Descent(33/49): objective=716832.2309893103
Gradient Descent(34/49): objective=716805.2335631507
Gradient Descent(35/49): objective=716779.3851449764
Gradient Descent(36/49): objective=716754.5893565833
Gradient Descent(37/49): objective=716730.7622817411
Gradient Descent(38/49): objective=716707.8306500015
Gradient Descent(39/49): objective=716685.7302914094
Gradient Descent(40/49): objective=716664.4048215485
```

```
Gradient Descent(41/49): objective=716643.8045224195
Gradient Descent(42/49): objective=716623.8853898316
Gradient Descent(43/49): objective=716604.6083223762
Gradient Descent(44/49): objective=716585.9384307899
Gradient Descent(45/49): objective=716567.844449691
Gradient Descent(46/49): objective=716550.2982363733
Gradient Descent(47/49): objective=716533.2743436344
Gradient Descent(48/49): objective=716516.7496555699
Gradient Descent(49/49): objective=716500.7030769235
Gradient Descent: execution time=0.002 seconds
```

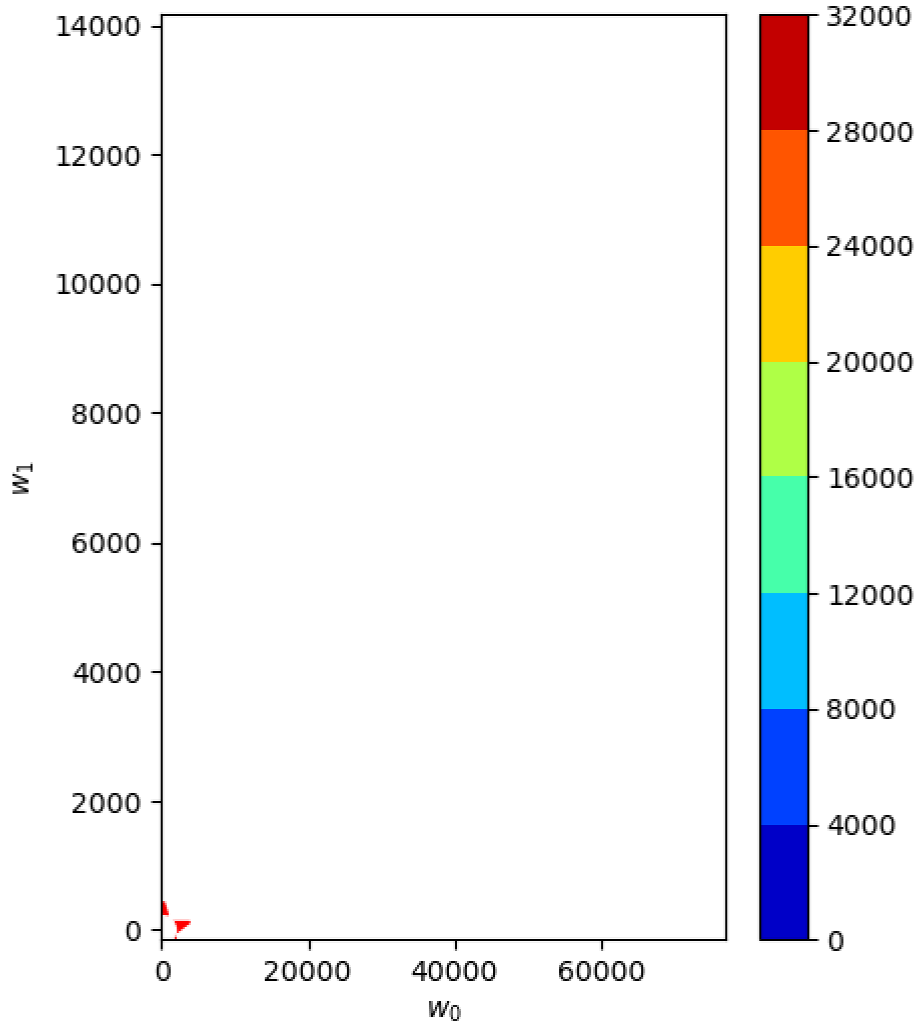
Plotting the Evolution of the Objective Function

In [94]:

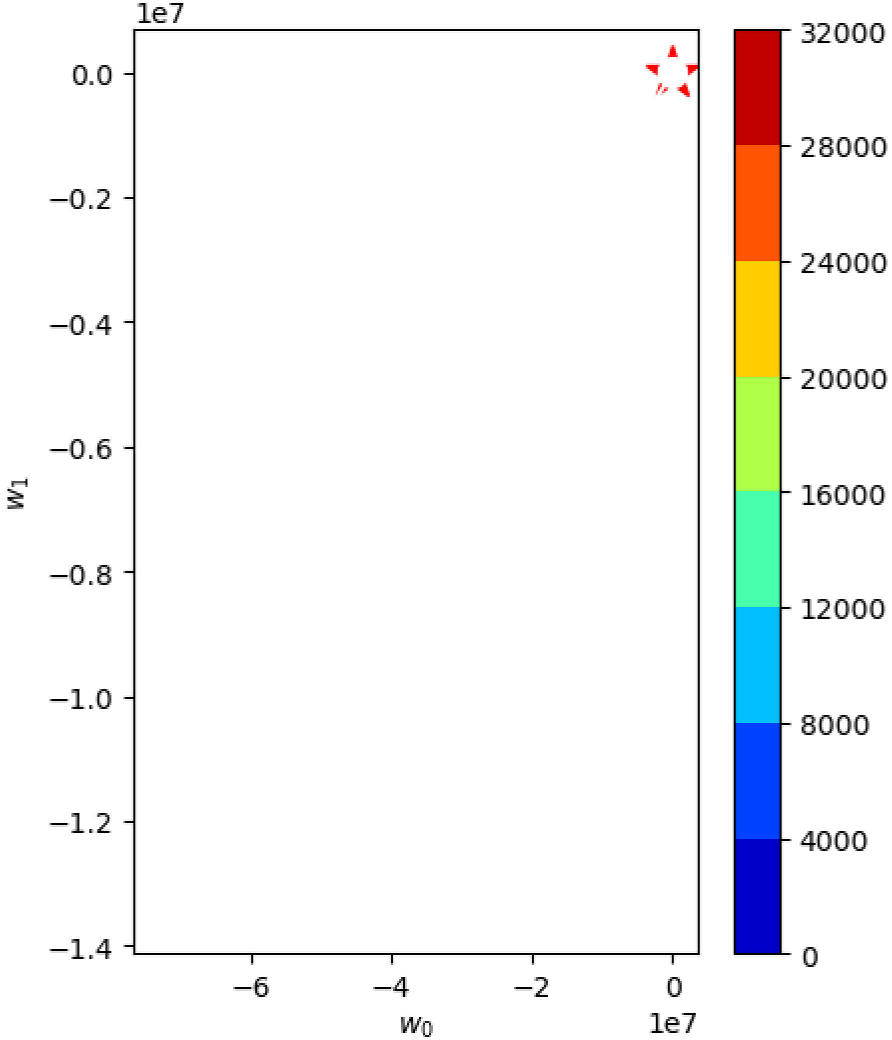
```
plt.figure(figsize=(8, 8))
plt.xlabel('Number of steps')
plt.ylabel('Objective Function')
#plt.yscale("log")
plt.plot(range(len(gradient_objectives)), gradient_objectives,'r', label='gradient descent with 1/L stepsize')
plt.plot(range(len(bd_gradient_objectives)), bd_gradient_objectives,'b', label='gradient descent assuming bounded gradients')
plt.plot(range(len(bd_gradient_objectives_averaged)), bd_gradient_objectives_averaged,'g', label='gradient descent assuming bo
plt.legend(loc='upper right')
plt.show()
```



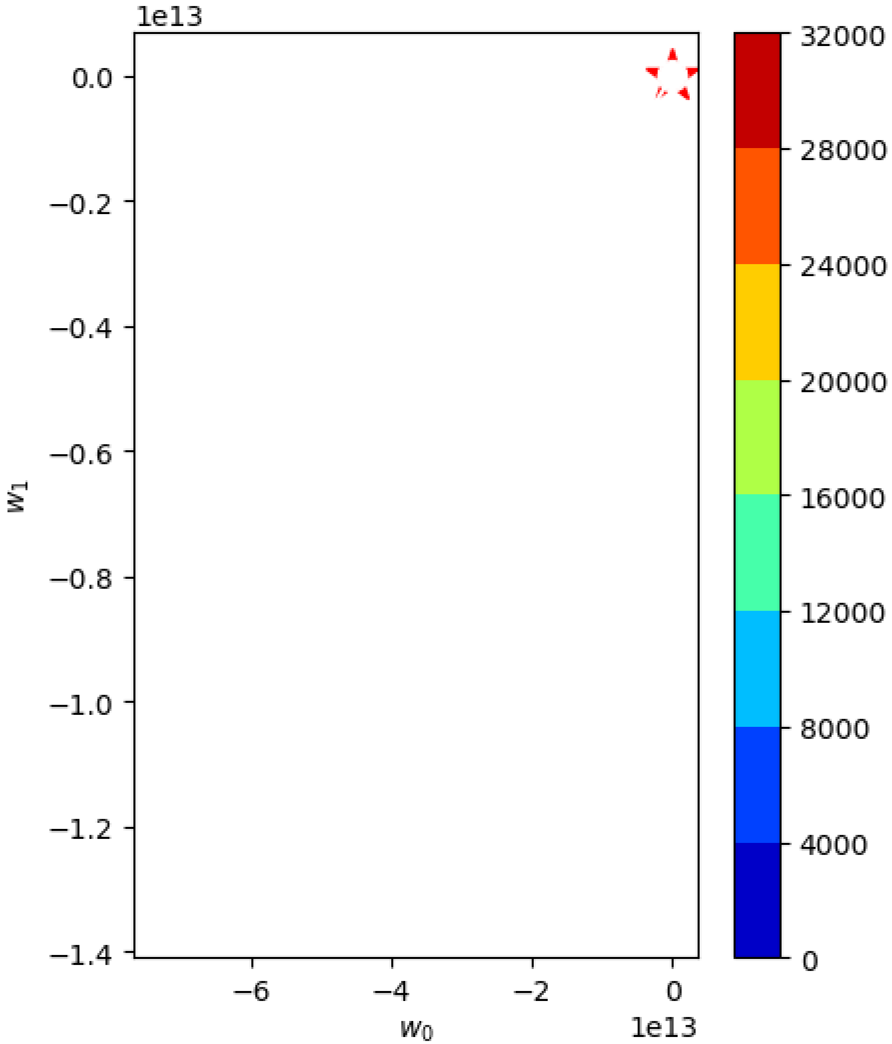
150
100
50
0
−50
−100
−150
−100
−50
0
50
100
150
200
w_0
w_1
32000
28000
24000
20000
16000
12000
8000
4000
0



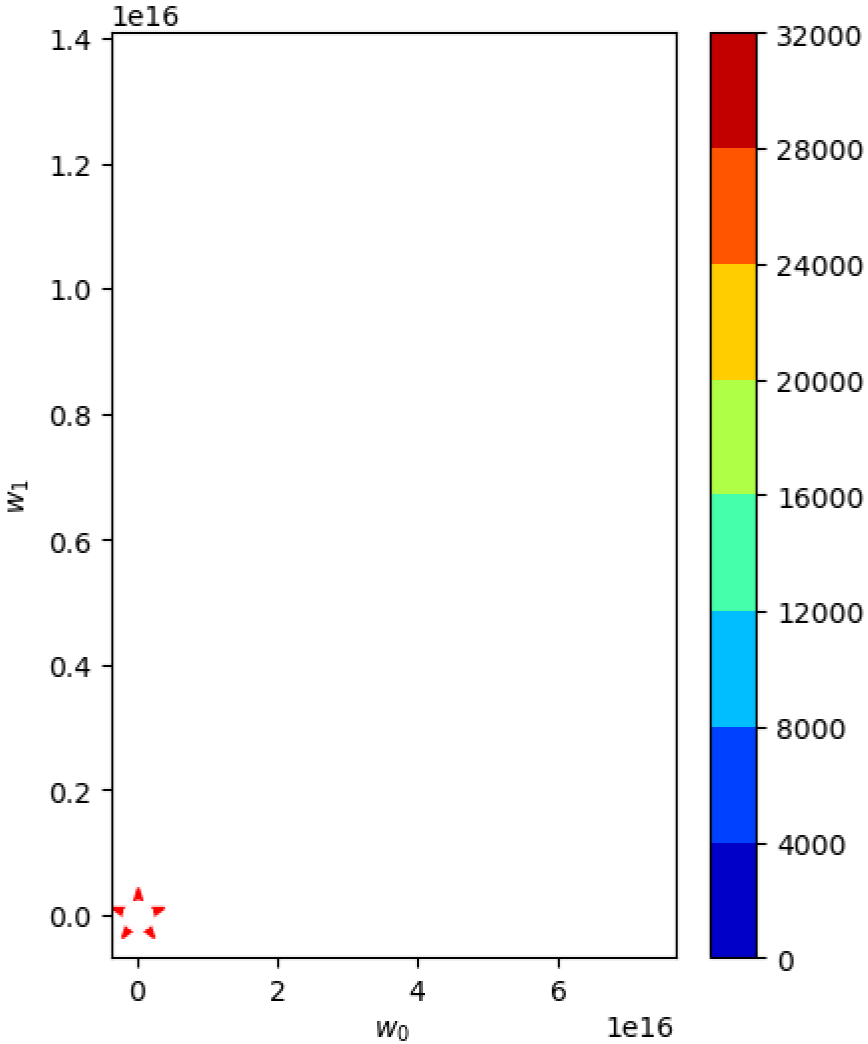
14000
12000
10000
8000
6000
4000
2000
0
w_1
0
20000
40000
60000
w_0
32000
28000
24000
20000
16000
12000
8000
4000
0

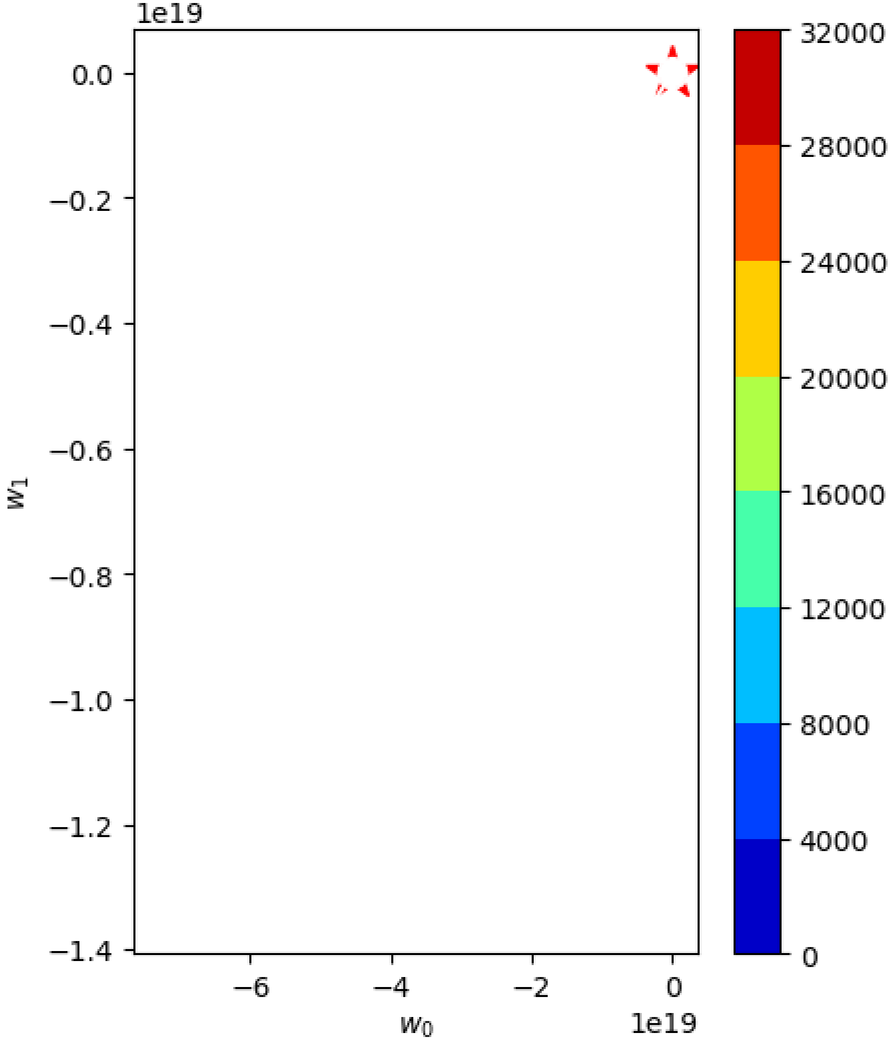


1e7
0.0
−0.2
−0.4
−0.6
−0.8
−1.0
−1.2
−1.4
w_1
−6
−4
−2
0
1e7
w_0
32000
28000
24000
20000
16000
12000
8000
4000
0

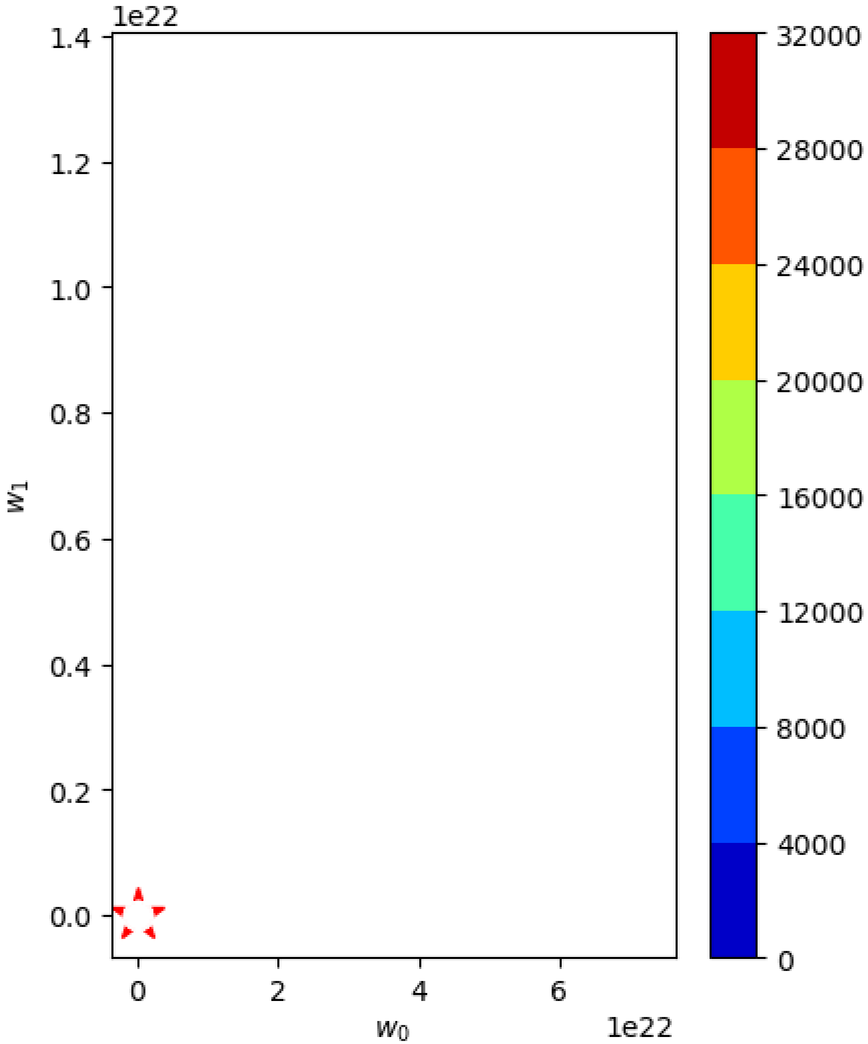


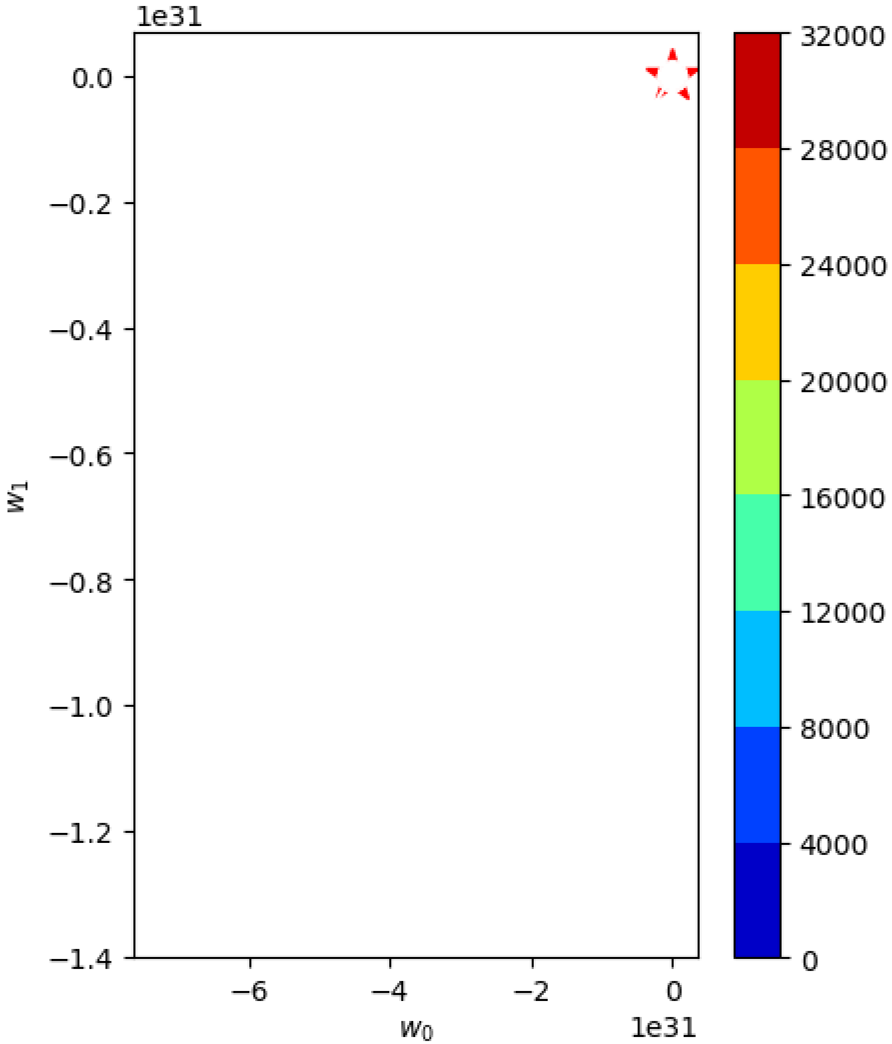
1e13
0.0
−0.2
−0.4
−0.6
−0.8
−1.0
−1.2
−1.4
w_1
−6
−4
−2
0
w_0
1e13
32000
28000
24000
20000
16000
12000
8000
4000
0

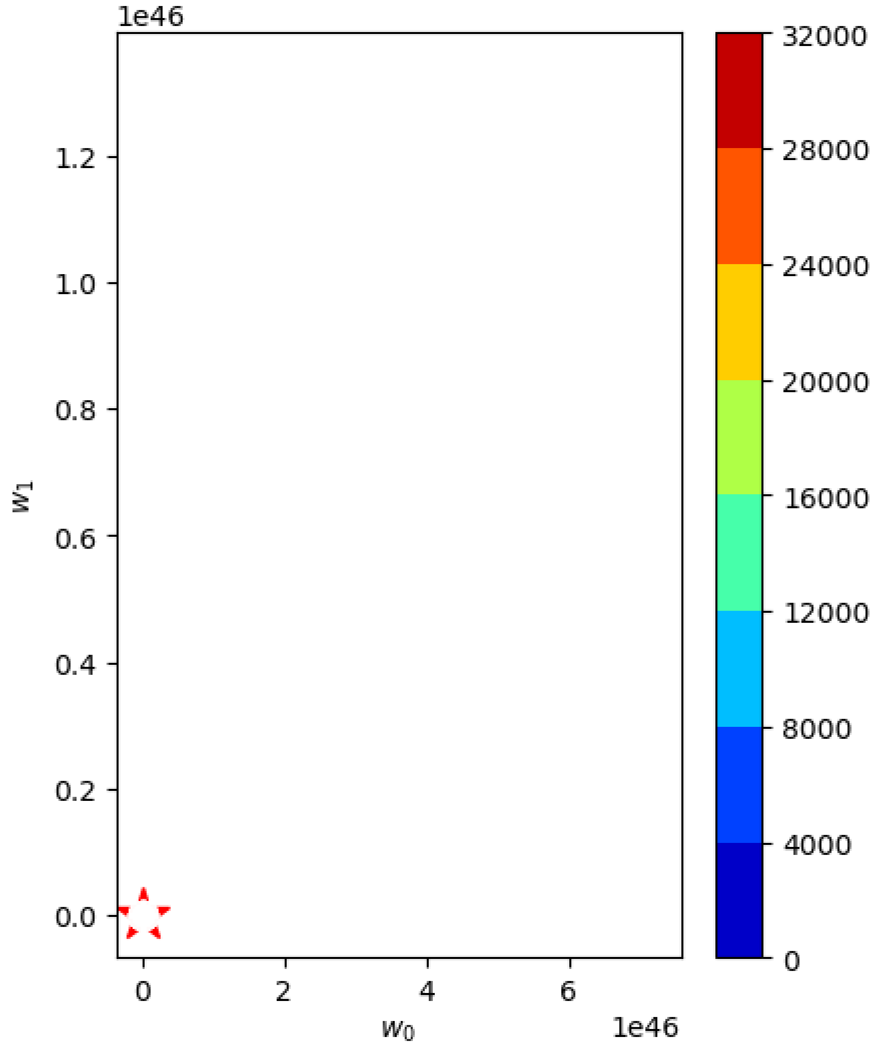

1e16
1.4
1.2
1.0
0.8
0.6
0.4
0.2
0.0
w_1
0
2
4
6
w_0
1e16
32000
28000
24000
20000
16000
12000
8000
4000
0



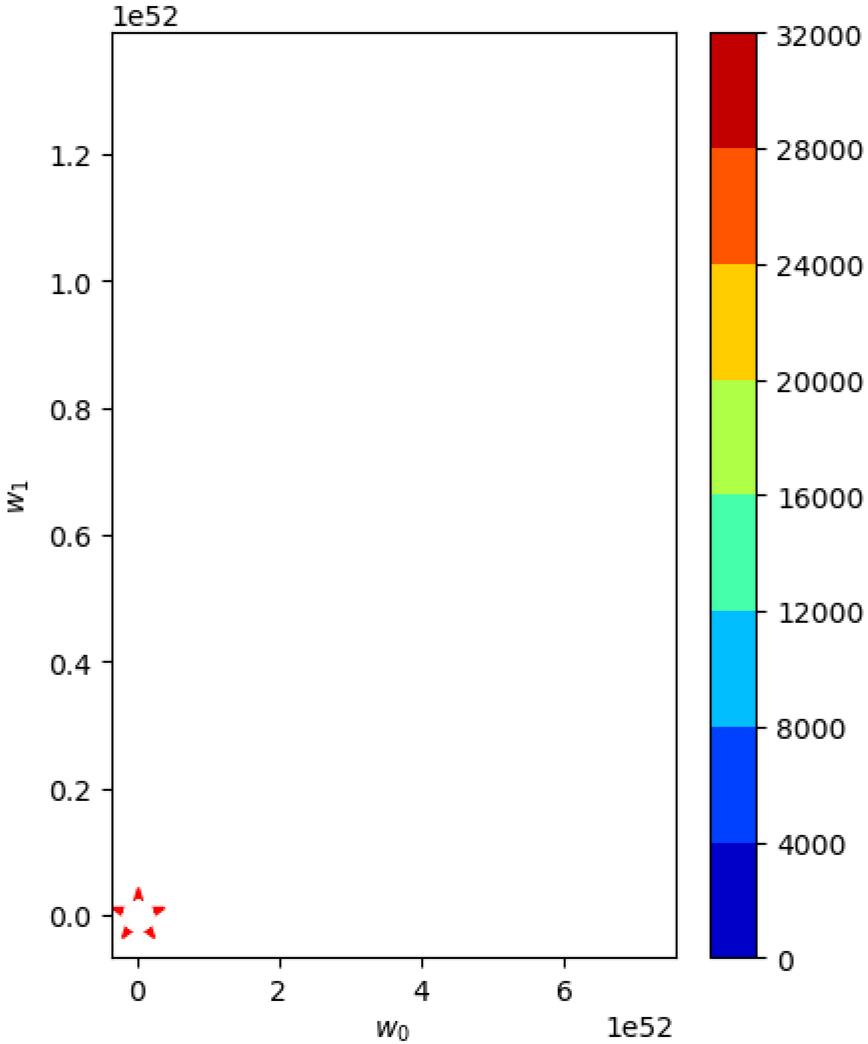
1e19
0.0
−0.2
−0.4
−0.6
−0.8
−1.0
−1.2
−1.4
w_1
−6
−4
−2
0
w_0
1e19
32000
28000
24000
20000
16000
12000
8000
4000
0

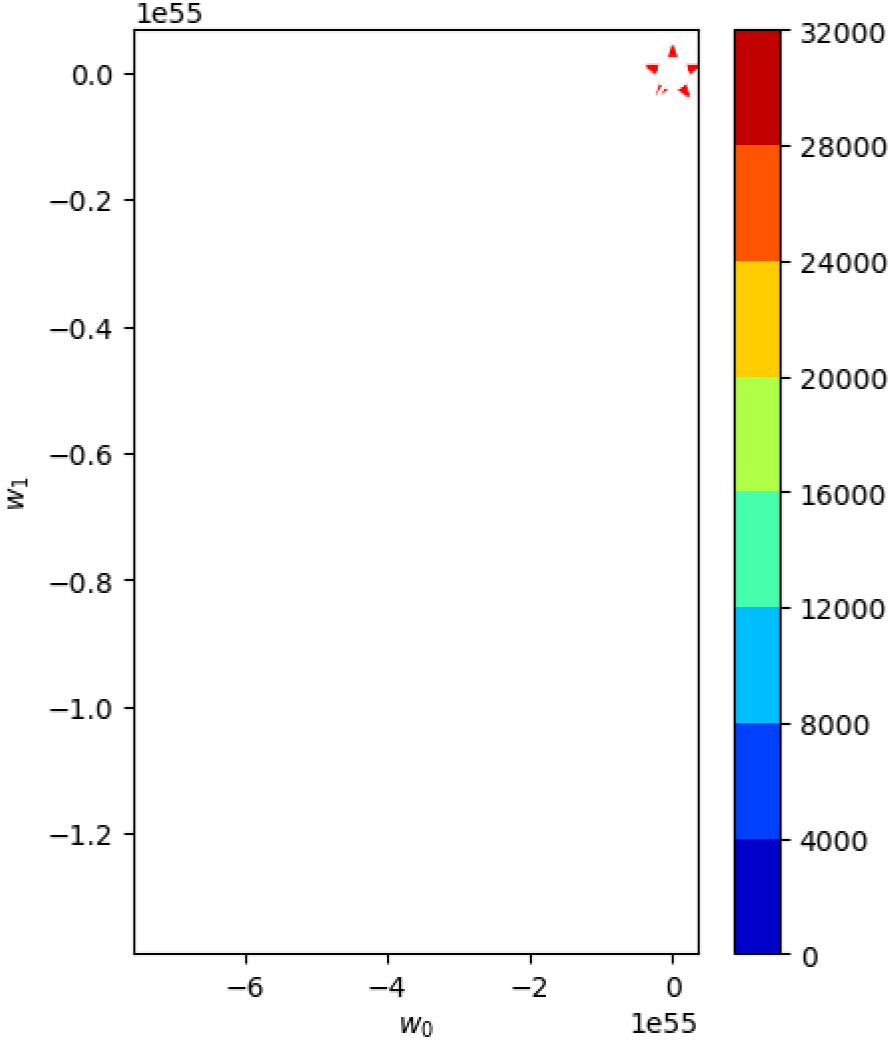

1e22
1.4
1.2
1.0
0.8
0.6
0.4
0.2
0.0
w_1
0
2
4
6
w_0
1e22
32000
28000
24000
20000
16000
12000
8000
4000
0


1e31
0.0
−0.2
−0.4
−0.6
−0.8
−1.0
−1.2
−1.4
w_1
−6
−4
−2
0
w_0
1e31
32000
28000
24000
20000
16000
12000
8000
4000
0

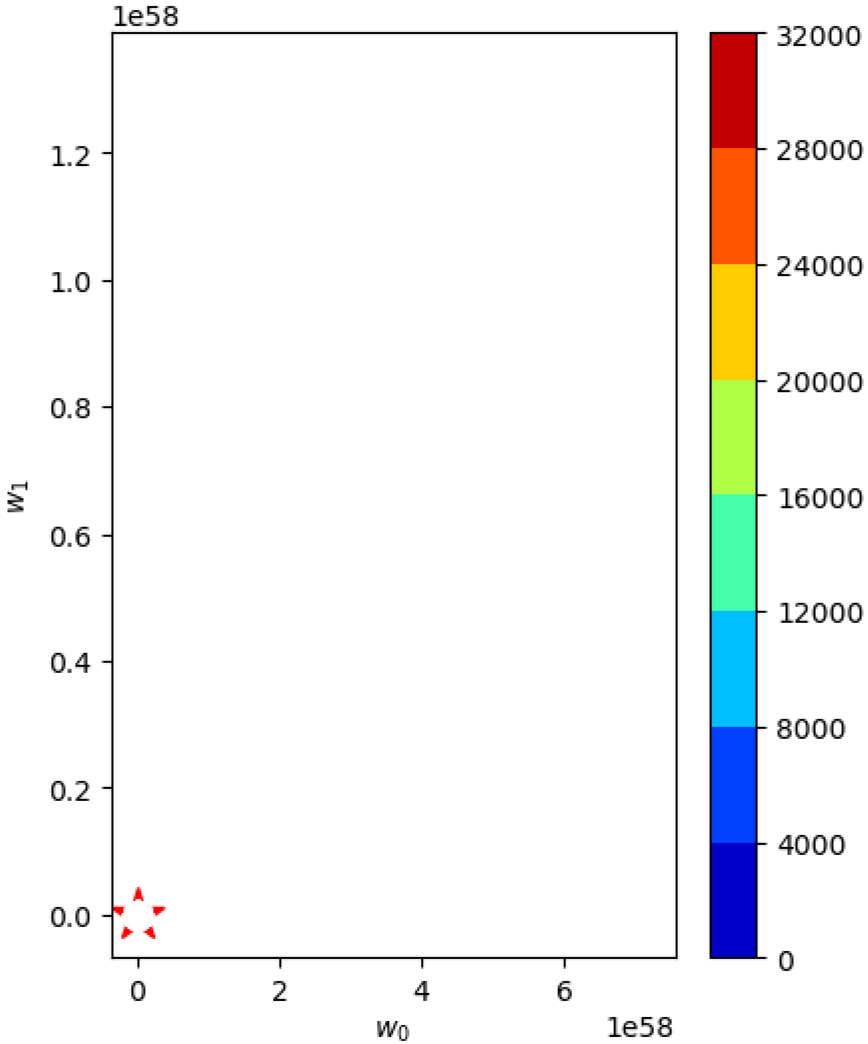


1e46
1.2
1.0
0.8
0.6
0.4
0.2
0.0
w_1
0
2
4
6
w_0
1e46
32000
28000
24000
20000
16000
12000
8000
4000
0

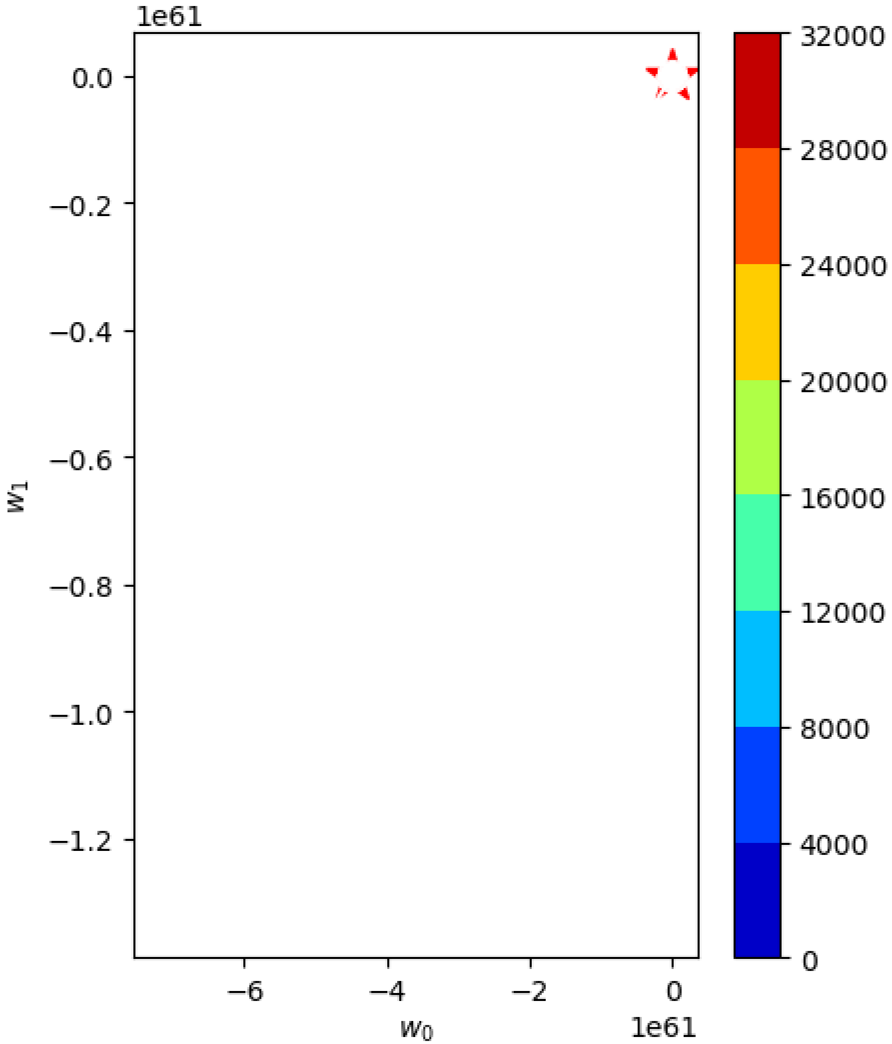

1e52
1.2
1.0
0.8
0.6
0.4
0.2
0.0
w_1
0
2
4
6
w_0
1e52
32000
28000
24000
20000
16000
12000
8000
4000
0



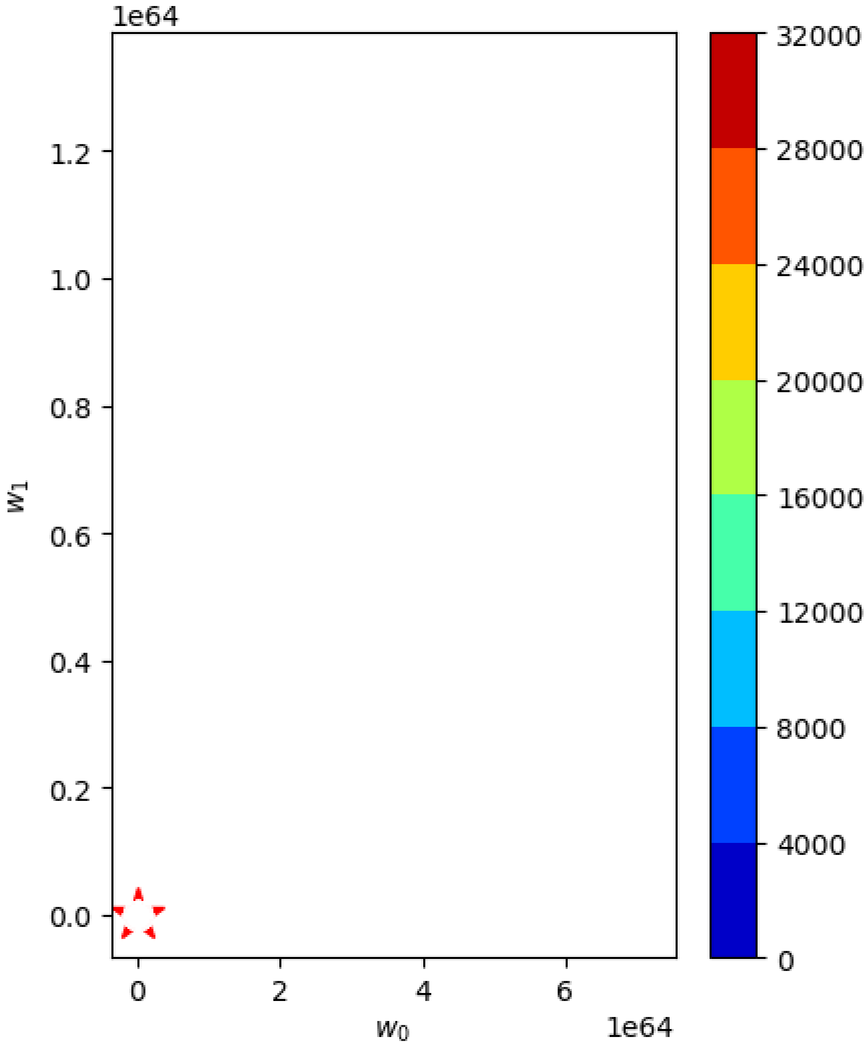
1e55
0.0
−0.2
−0.4
−0.6
−0.8
−1.0
−1.2
w_1
−6
−4
−2
0
1e55
w_0
32000
28000
24000
20000
16000
12000
8000
4000
0



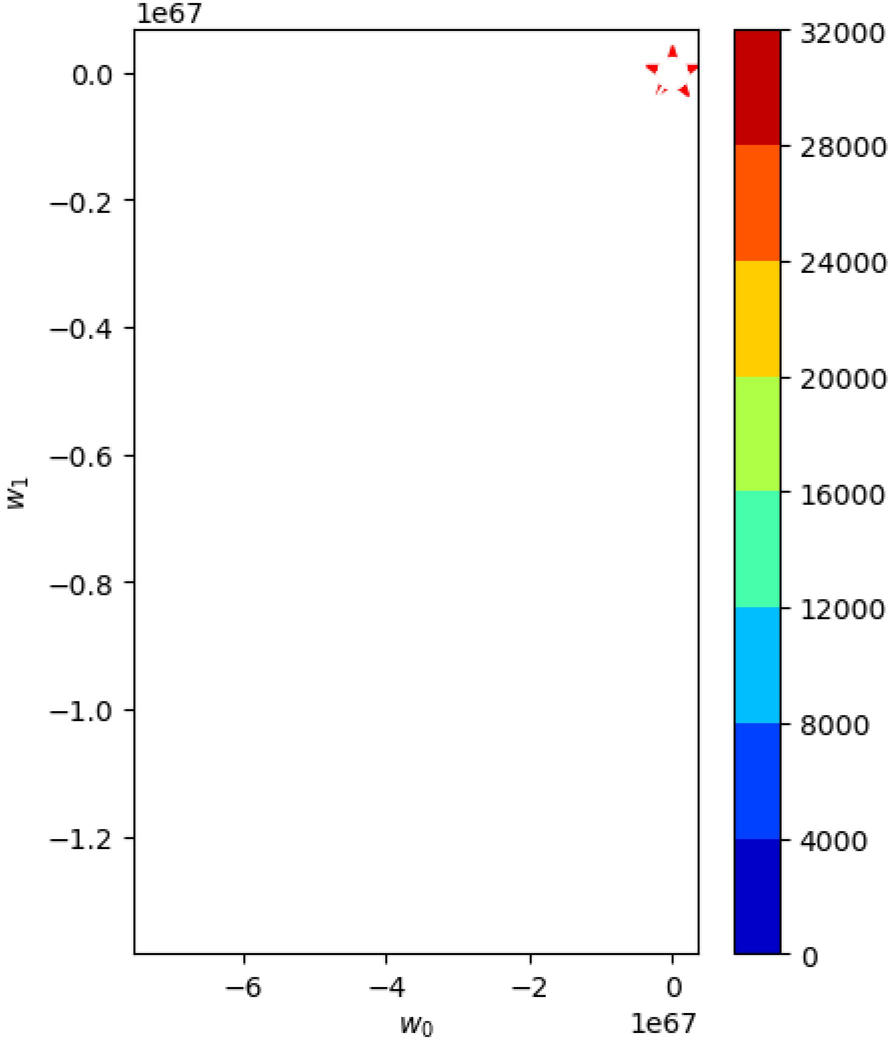
1e58
1.2
1.0
0.8
0.6
0.4
0.2
0.0
w_1
0
2
4
6
w_0
1e58
32000
28000
24000
20000
16000
12000
8000
4000
0

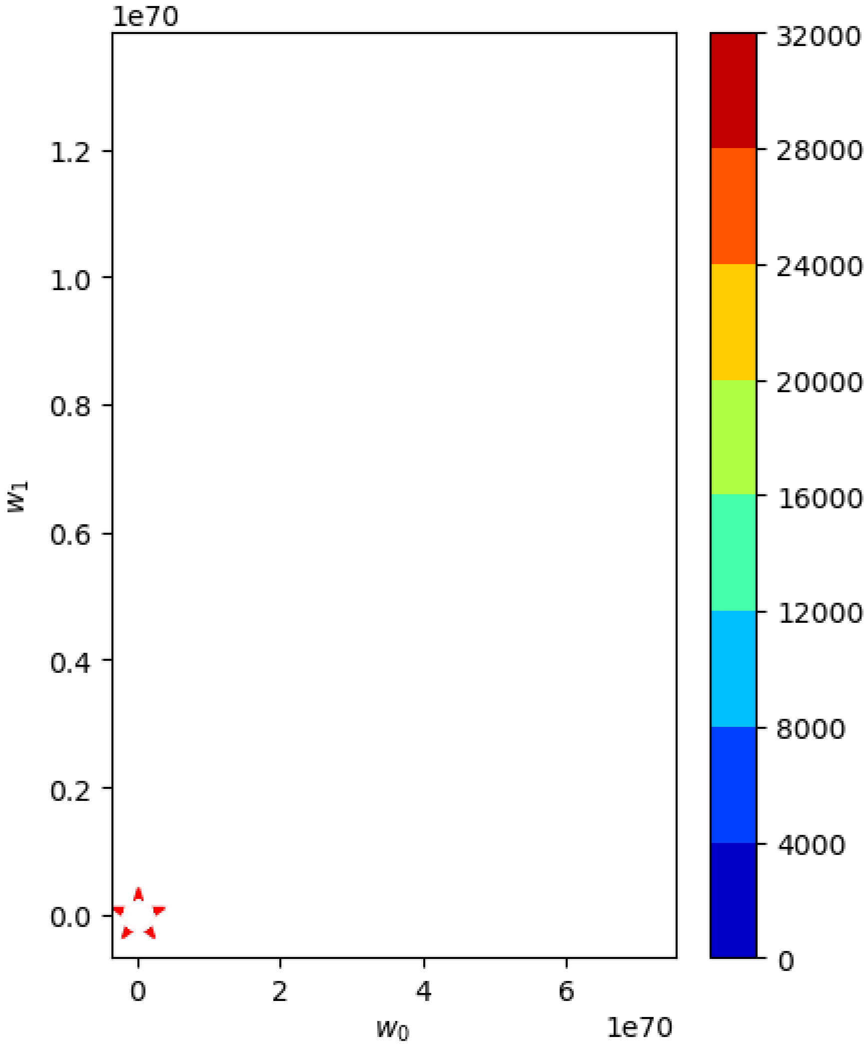


1e61
0.0
−0.2
−0.4
−0.6
−0.8
−1.0
−1.2
w_1
−6
−4
−2
0
w_0
1e61
32000
28000
24000
20000
16000
12000
8000
4000
0

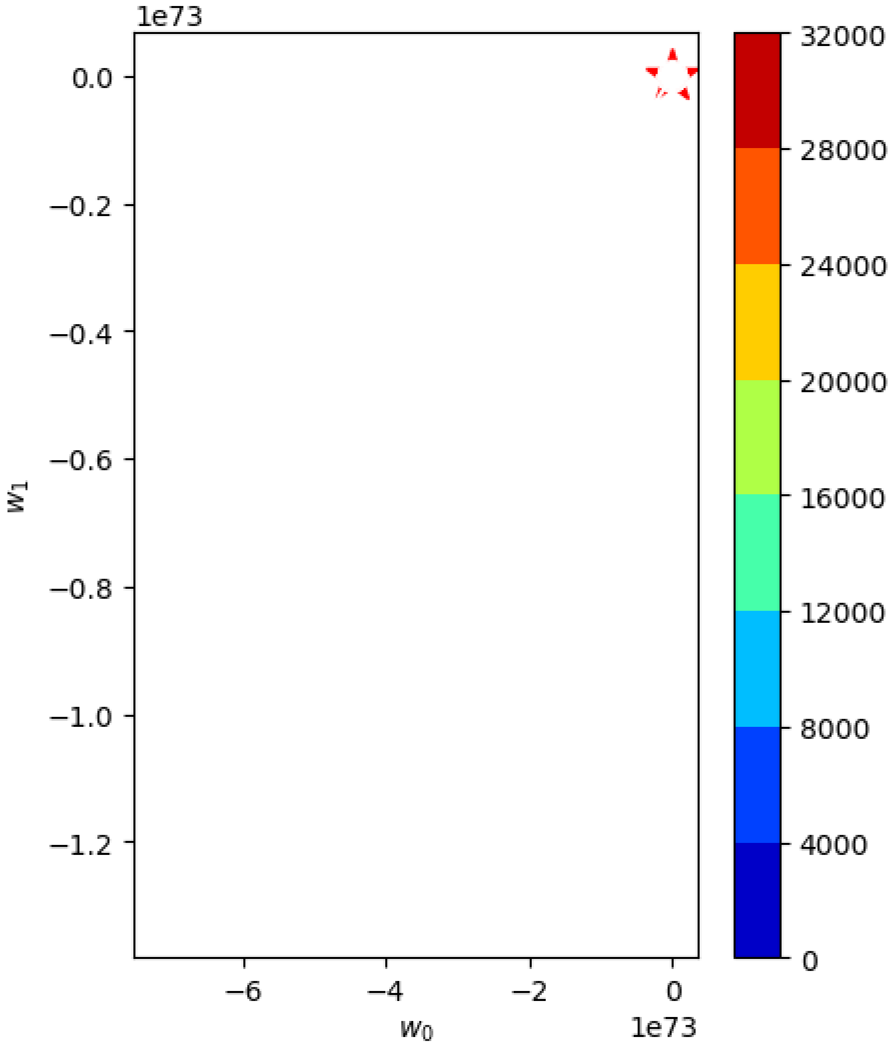


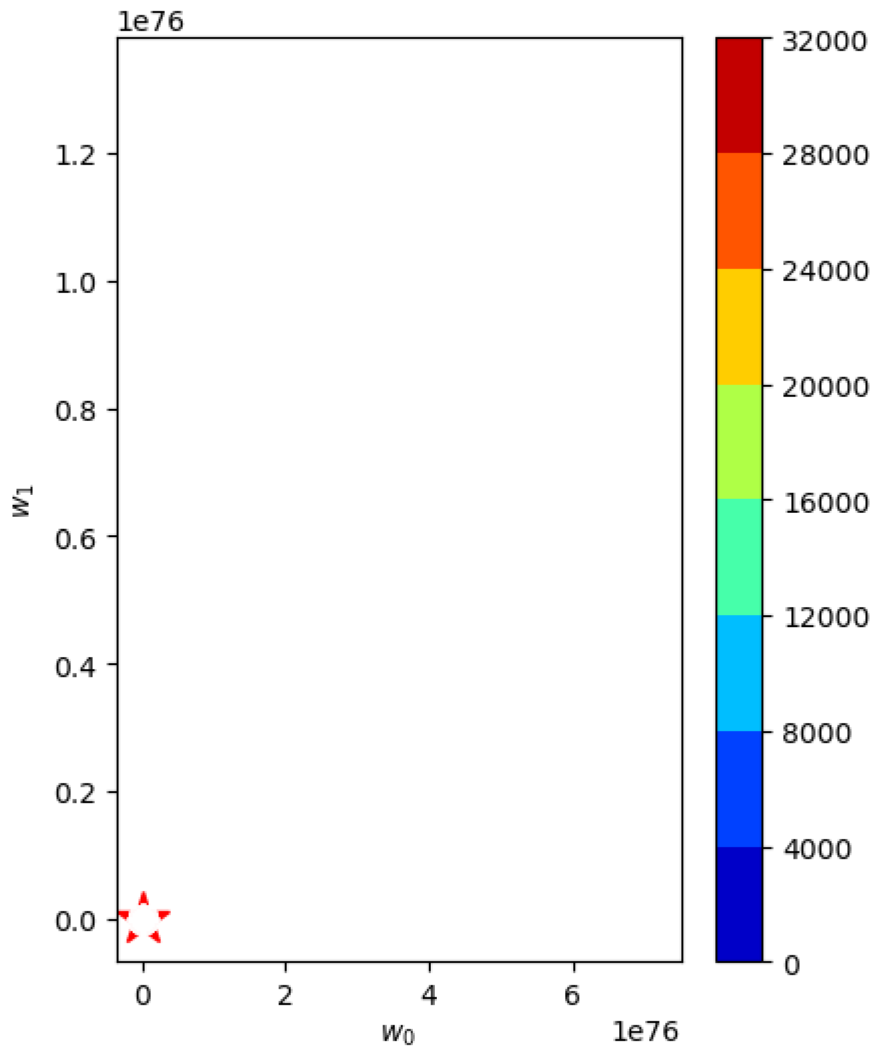
1e64
1.2
1.0
0.8
0.6
0.4
0.2
0.0
w_1
0
2
4
6
w_0
1e64
32000
28000
24000
20000
16000
12000
8000
4000
0

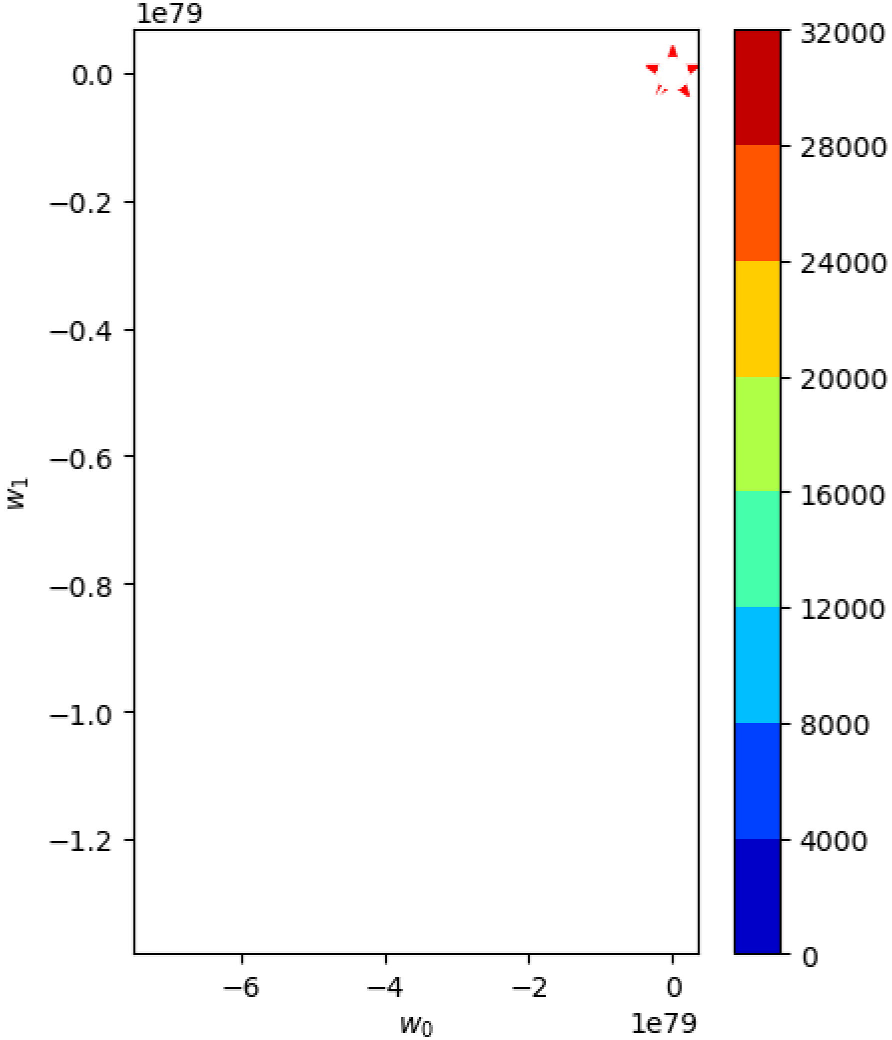

1e67
0.0
−0.2
−0.4
−0.6
−0.8
−1.0
−1.2
w_1
−6
−4
−2
0
1e67
w_0
32000
28000
24000
20000
16000
12000
8000
4000
0

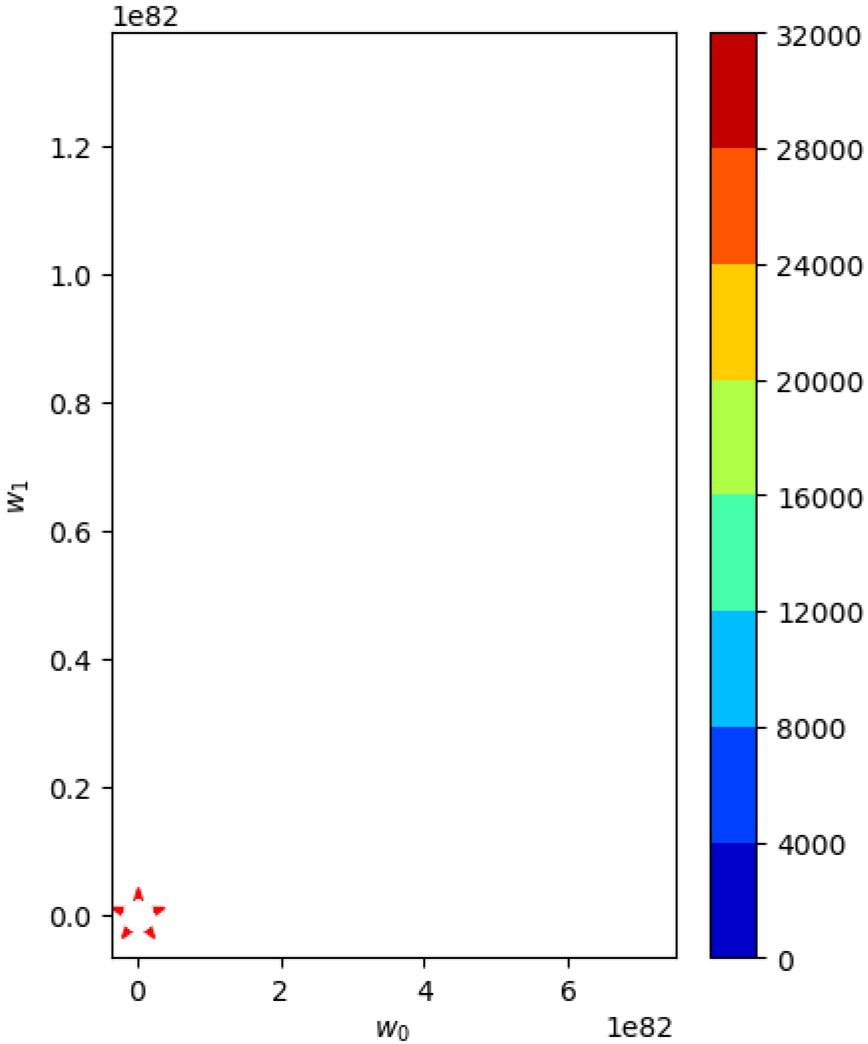


1e70
1.2
1.0
0.8
w_1
0.6
0.4
0.2
0.0
0
2
4
6
w_0
1e70
32000
28000
24000
20000
16000
12000
8000
4000
0

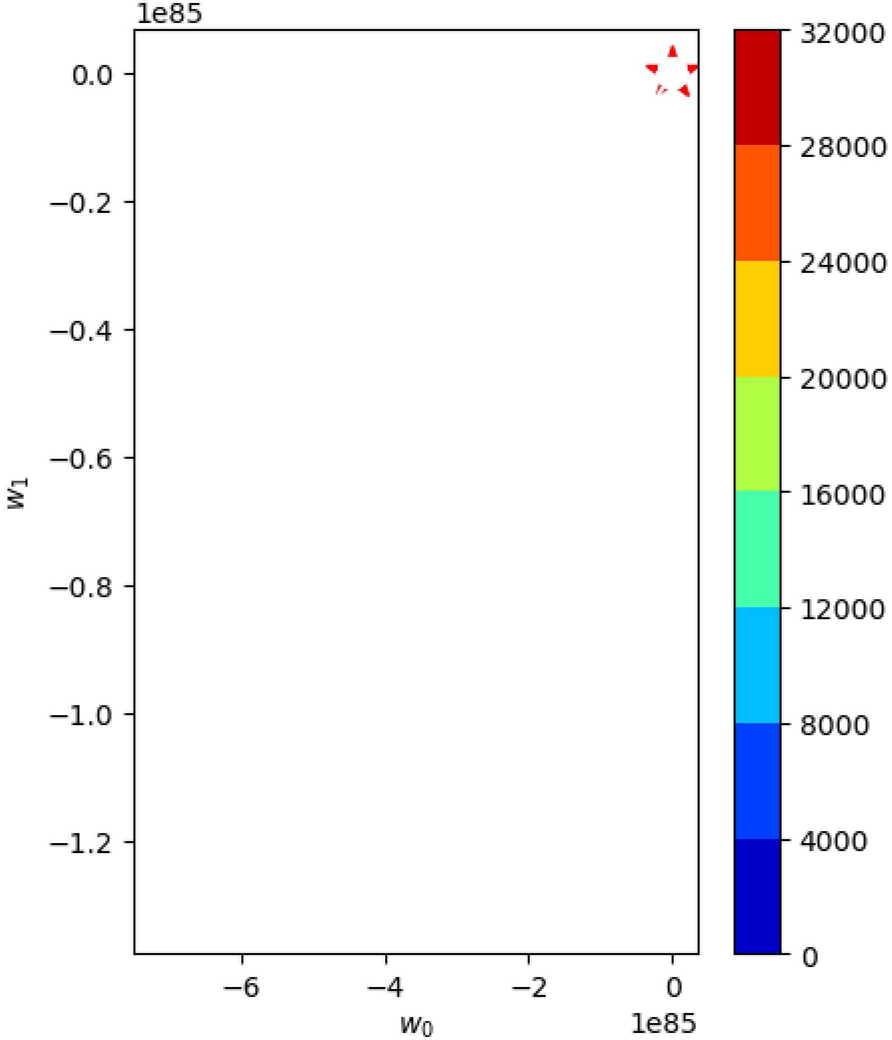

1e73
0.0
−0.2
−0.4
−0.6
−0.8
−1.0
−1.2
w_1
−6
−4
−2
0
1e73
w_0
32000
28000
24000
20000
16000
12000
8000
4000
0

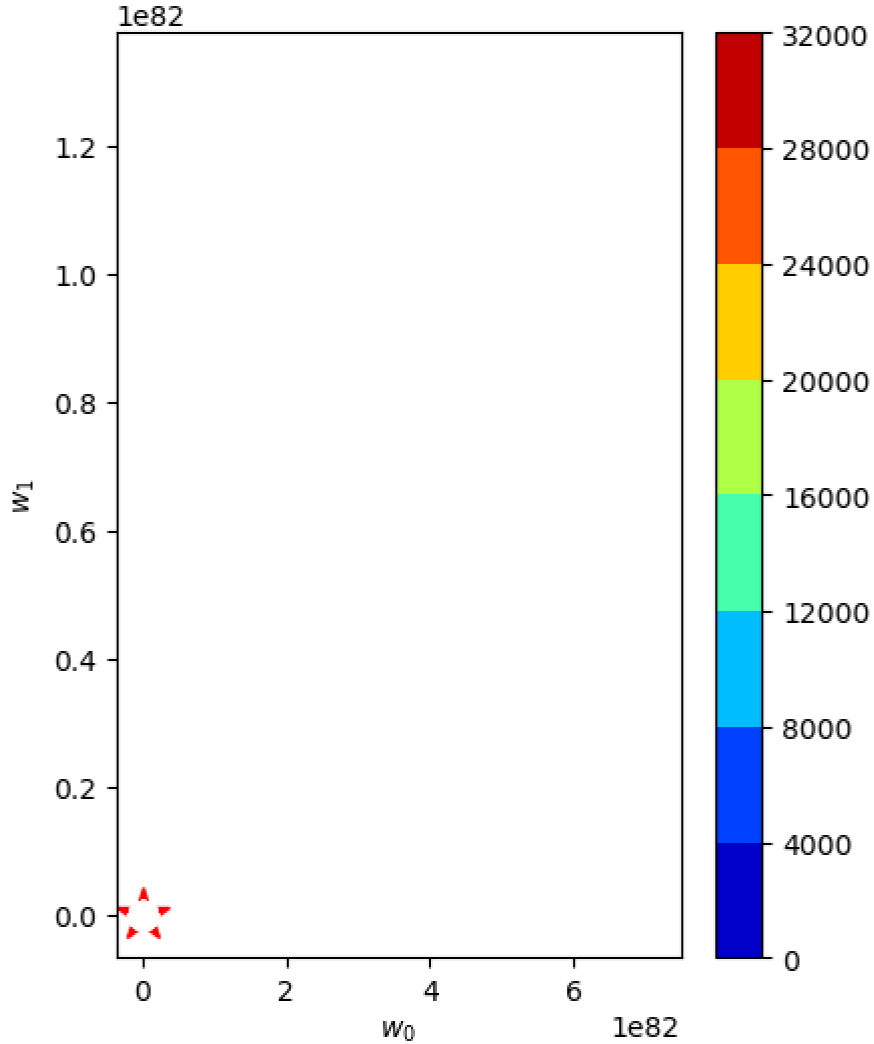

1e76
1.2
1.0
0.8
0.6
0.4
0.2
0.0
w_1
0
2
4
6
w_0
1e76
32000
28000
24000
20000
16000
12000
8000
4000
0

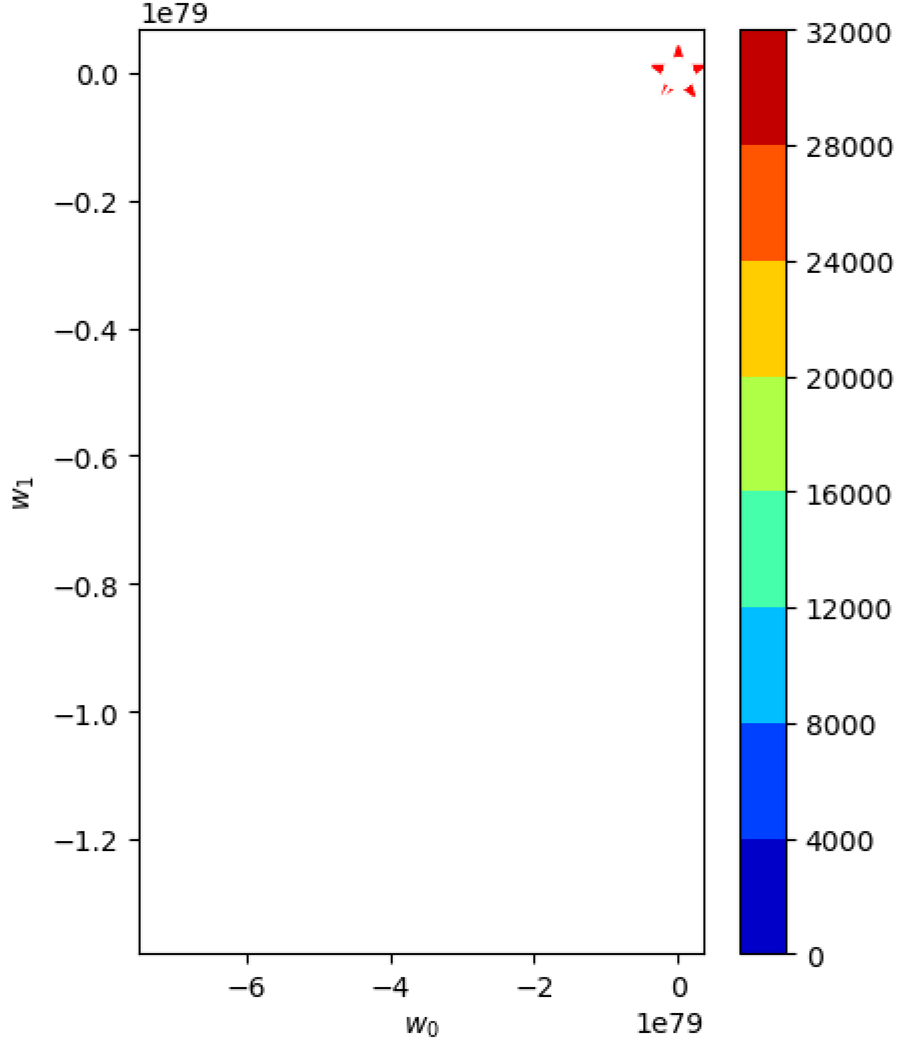

1e79
0.0
−0.2
−0.4
−0.6
−0.8
−1.0
−1.2
w_1
−6
−4
−2
0
1e79
w_0
32000
28000
24000
20000
16000
12000
8000
4000
0

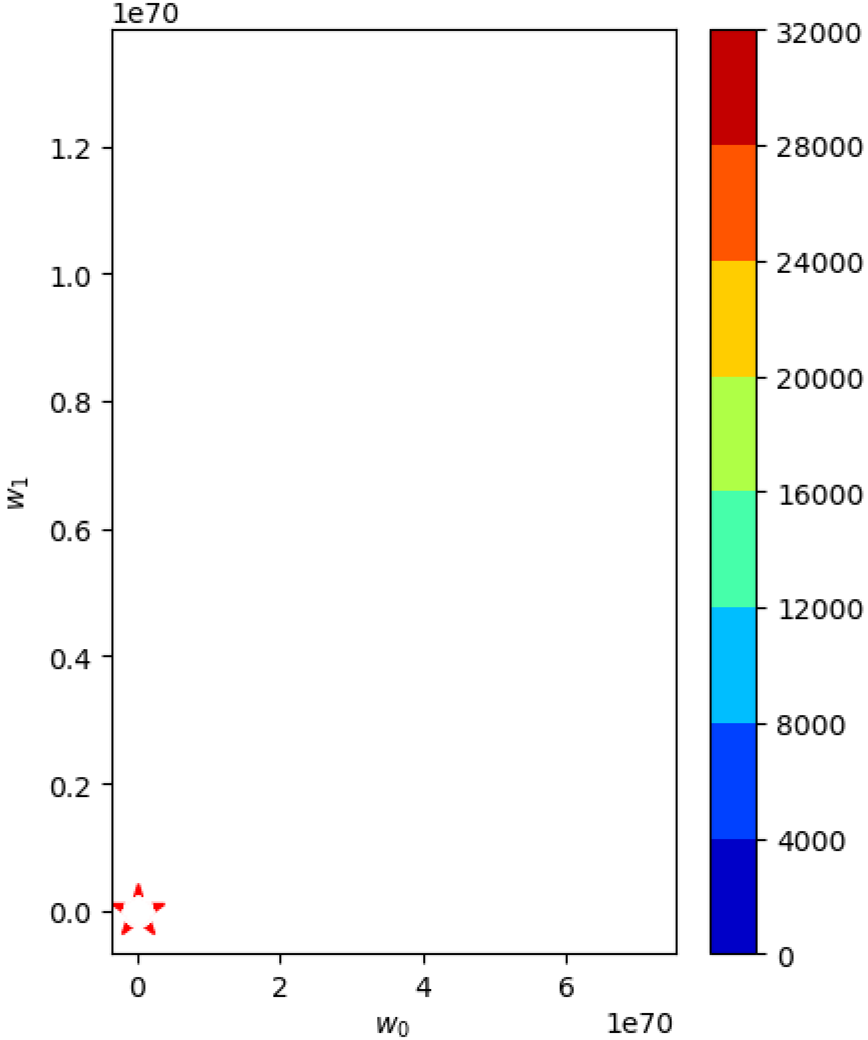


1e82
1.2
1.0
0.8
0.6
0.4
0.2
0.0
w_1
0
2
4
6
w_0
1e82
32000
28000
24000
20000
16000
12000
8000
4000
0

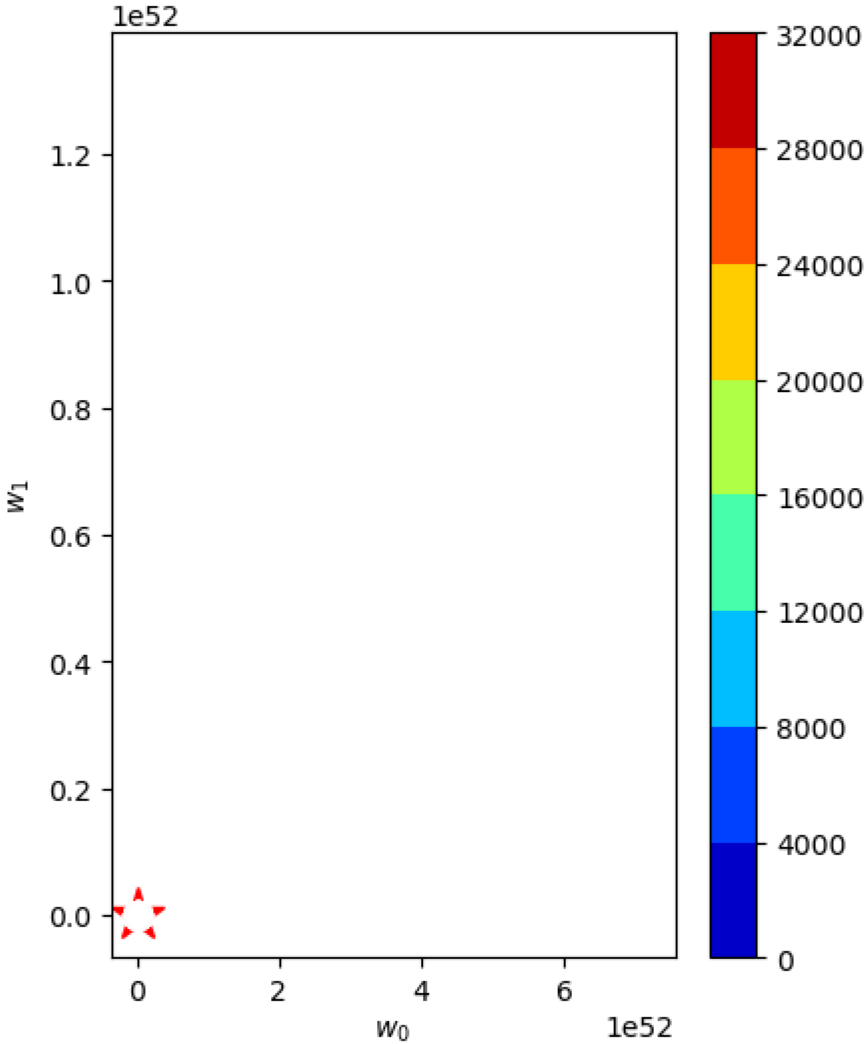

1e85
0.0
−0.2
−0.4
−0.6
−0.8
−1.0
−1.2
w_1
−6
−4
−2
0
w_0
1e85
32000
28000
24000
20000
16000
12000
8000
4000
0

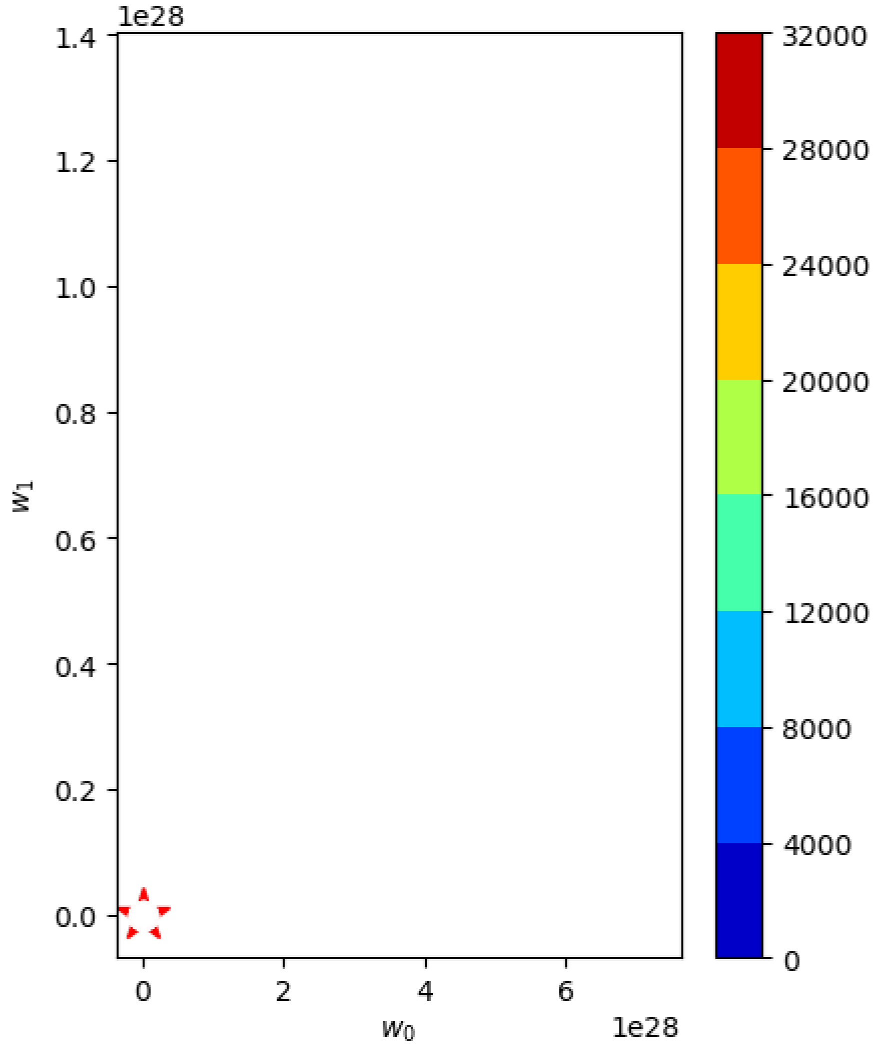

1e82
1.2
1.0
0.8
0.6
0.4
0.2
0.0
w_1
0
2
4
6
w_0
1e82
32000
28000
24000
20000
16000
12000
8000
4000
0

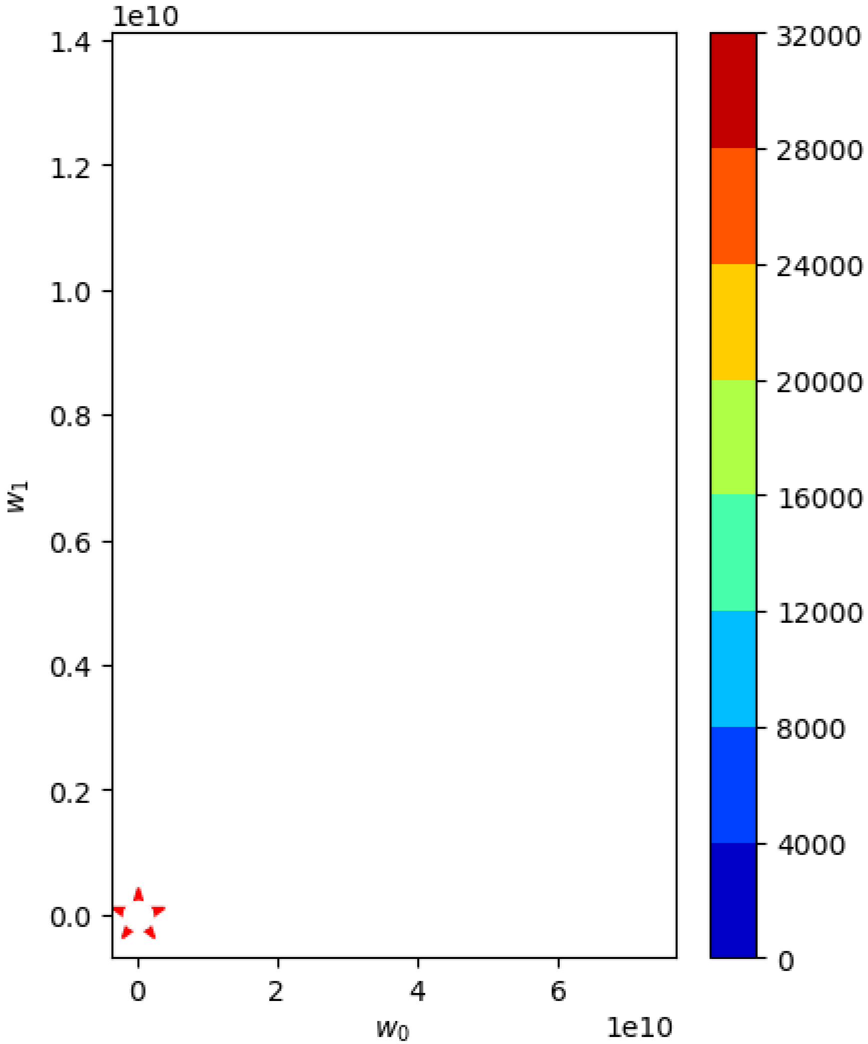

1e79
0.0
−0.2
−0.4
−0.6
−0.8
−1.0
−1.2
w_1
−6
−4
−2
0
1e79
w_0
32000
28000
24000
20000
16000
12000
8000
4000
0

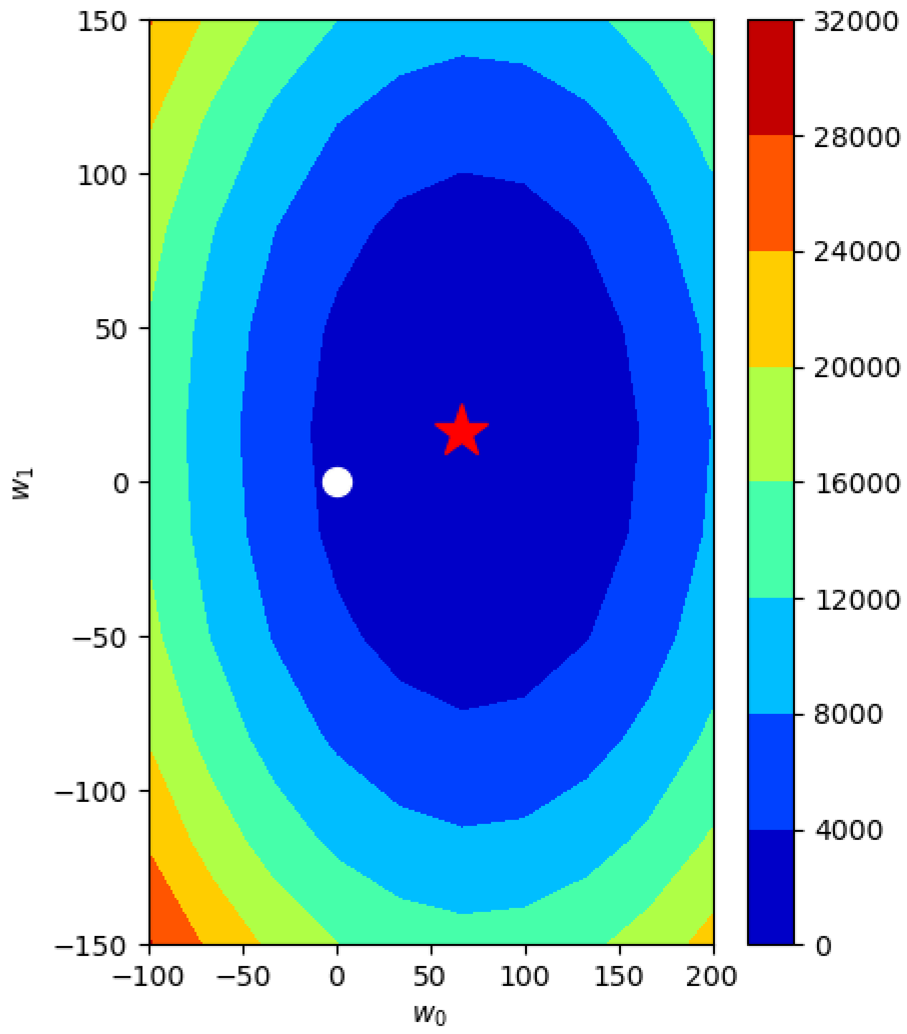


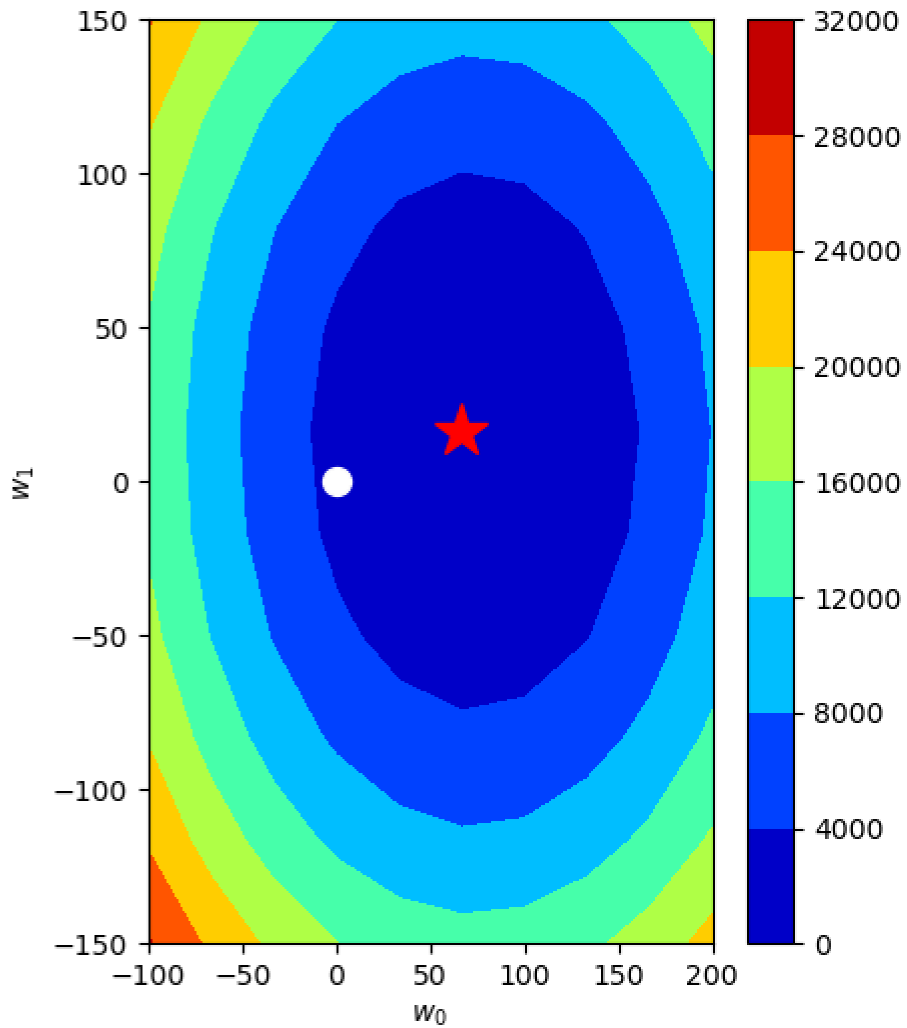
1e70
1.2
1.0
0.8
0.6
0.4
0.2
0.0
w_1
0
2
4
6
w_0
1e70
32000
28000
24000
20000
16000
12000
8000
4000
0

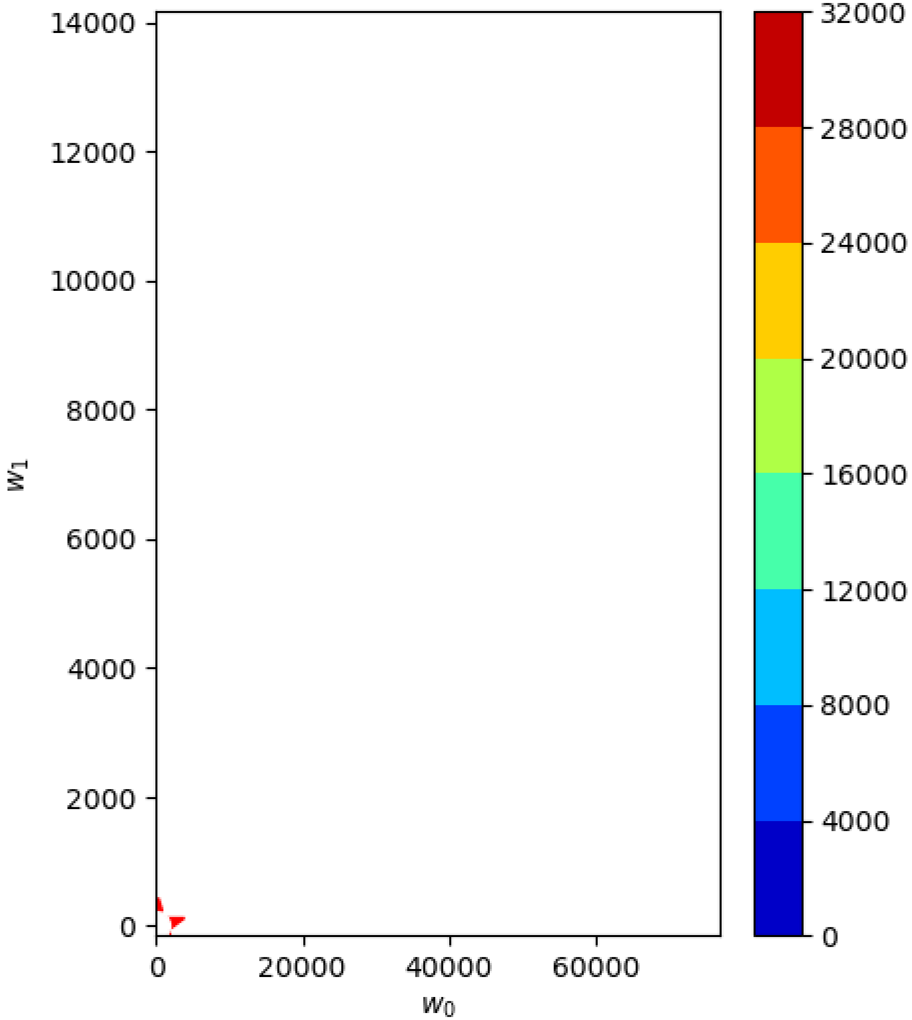

1e52
1.2
1.0
0.8
0.6
0.4
0.2
0.0
w_1
0
2
4
6
w_0
1e52
32000
28000
24000
20000
16000
12000
8000
4000
0


1e28
1.4
1.2
1.0
0.8
0.6
0.4
0.2
0.0
w_1
0
2
4
6
w_0
1e28
32000
28000
24000
20000
16000
12000
8000
4000
0

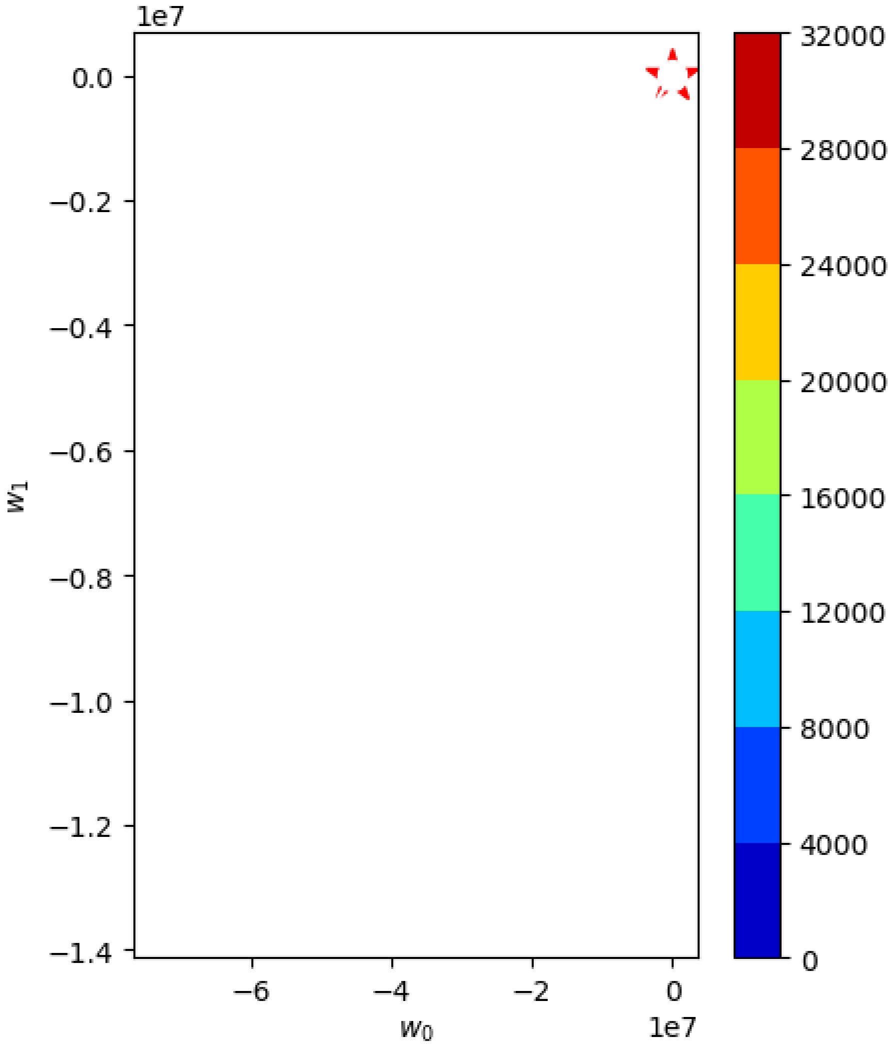

1e10
1.4
1.2
1.0
0.8
0.6
0.4
0.2
0.0
w_1
0
2
4
6
w_0
1e10
32000
28000
24000
20000
16000
12000
8000
4000
0

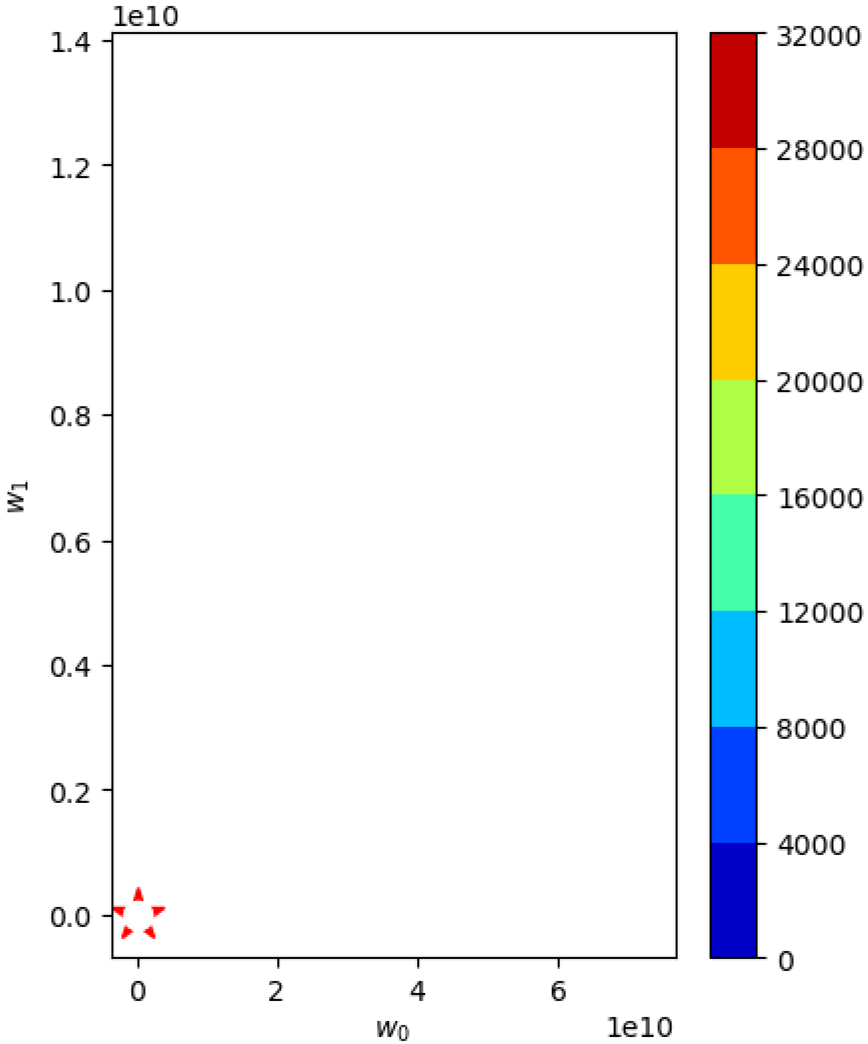

150
100
50
0
−50
−100
−150
w_1
−100
−50
0
50
100
150
200
w_0
32000
28000
24000
20000
16000
12000
8000
4000
0


150
100
50
0
−50
−100
−150
w_1
−100
−50
0
50
100
150
200
w_0
32000
28000
24000
20000
16000
12000
8000
4000
0

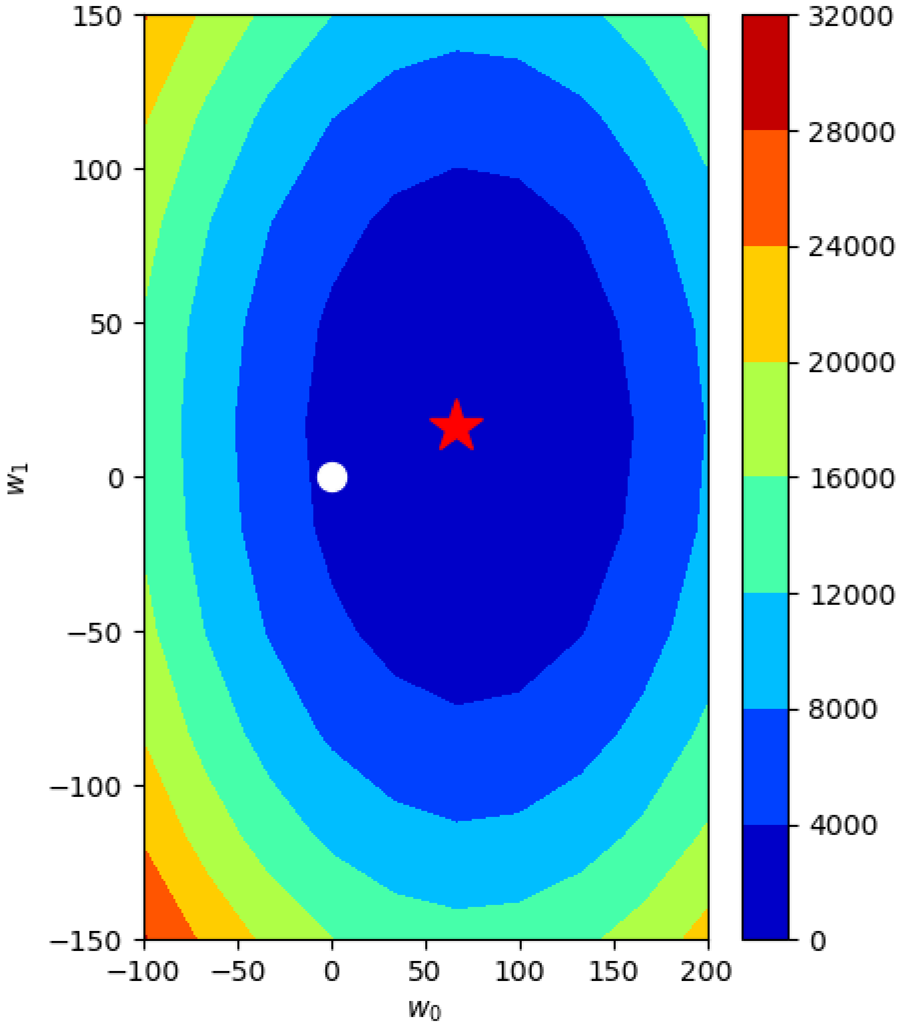


14000
12000
10000
8000
6000
4000
2000
0
w_1
0
20000
40000
60000
w_0
32000
28000
24000
20000
16000
12000
8000
4000
0

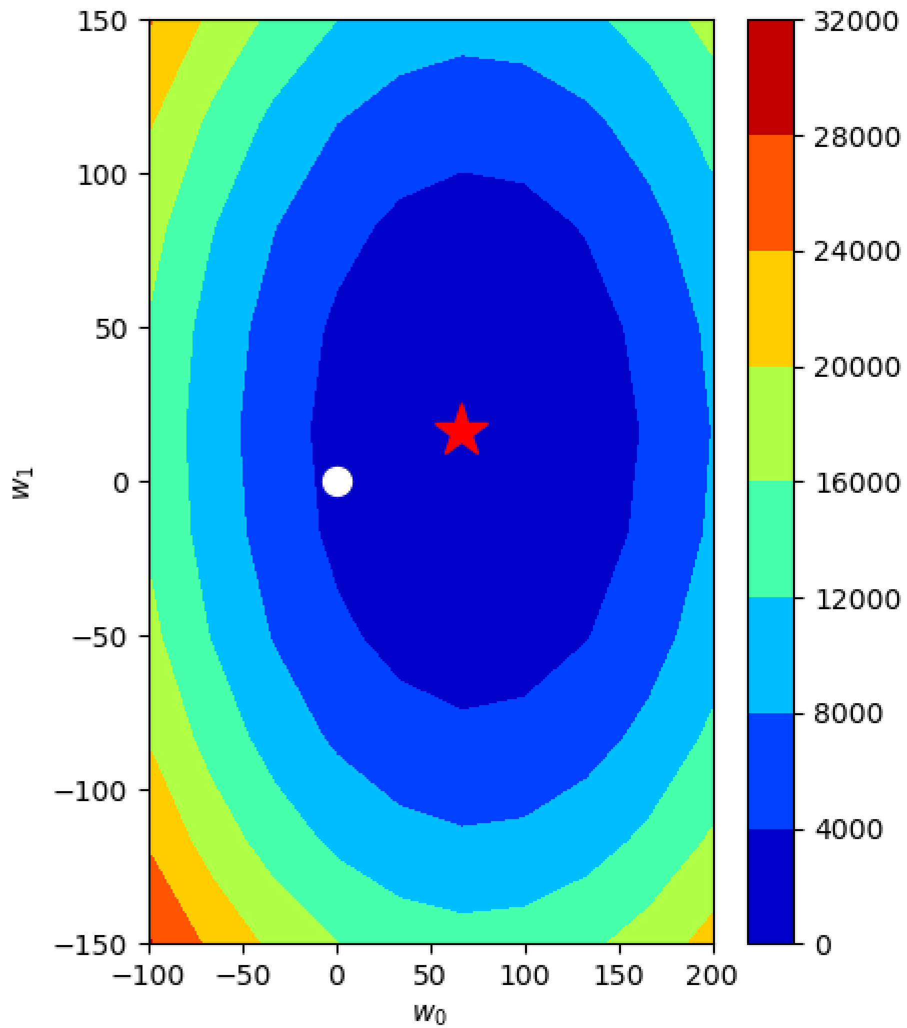

1e7
0.0
−0.2
−0.4
−0.6
−0.8
−1.0
−1.2
−1.4
w_1
−6
−4
−2
0
1e7
w_0
32000
28000
24000
20000
16000
12000
8000
4000
0



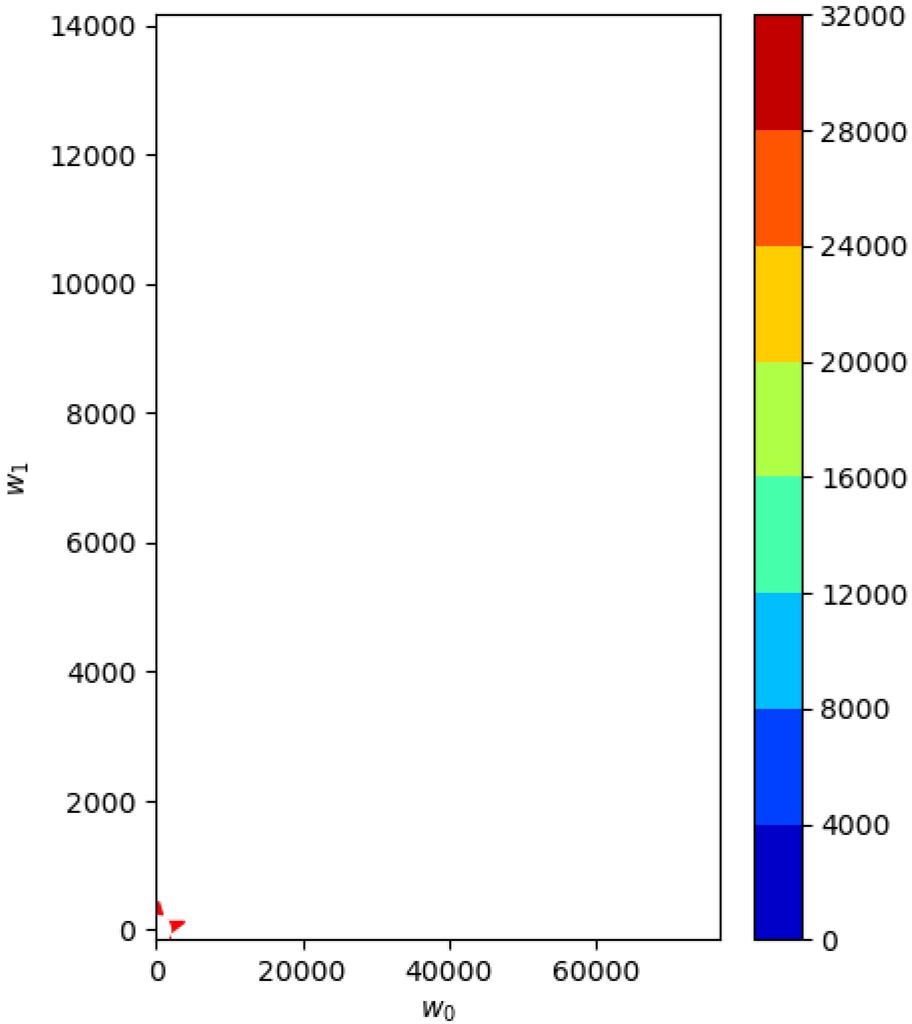
1e10
1.4
1.2
1.0
0.8
0.6
0.4
0.2
0.0
w_1
0
2
4
6
w_0
1e10
32000
28000
24000
20000
16000
12000
8000
4000
0



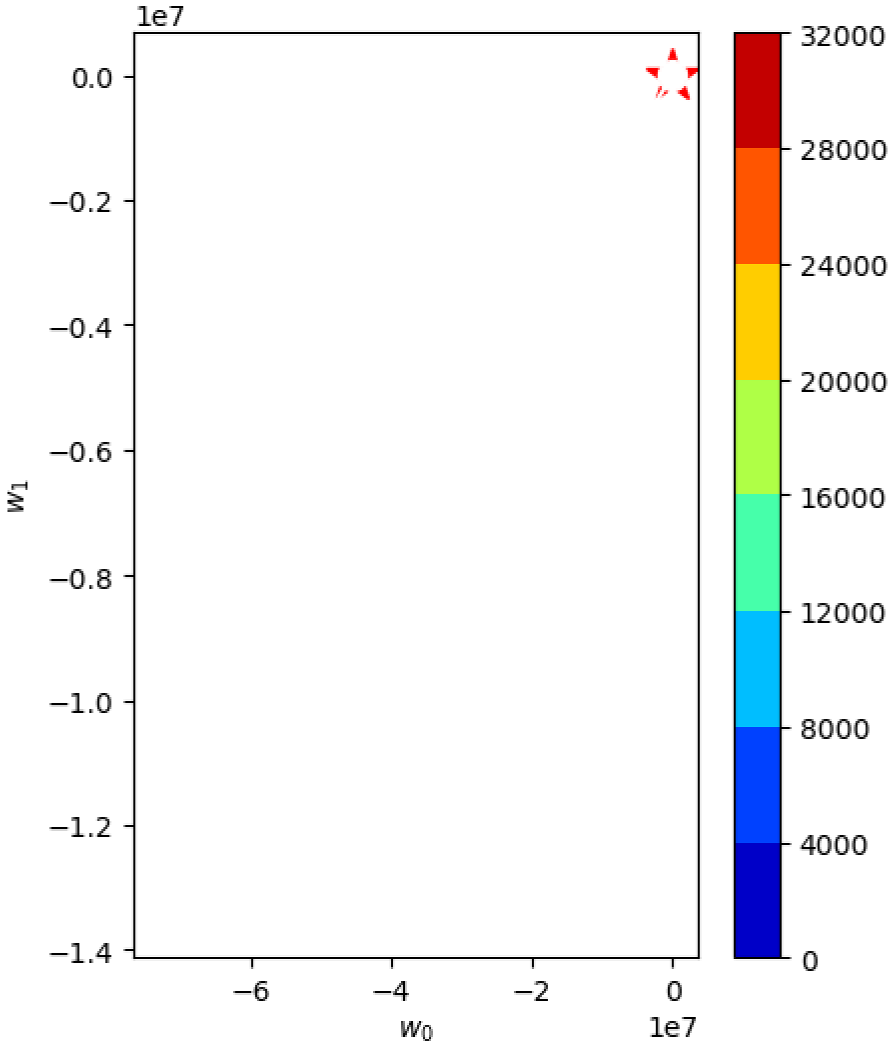
150
100
50
0
−50
−100
−150
w_1
−100
−50
0
50
100
150
200
w_0
32000
28000
24000
20000
16000
12000
8000
4000
0

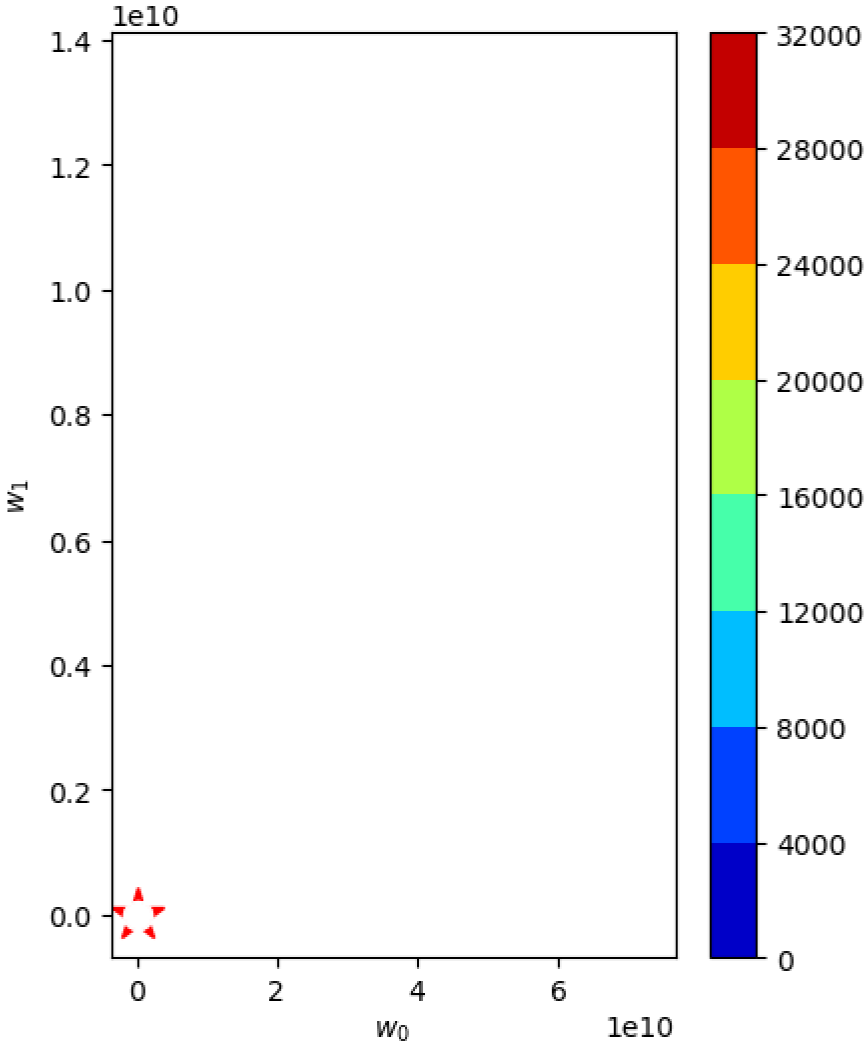


150
100
50
0
−50
−100
−150
w_1
−100
−50
0
50
100
150
200
w_0
32000
28000
24000
20000
16000
12000
8000
4000
0

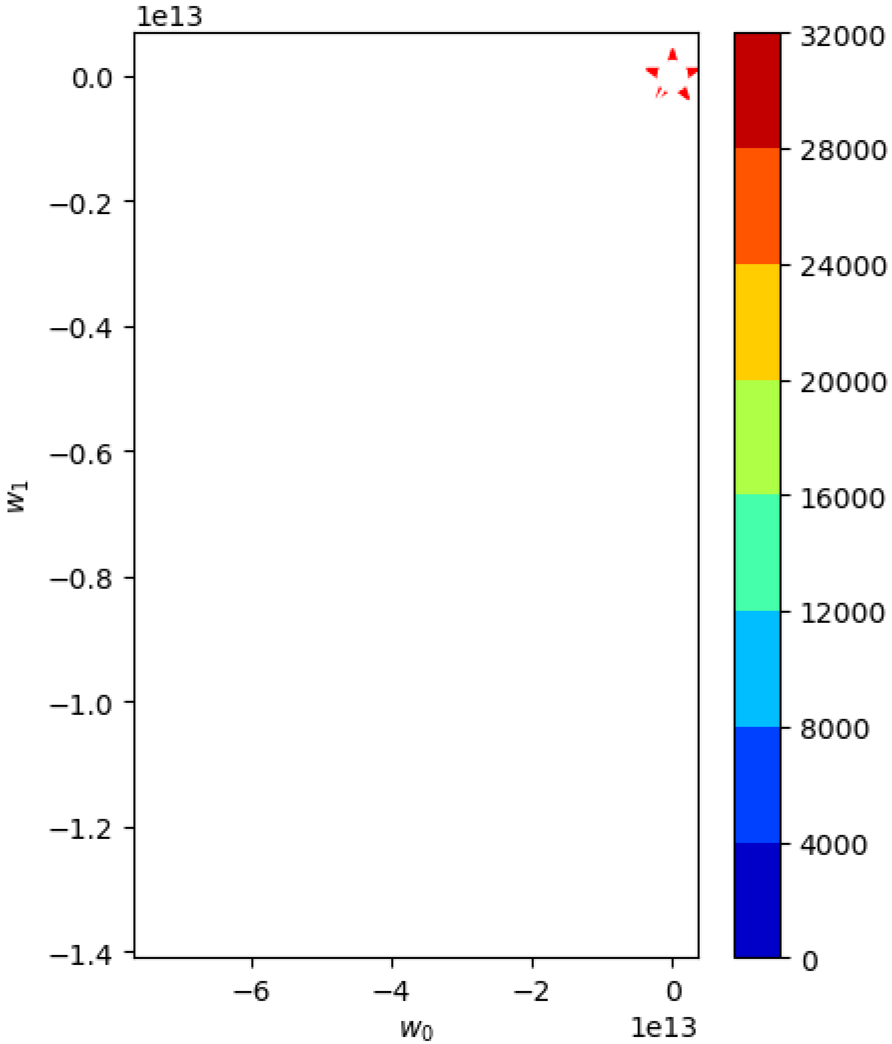


14000
12000
10000
8000
6000
4000
2000
0
w_1
0
20000
40000
60000
w_0
32000
28000
24000
20000
16000
12000
8000
4000
0

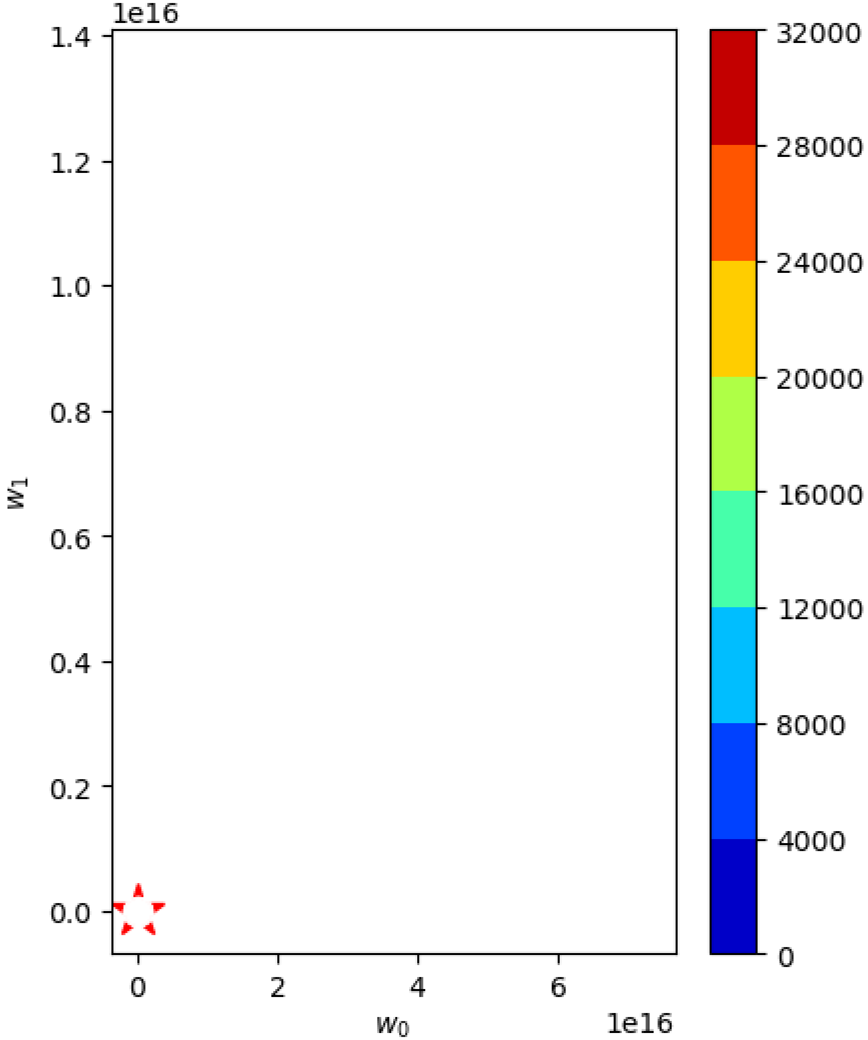

1e7
0.0
−0.2
−0.4
−0.6
−0.8
−1.0
−1.2
−1.4
w_1
−6
−4
−2
0
1e7
w_0
32000
28000
24000
20000
16000
12000
8000
4000
0



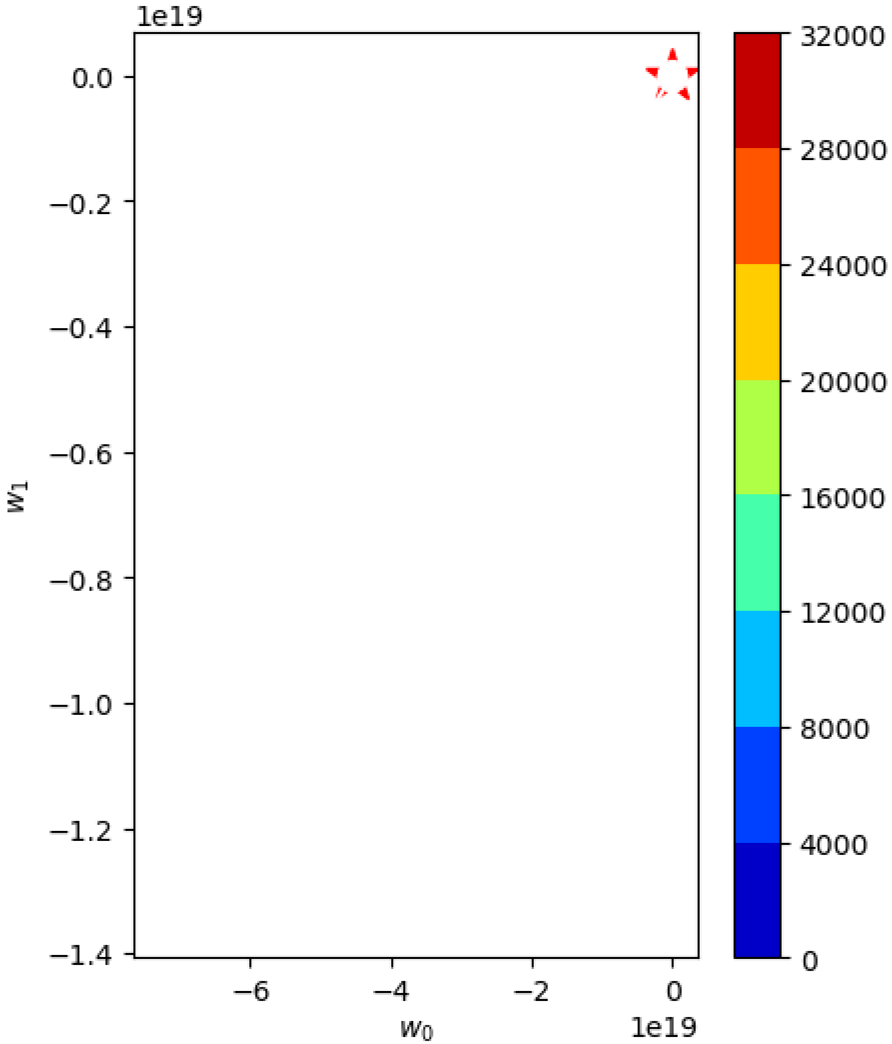
1e10
1.4
1.2
1.0
0.8
0.6
0.4
0.2
0.0
w_1
0
2
4
6
w_0
1e10
32000
28000
24000
20000
16000
12000
8000
4000
0



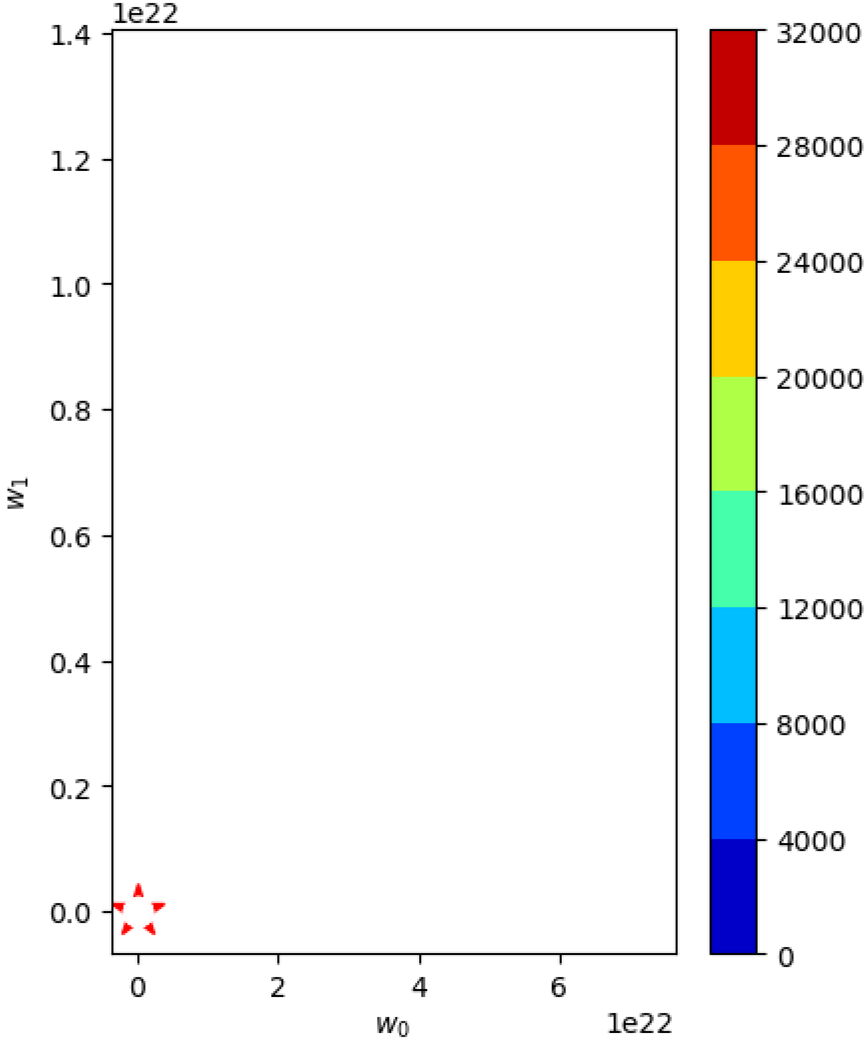
1e13
0.0
−0.2
−0.4
−0.6
−0.8
−1.0
−1.2
−1.4
w_1
−6
−4
−2
0
w_0
1e13
32000
28000
24000
20000
16000
12000
8000
4000
0

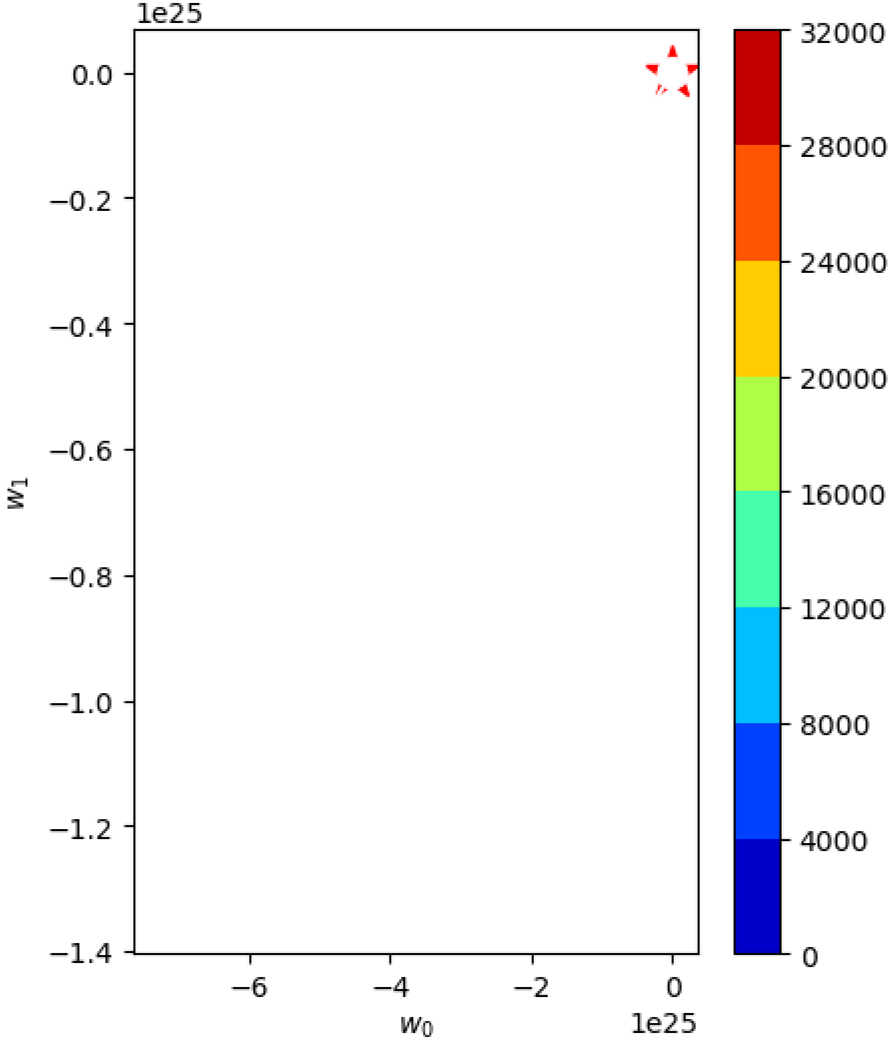


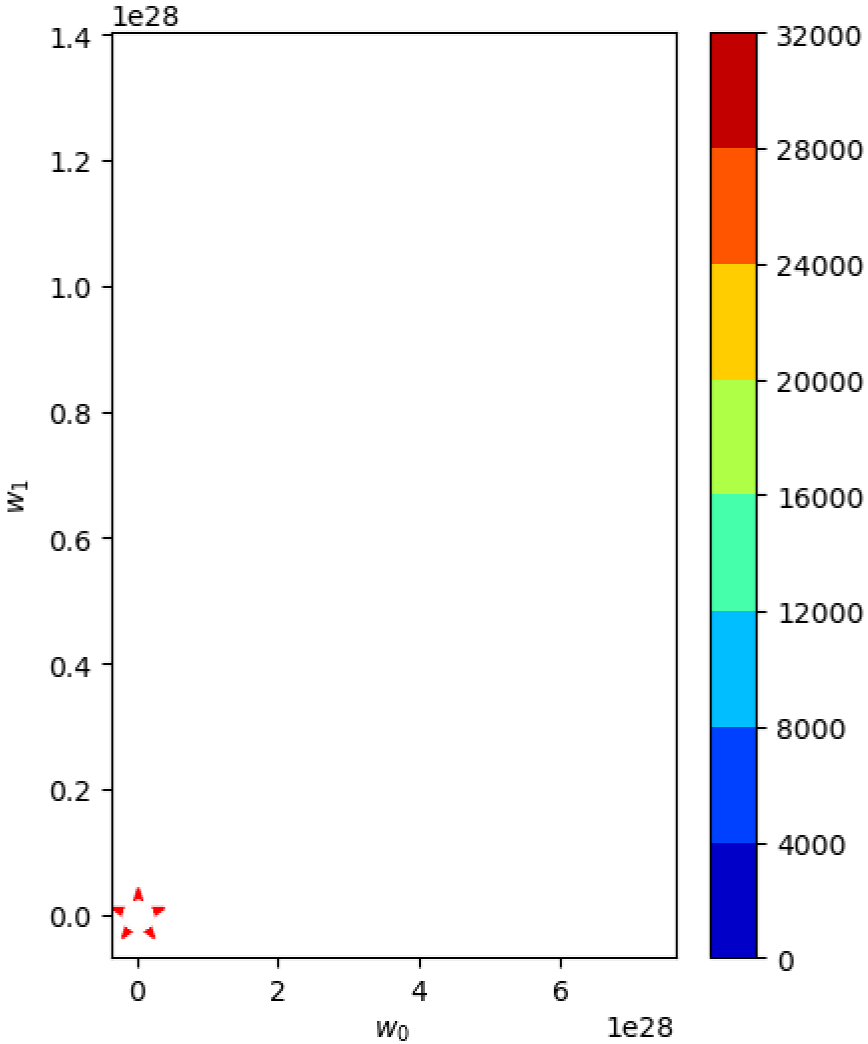
1e16
1.4
1.2
1.0
0.8
0.6
0.4
0.2
0.0
w_1
0
2
4
6
w_0
1e16
32000
28000
24000
20000
16000
12000
8000
4000
0



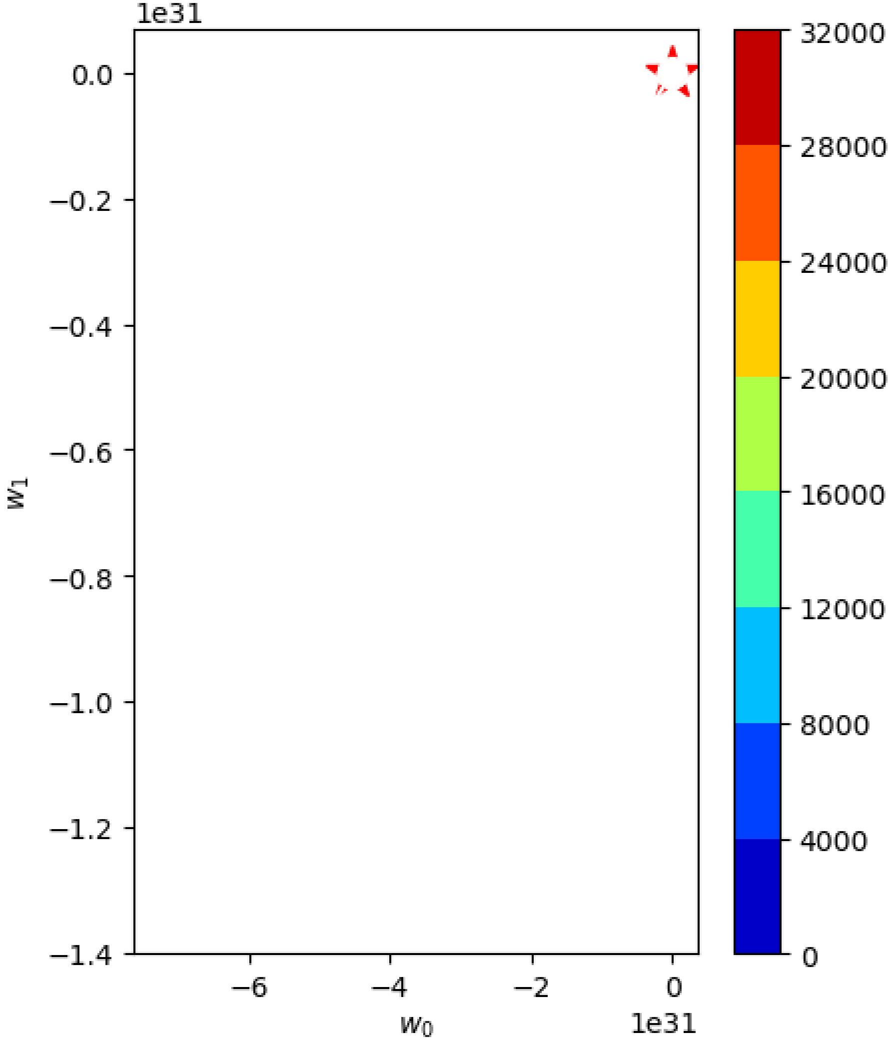
1e19
0.0
−0.2
−0.4
−0.6
−0.8
−1.0
−1.2
−1.4
w_1
−6
−4
−2
0
w_0
1e19
32000
28000
24000
20000
16000
12000
8000
4000
0

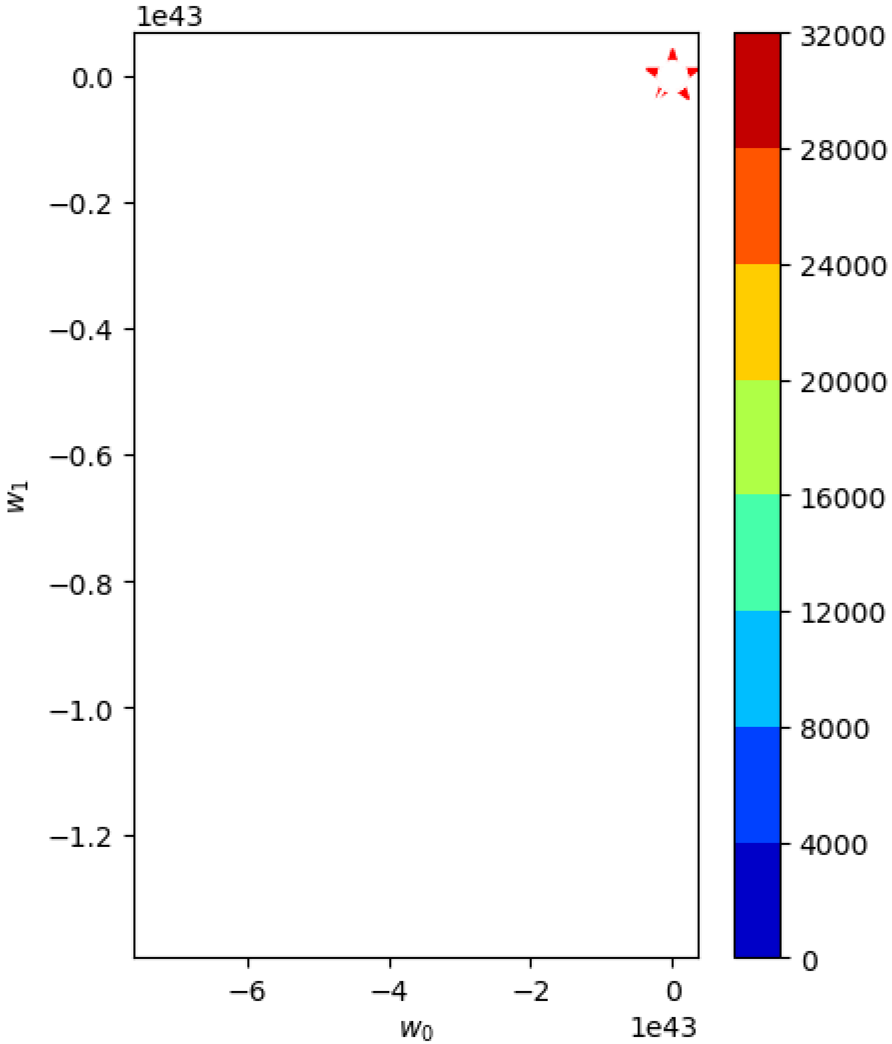

1e22
1.4
1.2
1.0
0.8
0.6
0.4
0.2
0.0
w_1
0
2
4
6
w_0
1e22
32000
28000
24000
20000
16000
12000
8000
4000
0

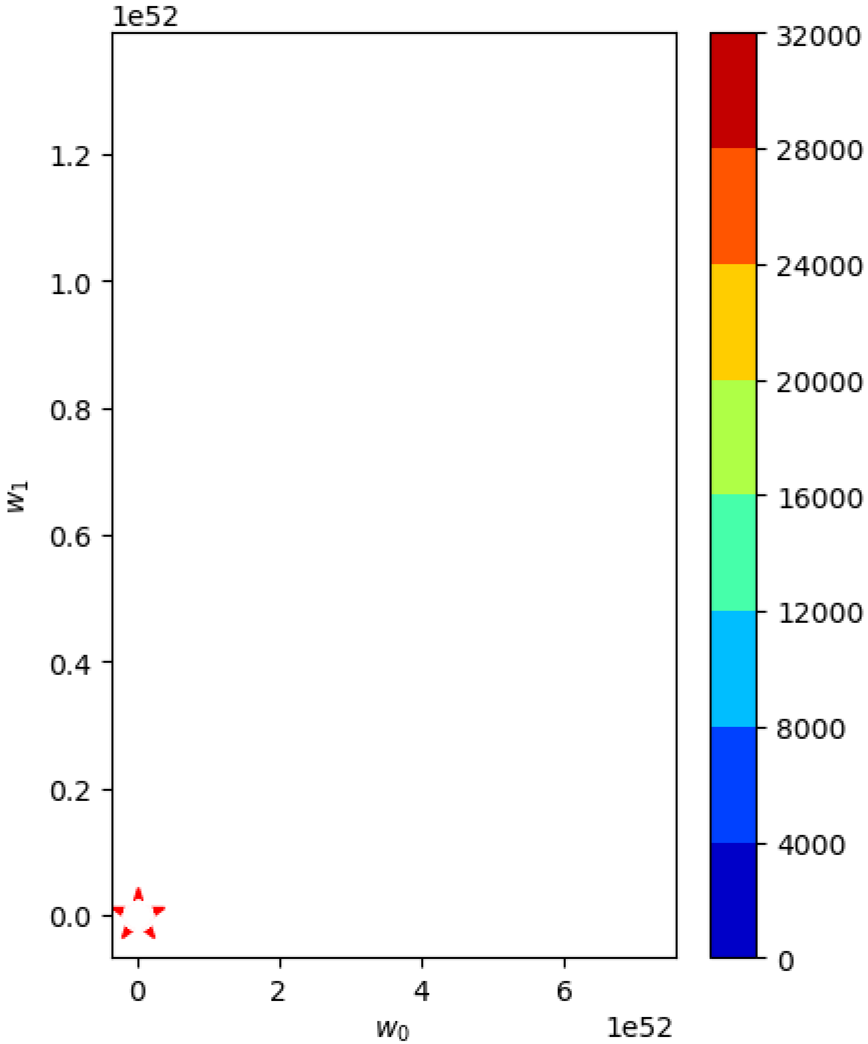

1e25
0.0
−0.2
−0.4
−0.6
−0.8
−1.0
−1.2
−1.4
w_1
−6
−4
−2
0
1e25
w_0
32000
28000
24000
20000
16000
12000
8000
4000
0



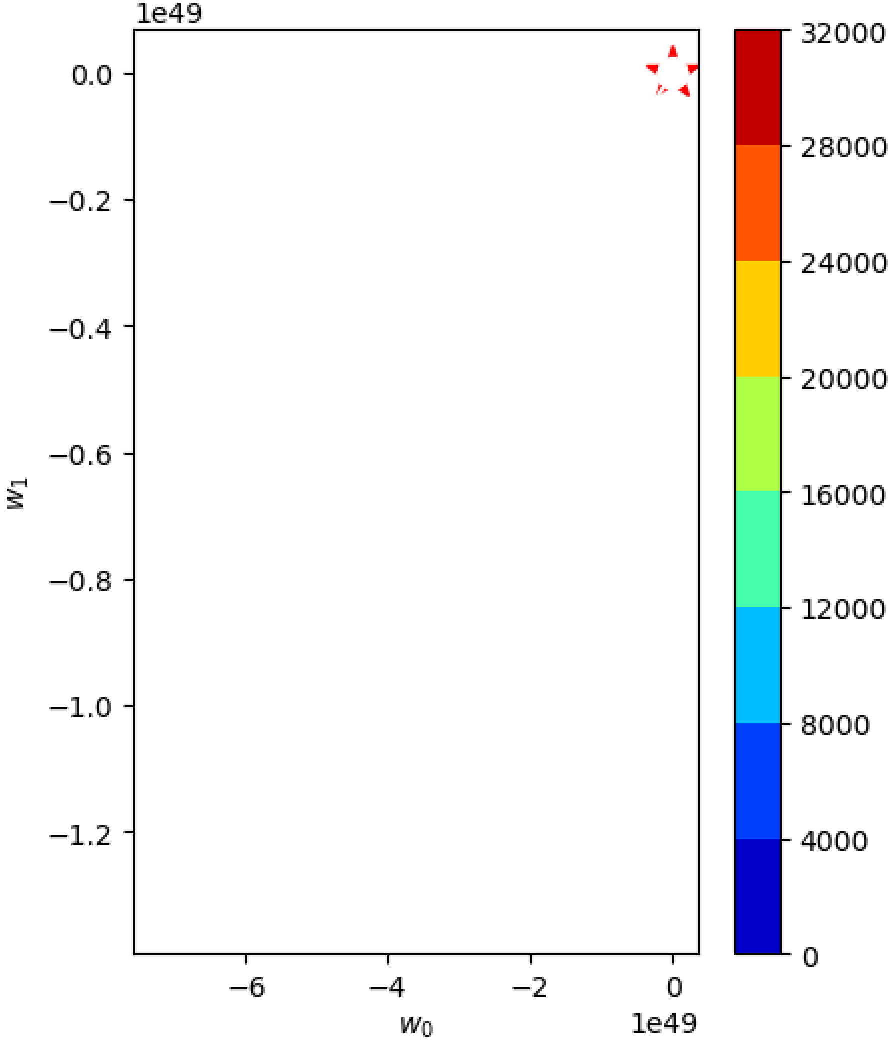
1e28
1.4
1.2
1.0
0.8
0.6
0.4
0.2
0.0
w_1
0
2
4
6
w_0
1e28
32000
28000
24000
20000
16000
12000
8000
4000
0


1e31
0.0
−0.2
−0.4
−0.6
−0.8
−1.0
−1.2
−1.4
w_1
−6
−4
−2
0
w_0
1e31
32000
28000
24000
20000
16000
12000
8000
4000
0

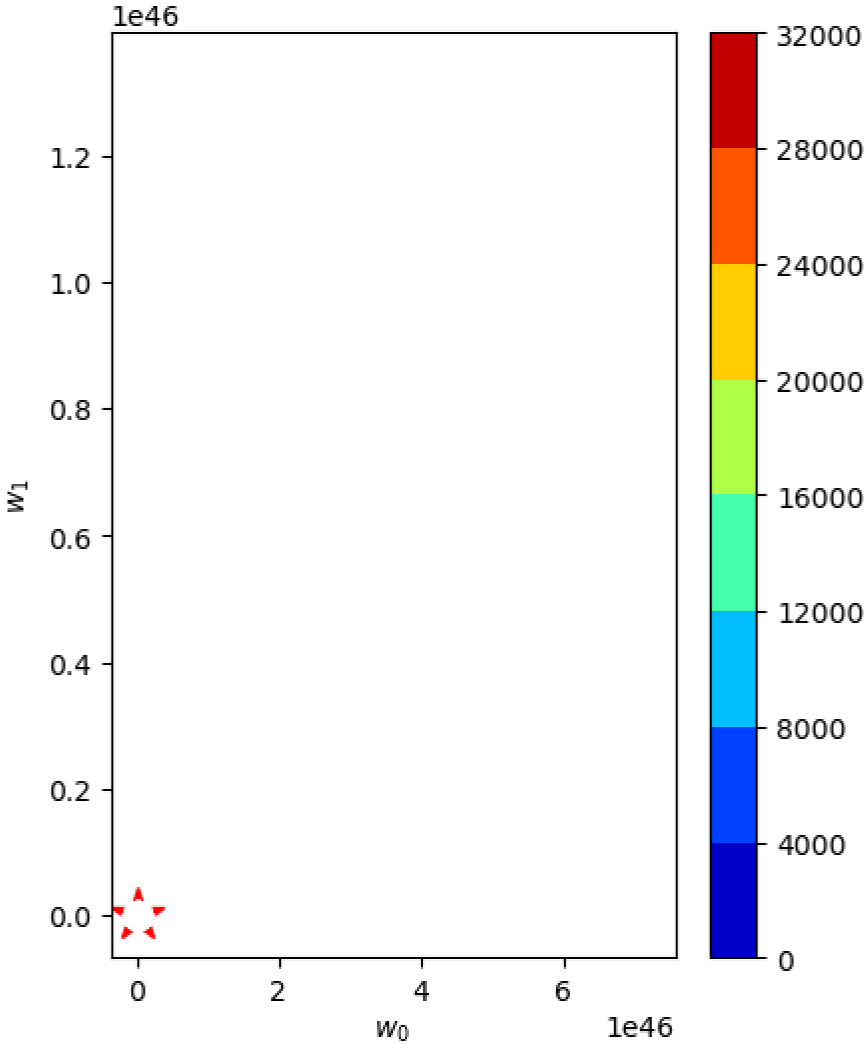

1e43
0.0
−0.2
−0.4
−0.6
−0.8
−1.0
−1.2
w_1
−6
−4
−2
0
w_0
1e43
32000
28000
24000
20000
16000
12000
8000
4000
0

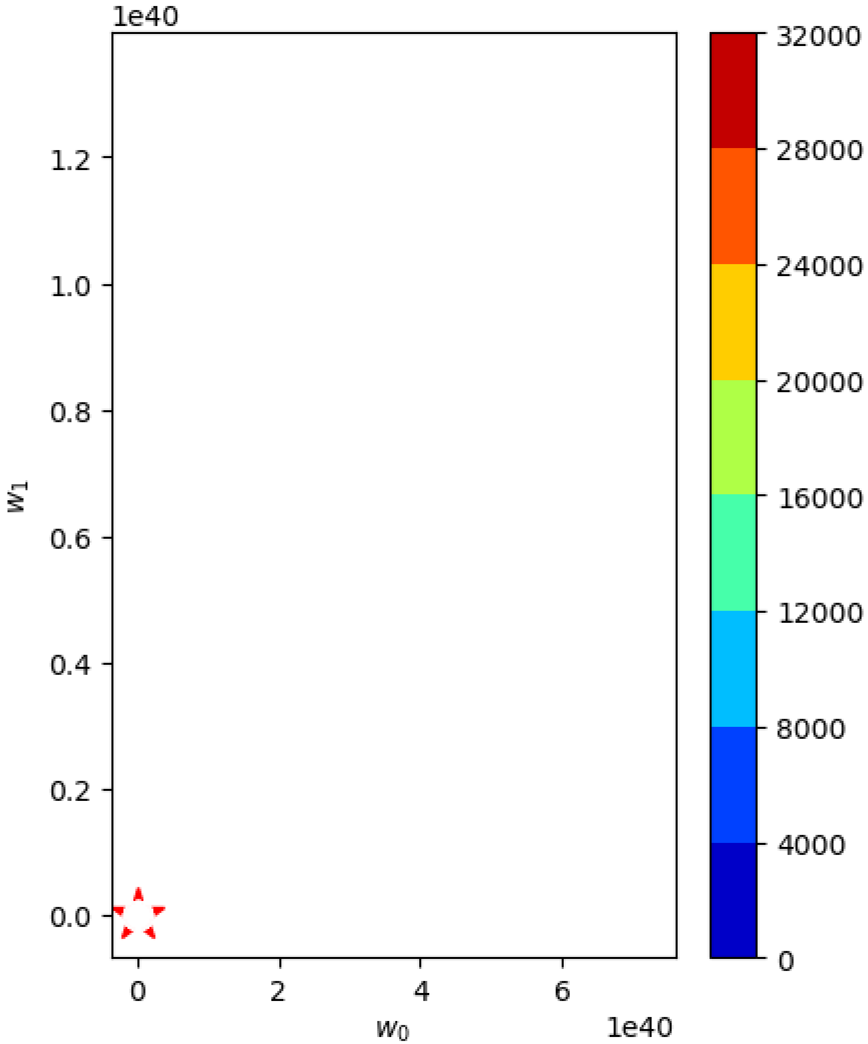


1e52
1.2
1.0
0.8
0.6
0.4
0.2
0.0
w_1
0
2
4
6
w_0
1e52
32000
28000
24000
20000
16000
12000
8000
4000
0

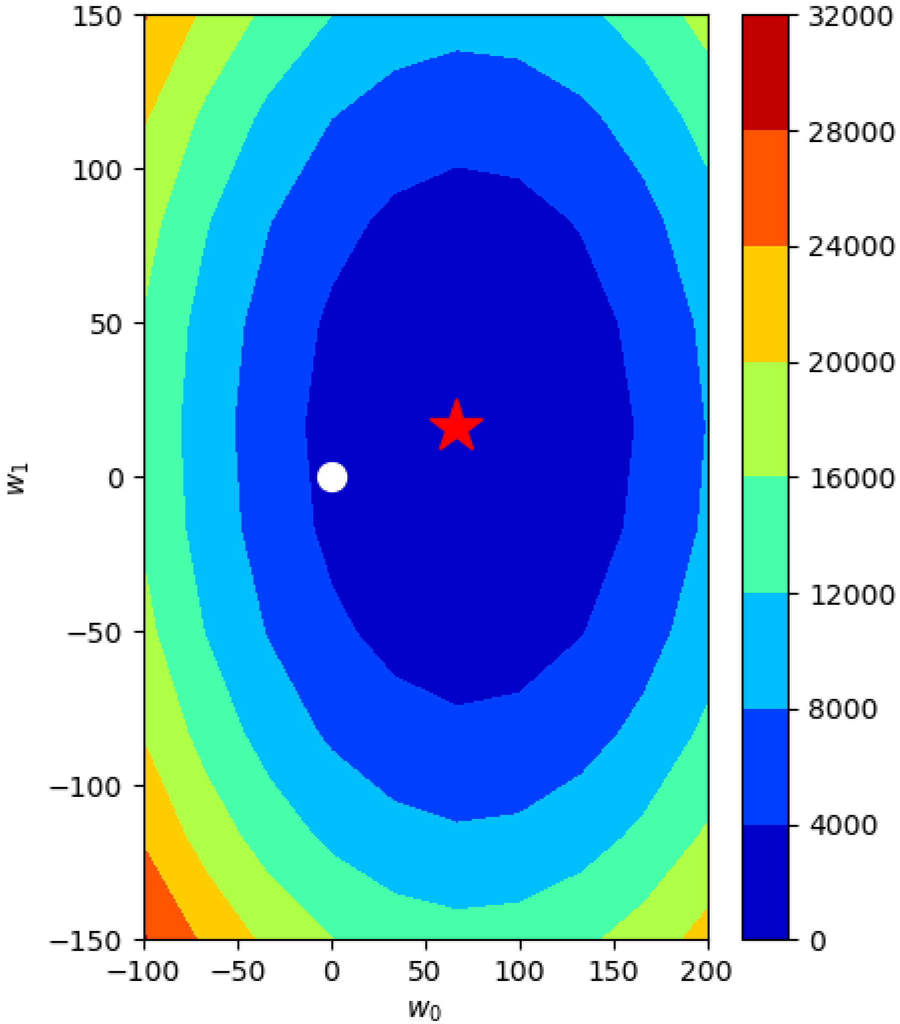


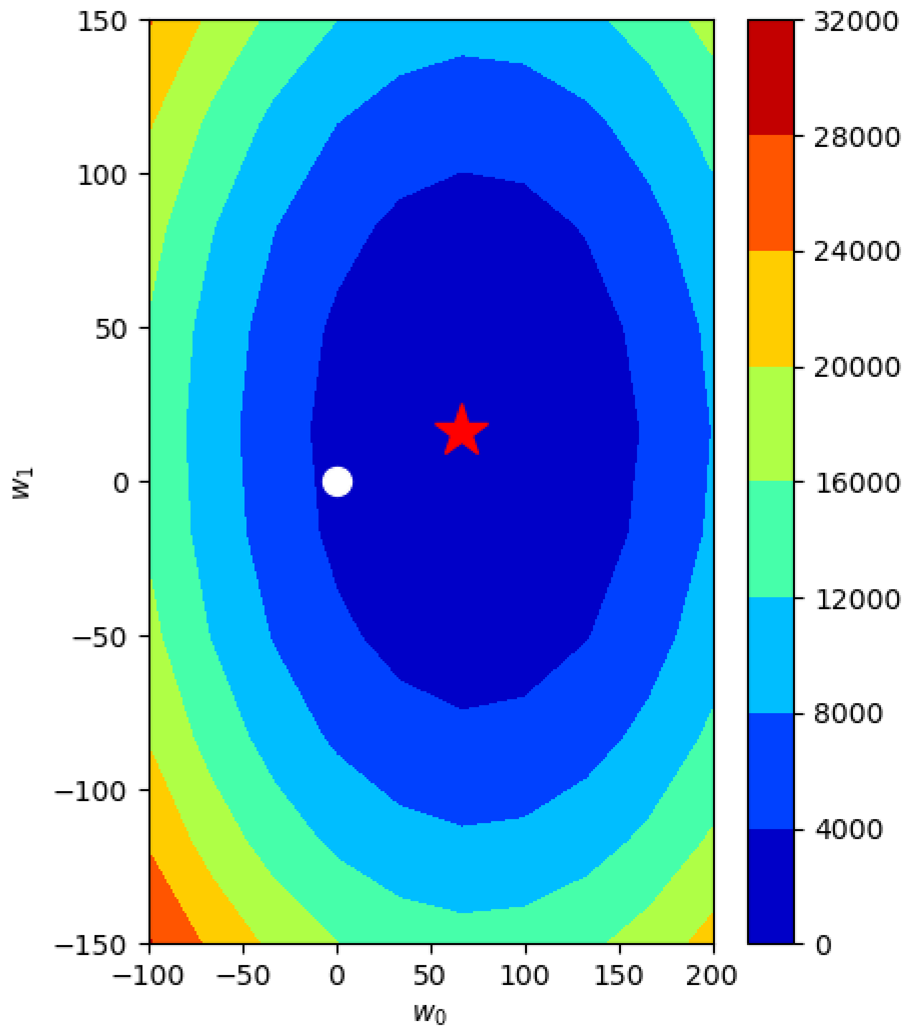
1e49
0.0
−0.2
−0.4
−0.6
−0.8
−1.0
−1.2
w_1
−6
−4
−2
0
1e49
w_0
32000
28000
24000
20000
16000
12000
8000
4000
0


1e46
1.2
1.0
0.8
0.6
0.4
0.2
0.0
w_1
0
2
4
6
w_0
1e46
32000
28000
24000
20000
16000
12000
8000
4000
0

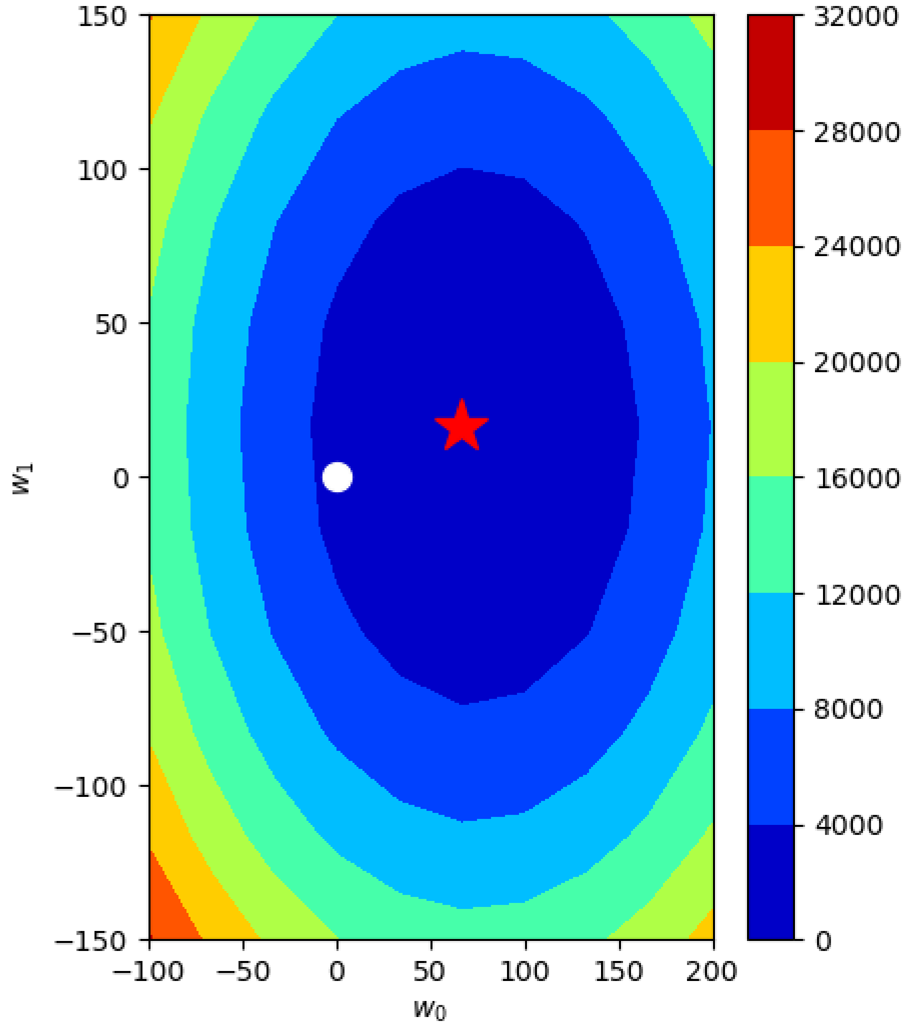


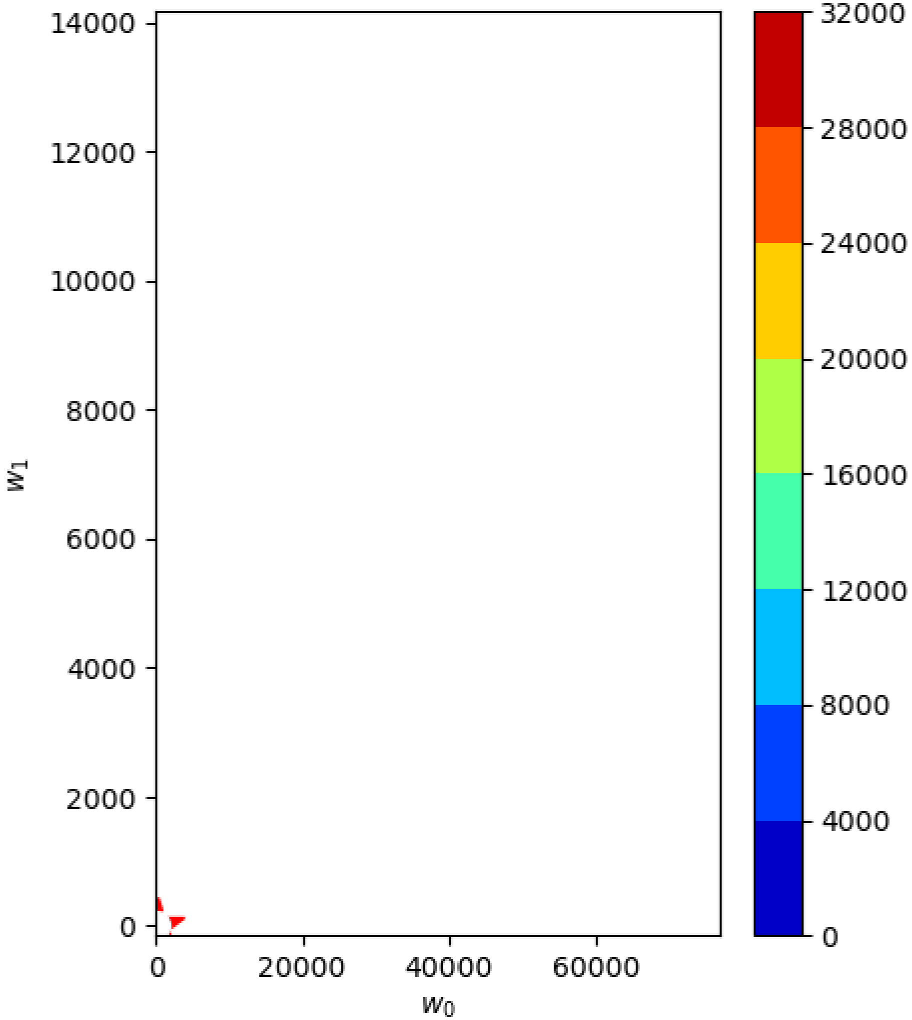
1e40
1.2
1.0
0.8
0.6
0.4
0.2
0.0
w_1
0
2
4
6
w_0
1e40
32000
28000
24000
20000
16000
12000
8000
4000
0


150
100
50
0
−50
−100
−150
w_1
−100
−50
0
50
100
150
200
w_0
32000
28000
24000
20000
16000
12000
8000
4000
0

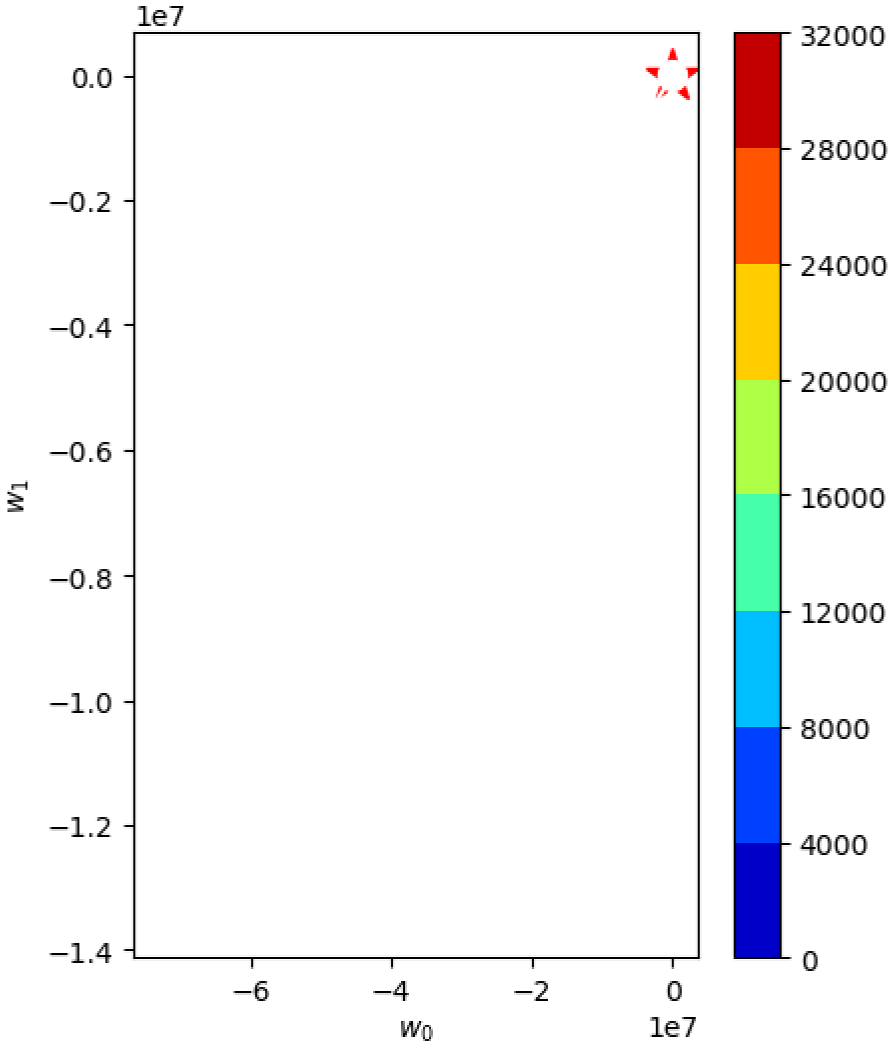


150
100
50
0
−50
−100
−150
w_1
−100
−50
0
50
100
150
200
w_0
32000
28000
24000
20000
16000
12000
8000
4000
0

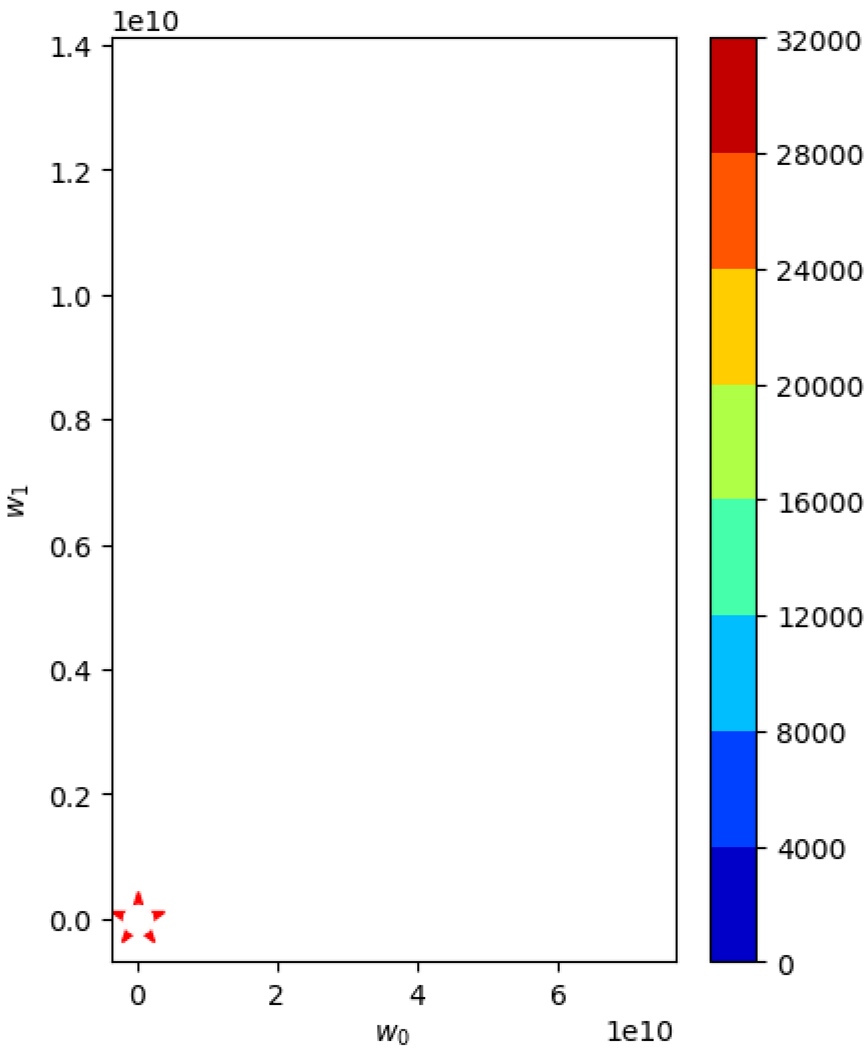

150
100
50
0
−50
−100
−150
w_1
−100
−50
0
50
100
150
200
w_0
32000
28000
24000
20000
16000
12000
8000
4000
0

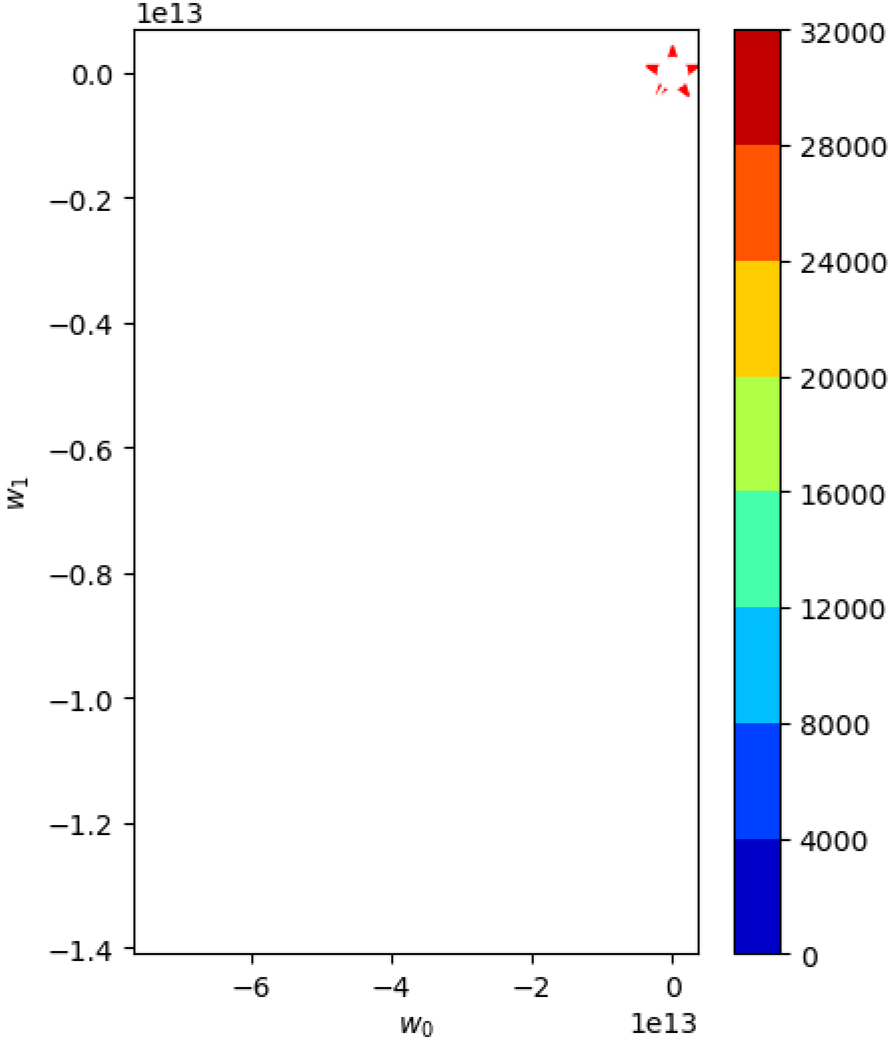


14000
12000
10000
8000
6000
4000
2000
0
w_1
0
20000
40000
60000
w_0
32000
28000
24000
20000
16000
12000
8000
4000
0

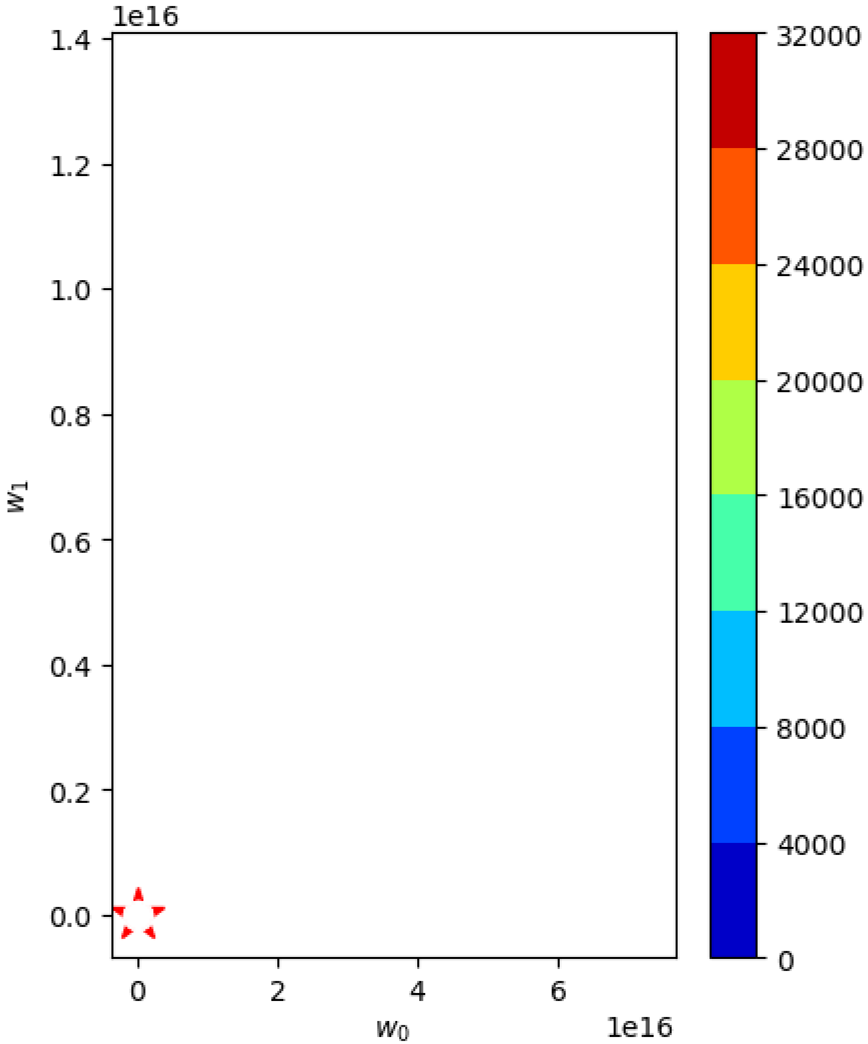


1e7
0.0
−0.2
−0.4
−0.6
−0.8
−1.0
−1.2
−1.4
w_1
−6
−4
−2
0
1e7
w_0
32000
28000
24000
20000
16000
12000
8000
4000
0

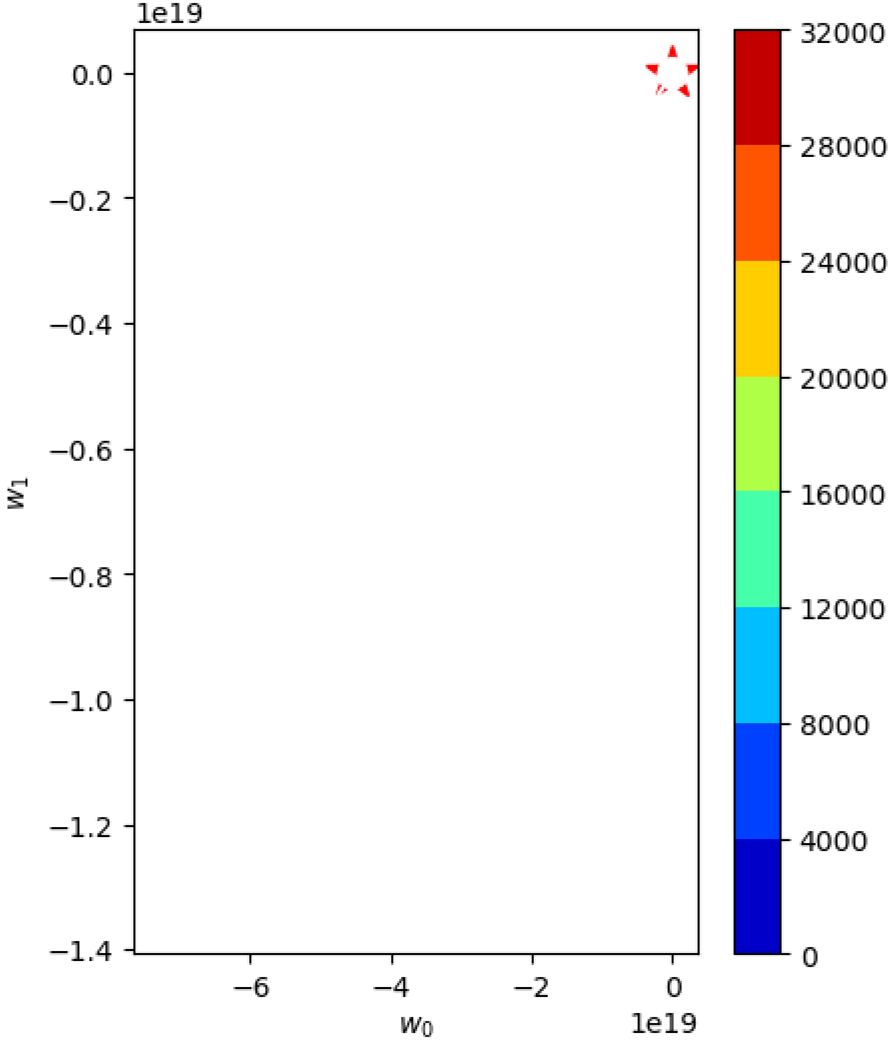

1e10
1.4
1.2
1.0
0.8
0.6
0.4
0.2
0.0
w_1
0
2
4
6
w_0
1e10
32000
28000
24000
20000
16000
12000
8000
4000
0



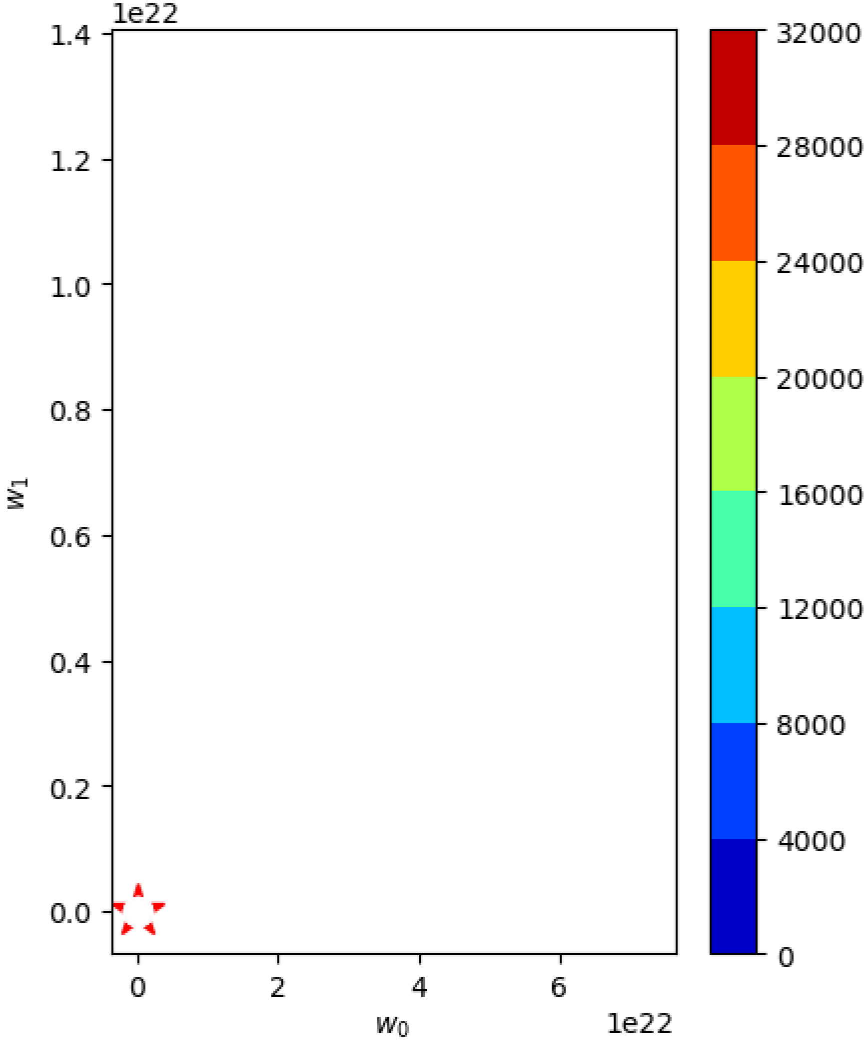
1e13
0.0
−0.2
−0.4
−0.6
−0.8
−1.0
−1.2
−1.4
w_1
−6
−4
−2
0
1e13
w_0
32000
28000
24000
20000
16000
12000
8000
4000
0

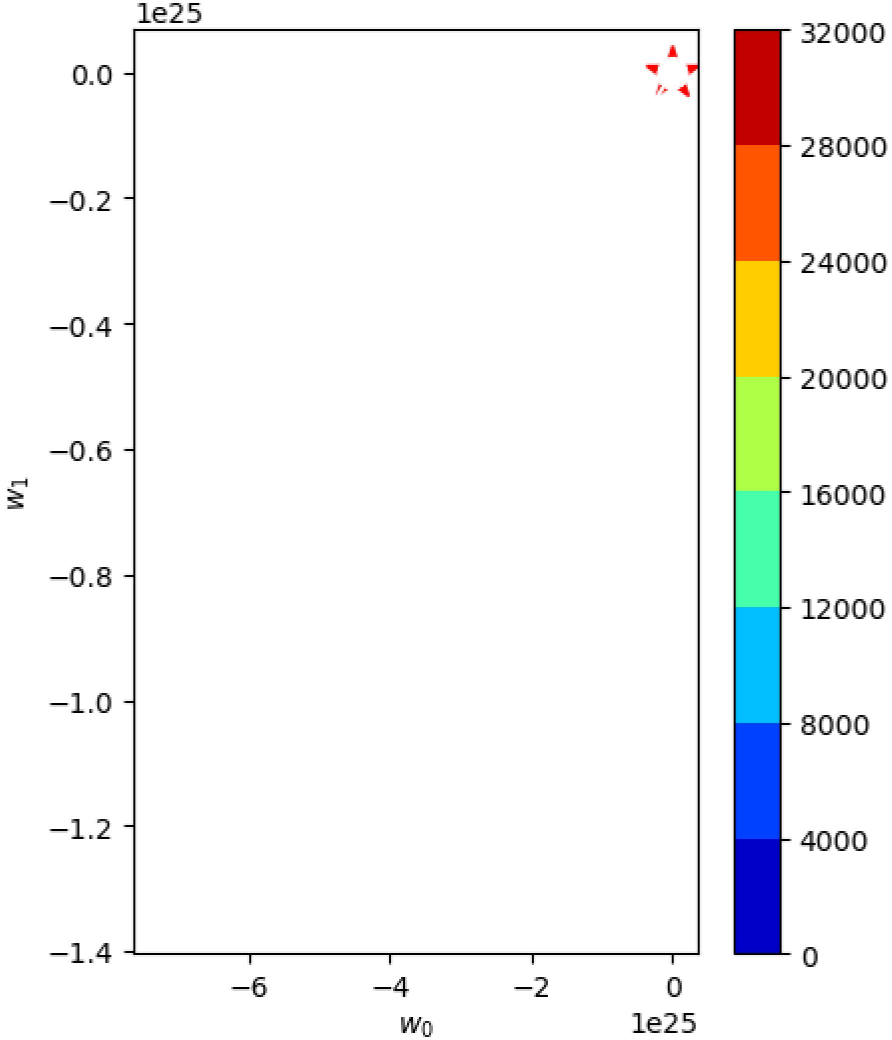


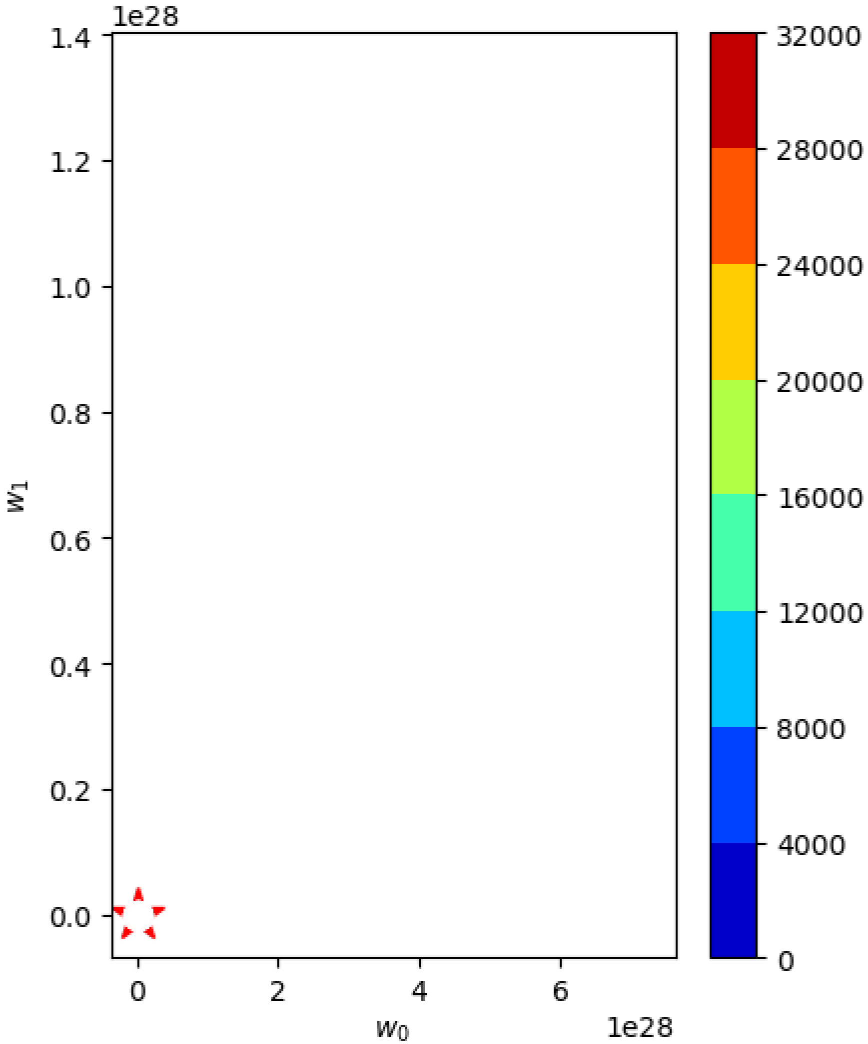
1e16
1.4
1.2
1.0
0.8
0.6
0.4
0.2
0.0
w_1
0
2
4
6
w_0
1e16
32000
28000
24000
20000
16000
12000
8000
4000
0



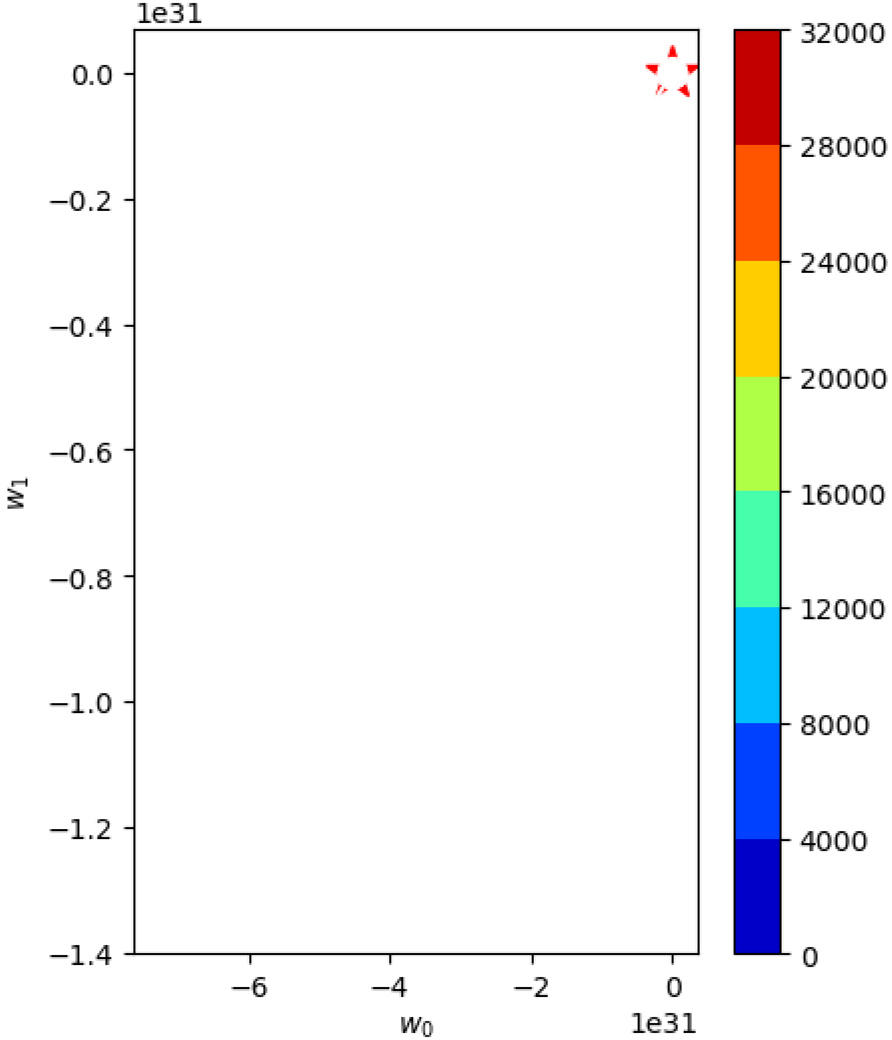
1e19
0.0
−0.2
−0.4
−0.6
−0.8
−1.0
−1.2
−1.4
w_1
−6
−4
−2
0
1e19
w_0
32000
28000
24000
20000
16000
12000
8000
4000
0

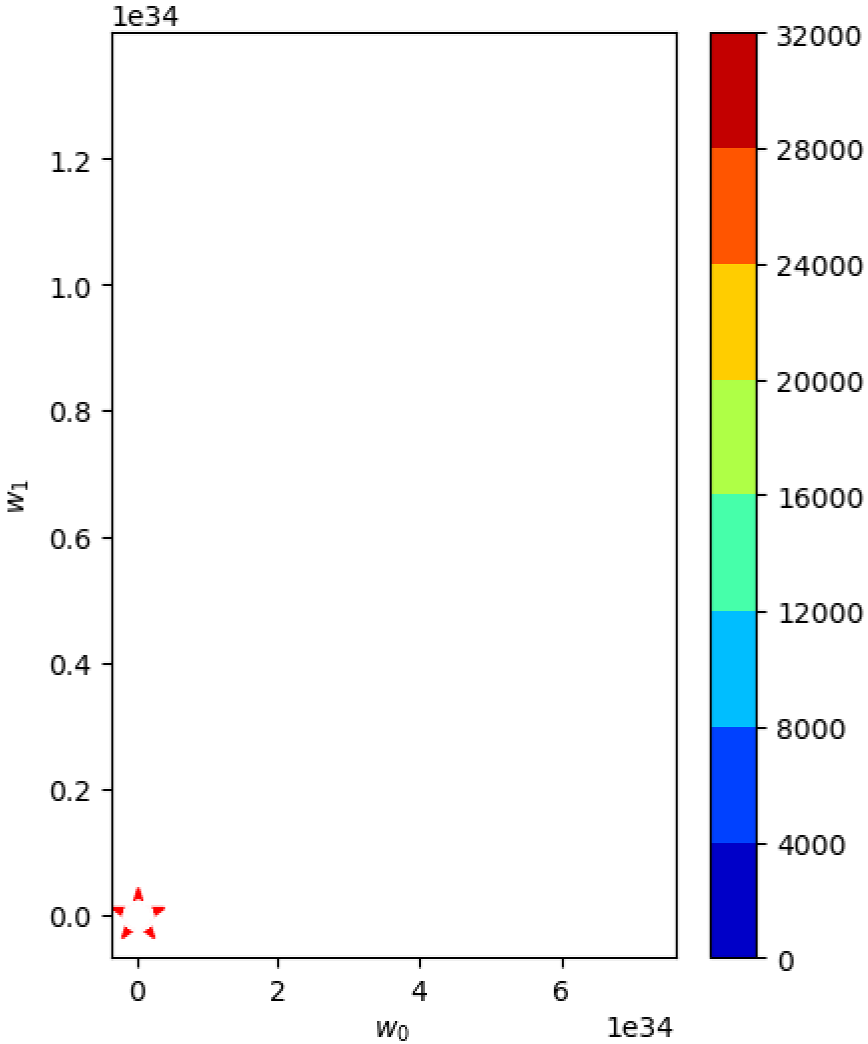

1e22
1.4
1.2
1.0
0.8
0.6
0.4
0.2
0.0
w_1
0
2
4
6
w_0
1e22
32000
28000
24000
20000
16000
12000
8000
4000
0


1e25
0.0
−0.2
−0.4
−0.6
−0.8
−1.0
−1.2
−1.4
w_1
−6
−4
−2
0
1e25
w_0
32000
28000
24000
20000
16000
12000
8000
4000
0

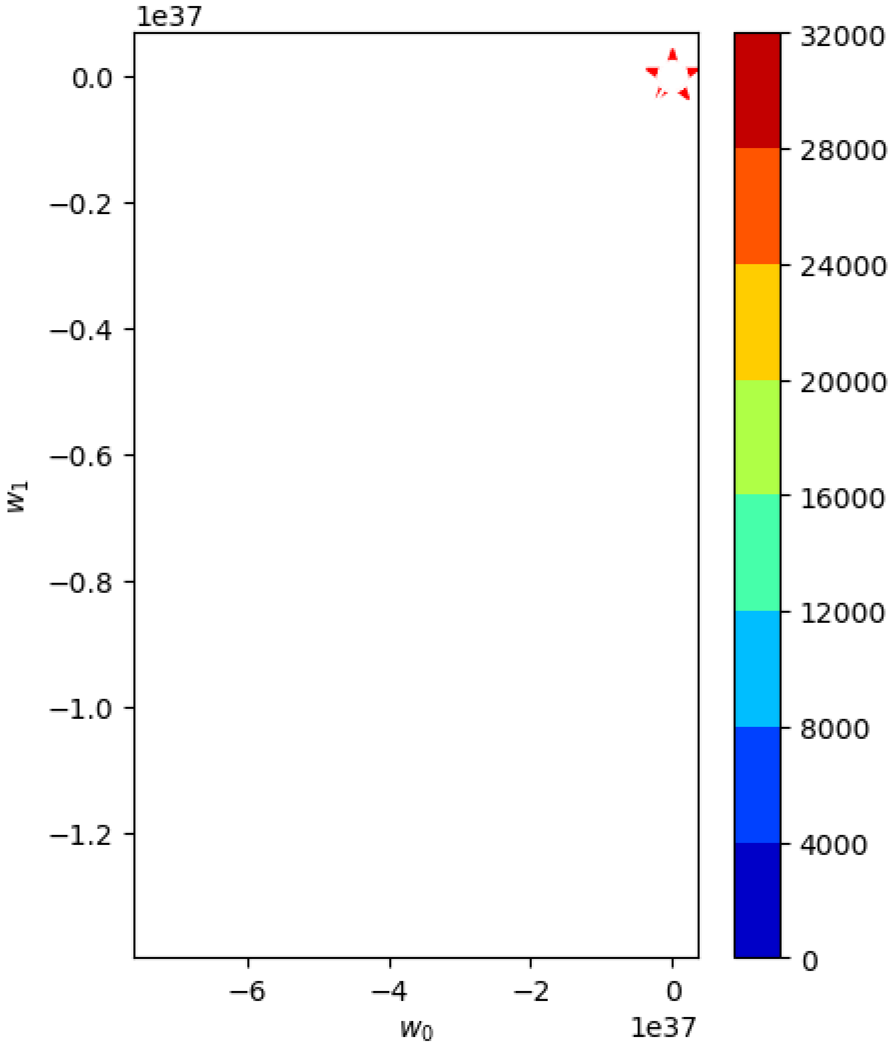


1e28
1.4
1.2
1.0
0.8
0.6
0.4
0.2
0.0
w_1
0
2
4
6
w_0
1e28
32000
28000
24000
20000
16000
12000
8000
4000
0

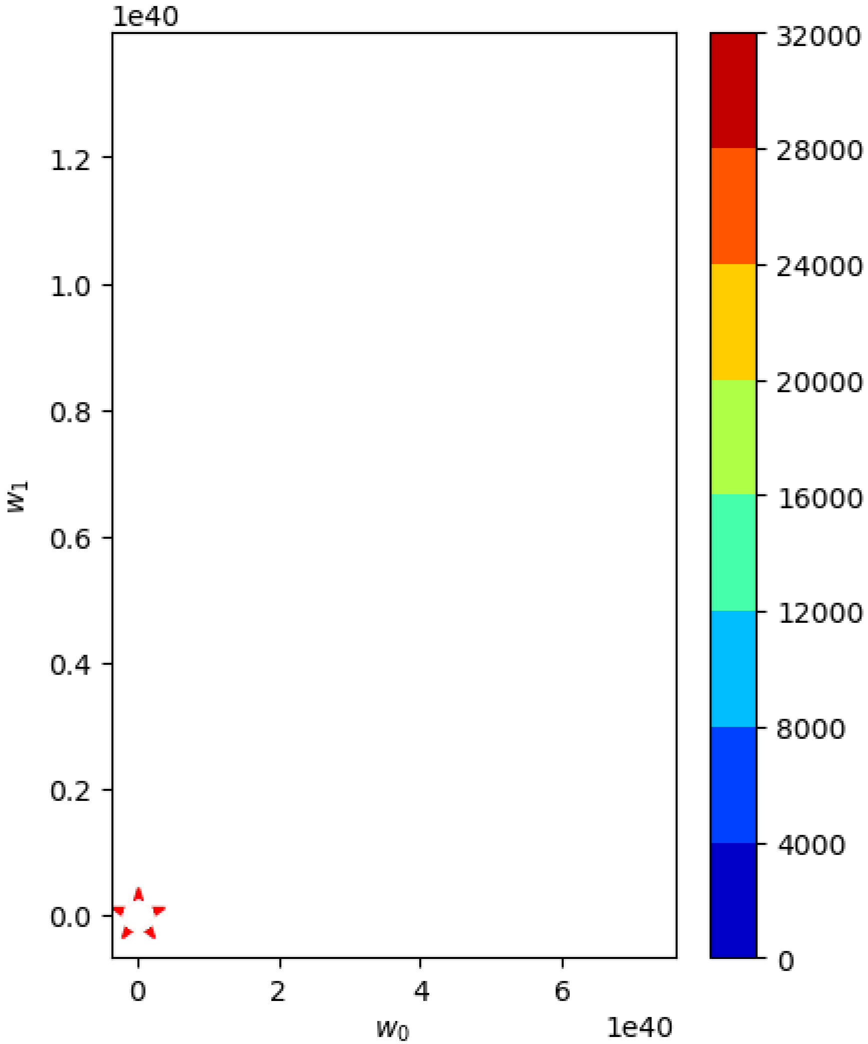

1e31
0.0
−0.2
−0.4
−0.6
−0.8
−1.0
−1.2
−1.4
w_1
−6
−4
−2
0
w_0
1e31
32000
28000
24000
20000
16000
12000
8000
4000
0

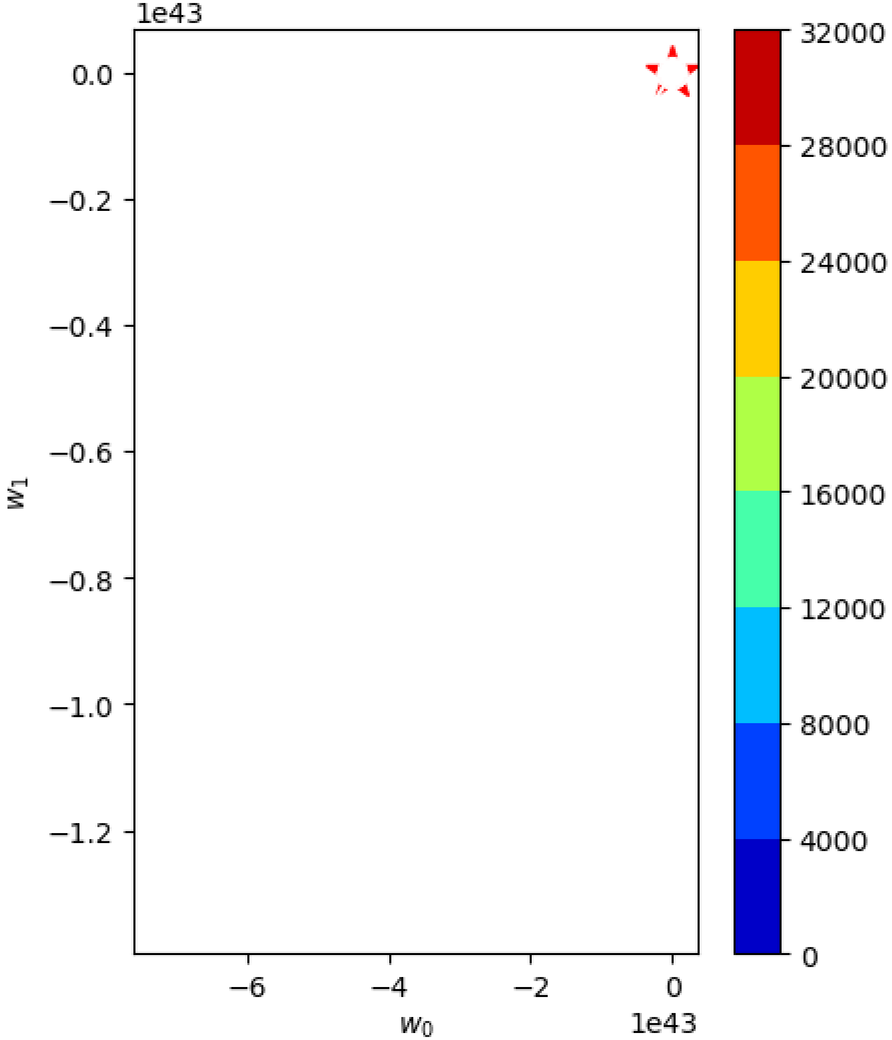


1e34
1.2
1.0
0.8
0.6
0.4
0.2
0.0
w_1
0
2
4
6
w_0
1e34
32000
28000
24000
20000
16000
12000
8000
4000
0

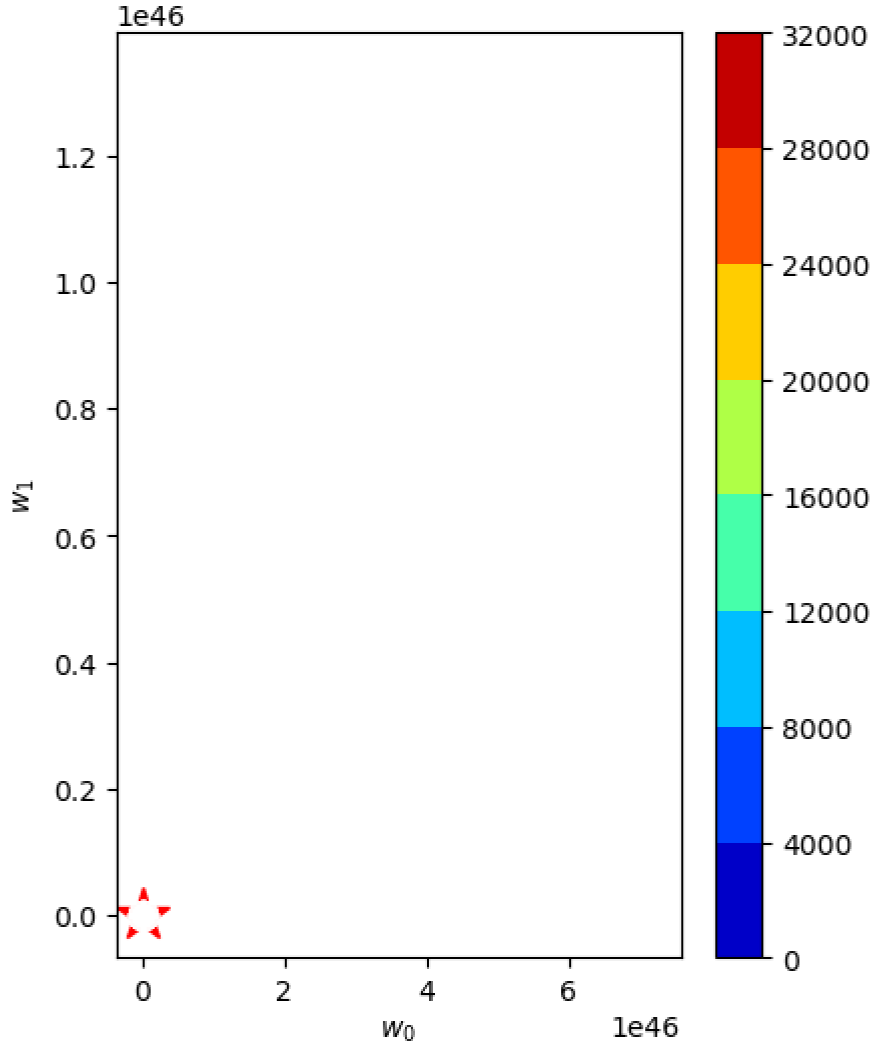


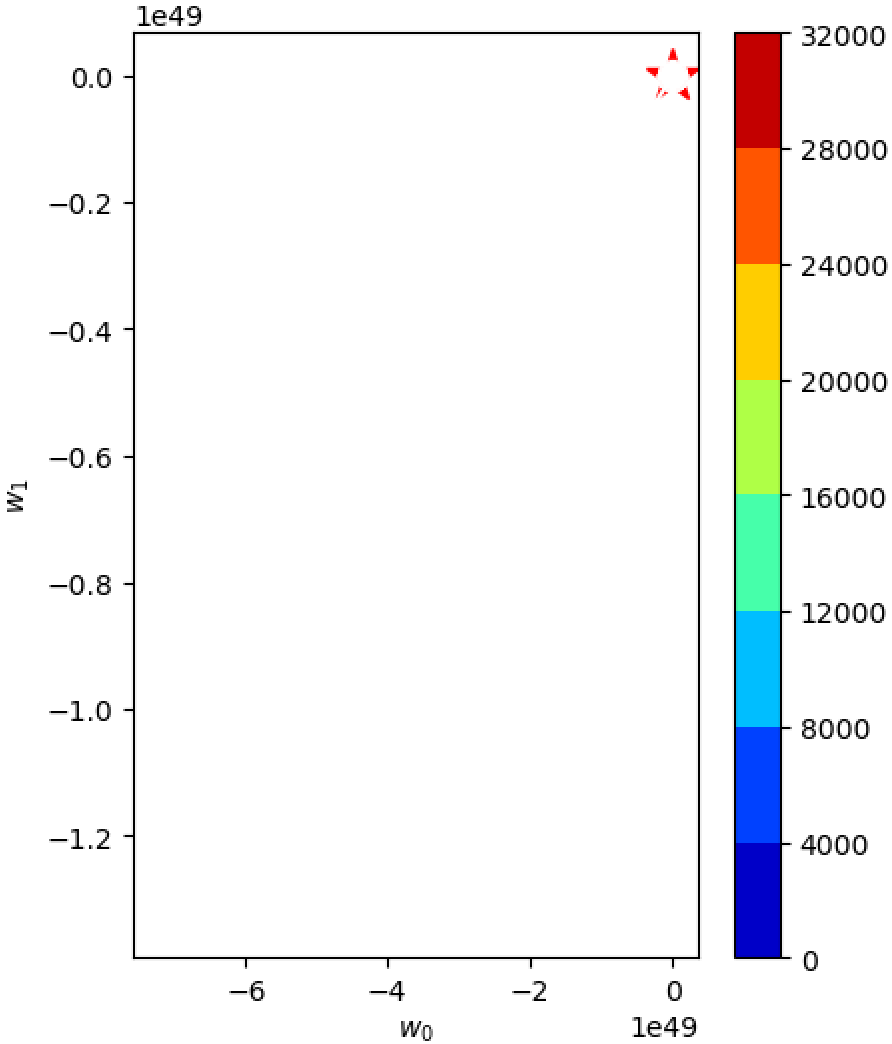
1e37
0.0
−0.2
−0.4
−0.6
−0.8
−1.0
−1.2
w_1
−6
−4
−2
0
w_0
1e37
32000
28000
24000
20000
16000
12000
8000
4000
0

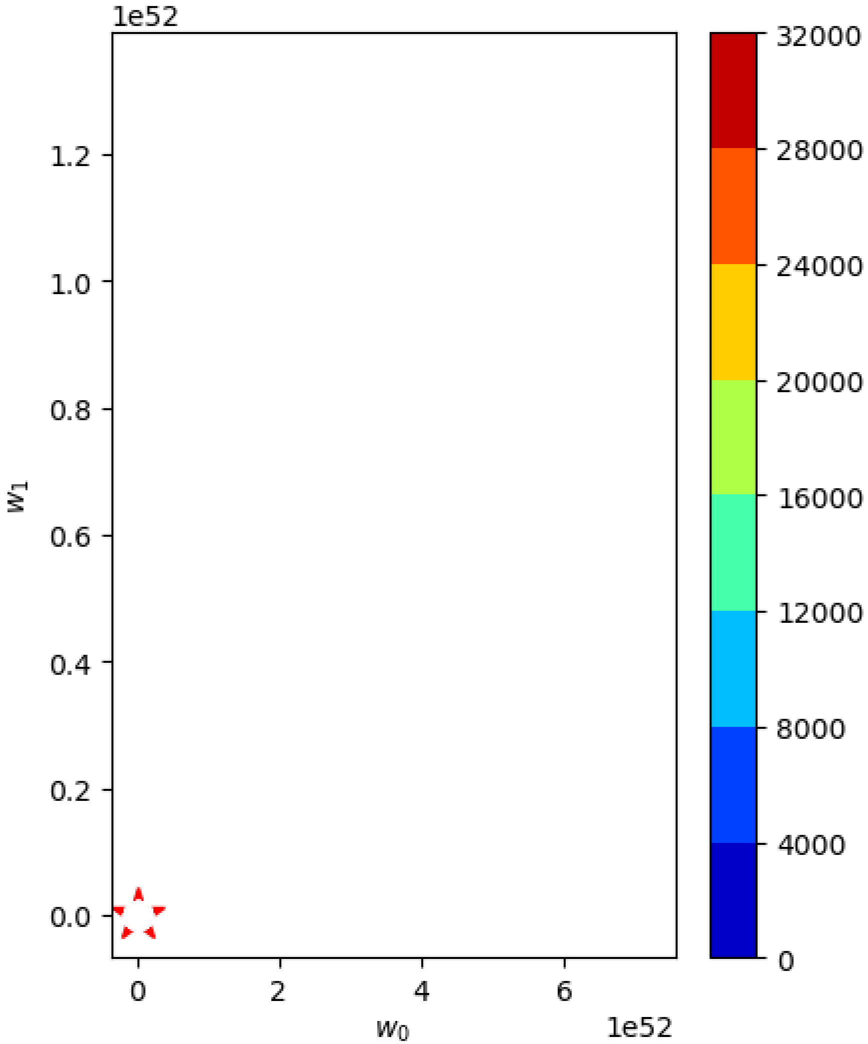

1e40
1.2
1.0
0.8
0.6
0.4
0.2
0.0
w_1
0
2
4
6
w_0
1e40
32000
28000
24000
20000
16000
12000
8000
4000
0



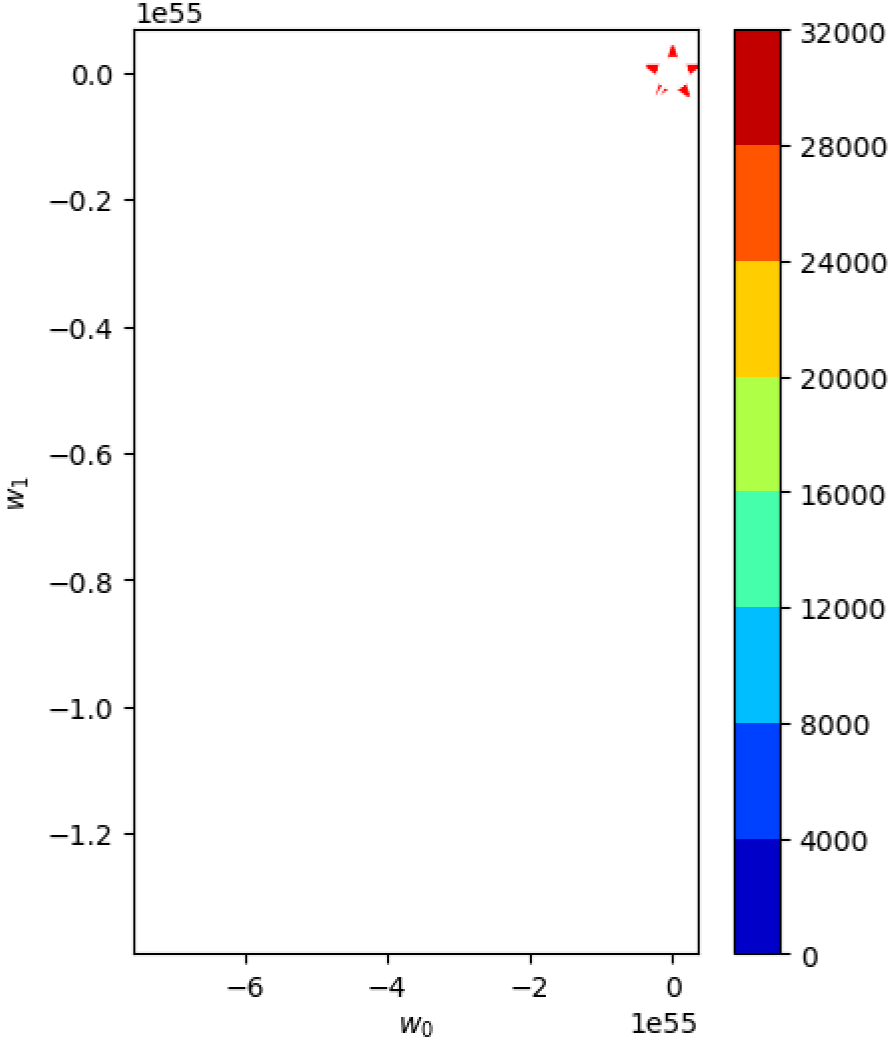
1e43
0.0
−0.2
−0.4
−0.6
−0.8
−1.0
−1.2
w_1
−6
−4
−2
0
w_0
1e43
32000
28000
24000
20000
16000
12000
8000
4000
0


1e46
1.2
1.0
0.8
0.6
0.4
0.2
0.0
w_1
0
2
4
6
w_0
1e46
32000
28000
24000
20000
16000
12000
8000
4000
0

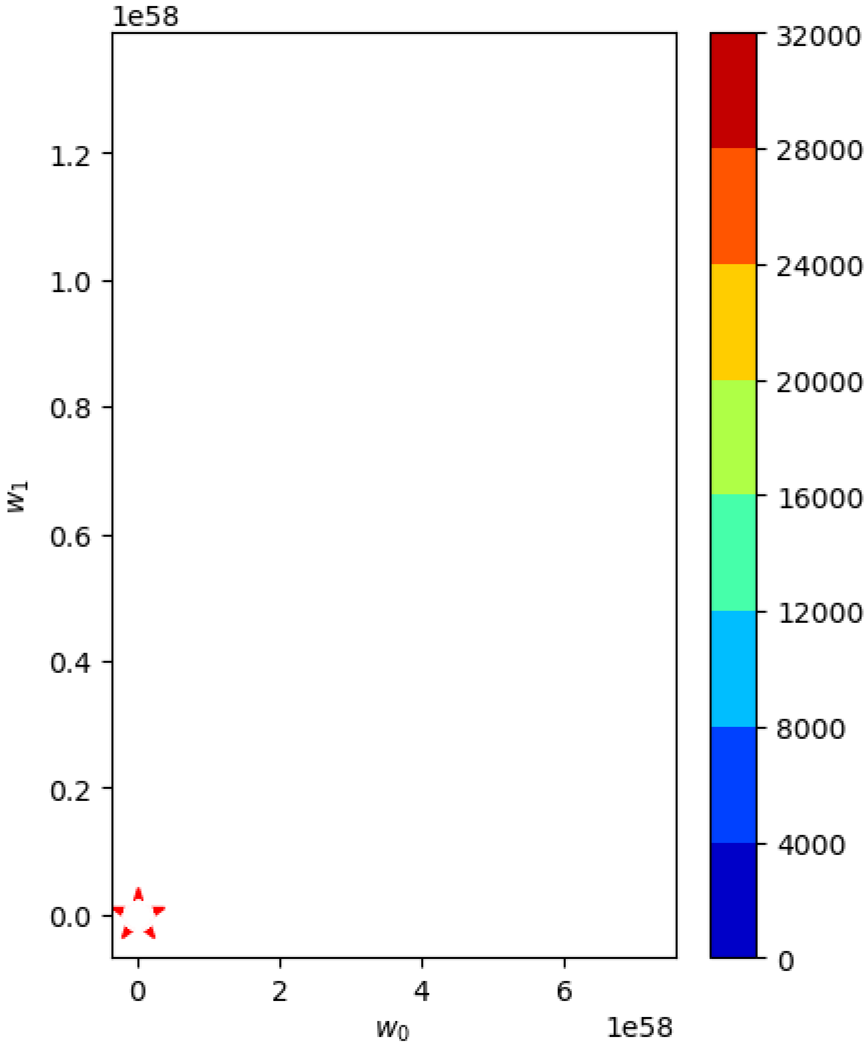

1e49
0.0
−0.2
−0.4
−0.6
−0.8
−1.0
−1.2
w_1
−6
−4
−2
0
1e49
w_0
32000
28000
24000
20000
16000
12000
8000
4000
0

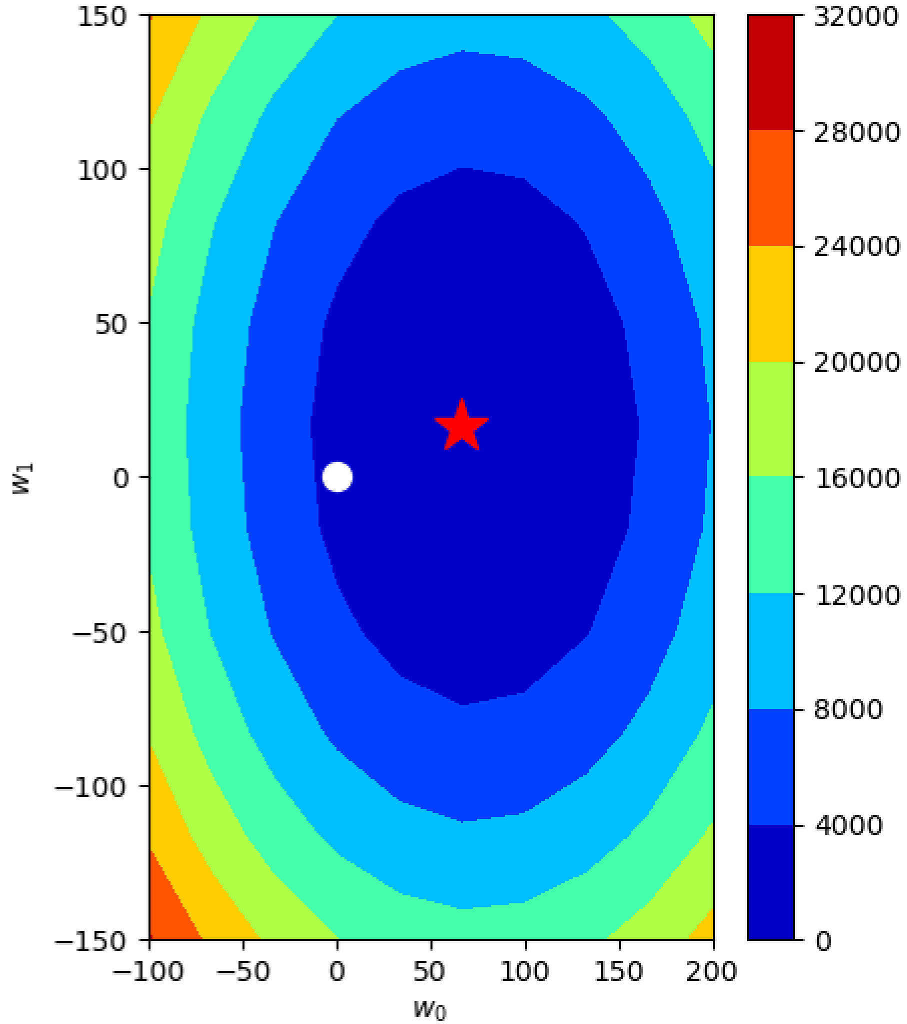


1e52
1.2
1.0
0.8
0.6
0.4
0.2
0.0
w_1
0
2
4
6
w_0
1e52
32000
28000
24000
20000
16000
12000
8000
4000
0

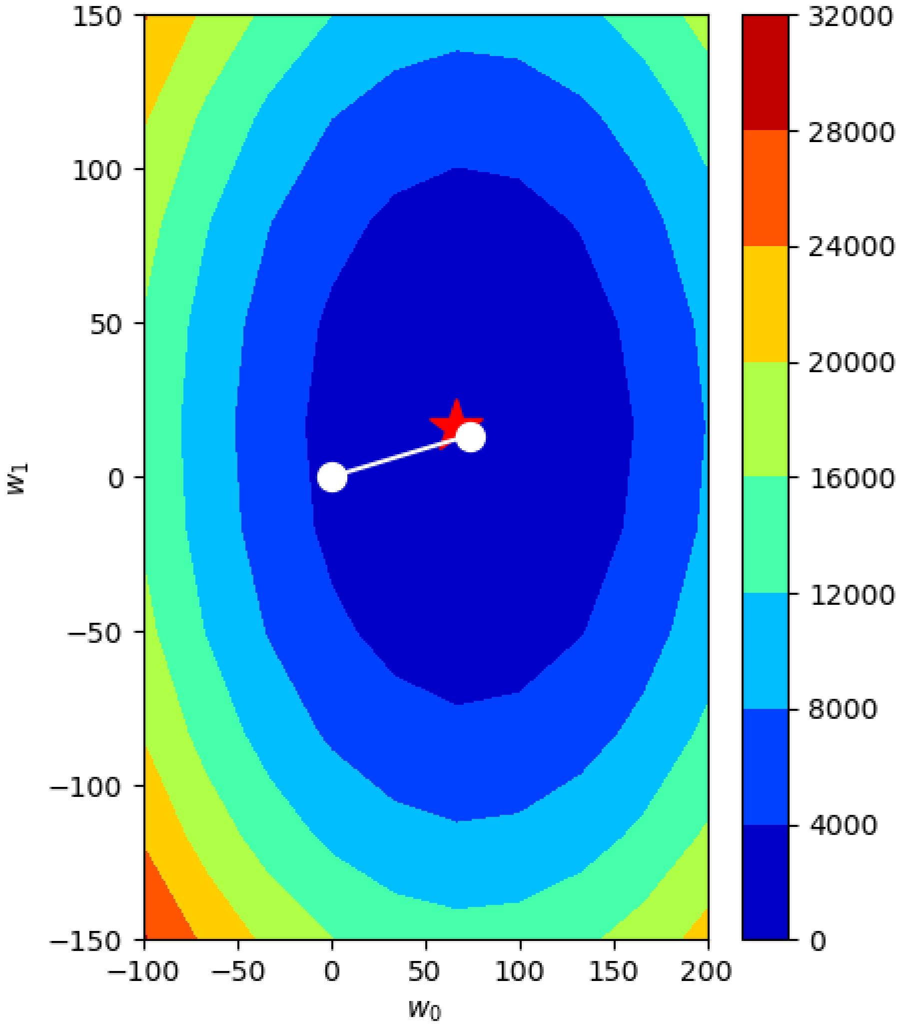


1e55
0.0
−0.2
−0.4
−0.6
−0.8
−1.0
−1.2
w_1
−6
−4
−2
0
1e55
w_0
32000
28000
24000
20000
16000
12000
8000
4000
0

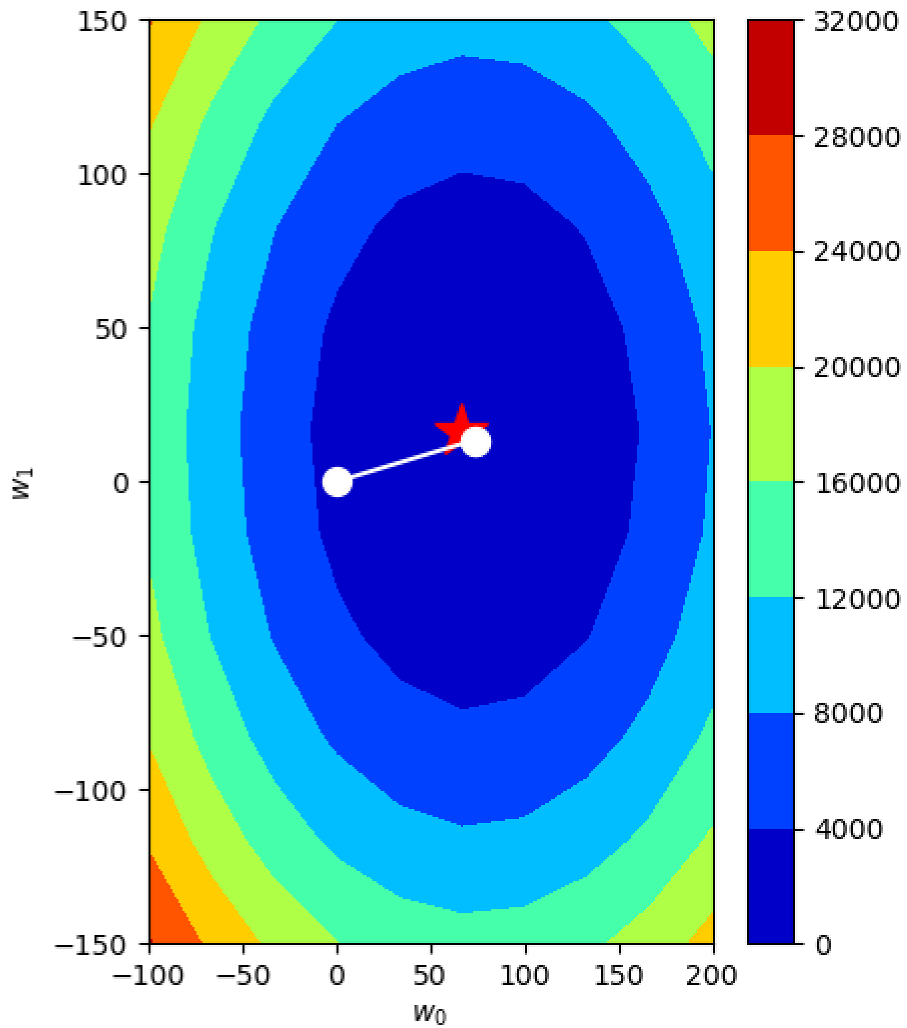

1e58
1.2
1.0
0.8
0.6
0.4
0.2
0.0
w_1
0
2
4
6
w_0
1e58
32000
28000
24000
20000
16000
12000
8000
4000
0



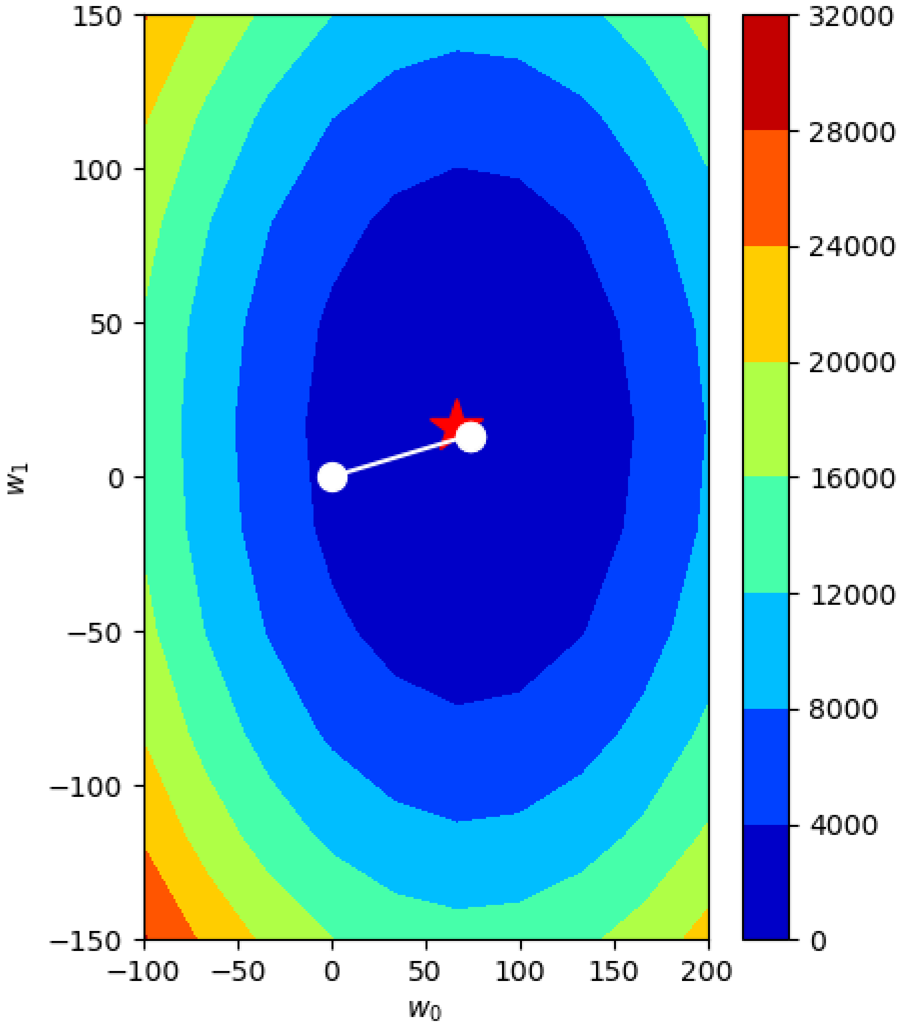
150
100
50
0
−50
−100
−150
w_1
−100
−50
0
50
100
150
200
w_0
32000
28000
24000
20000
16000
12000
8000
4000
0



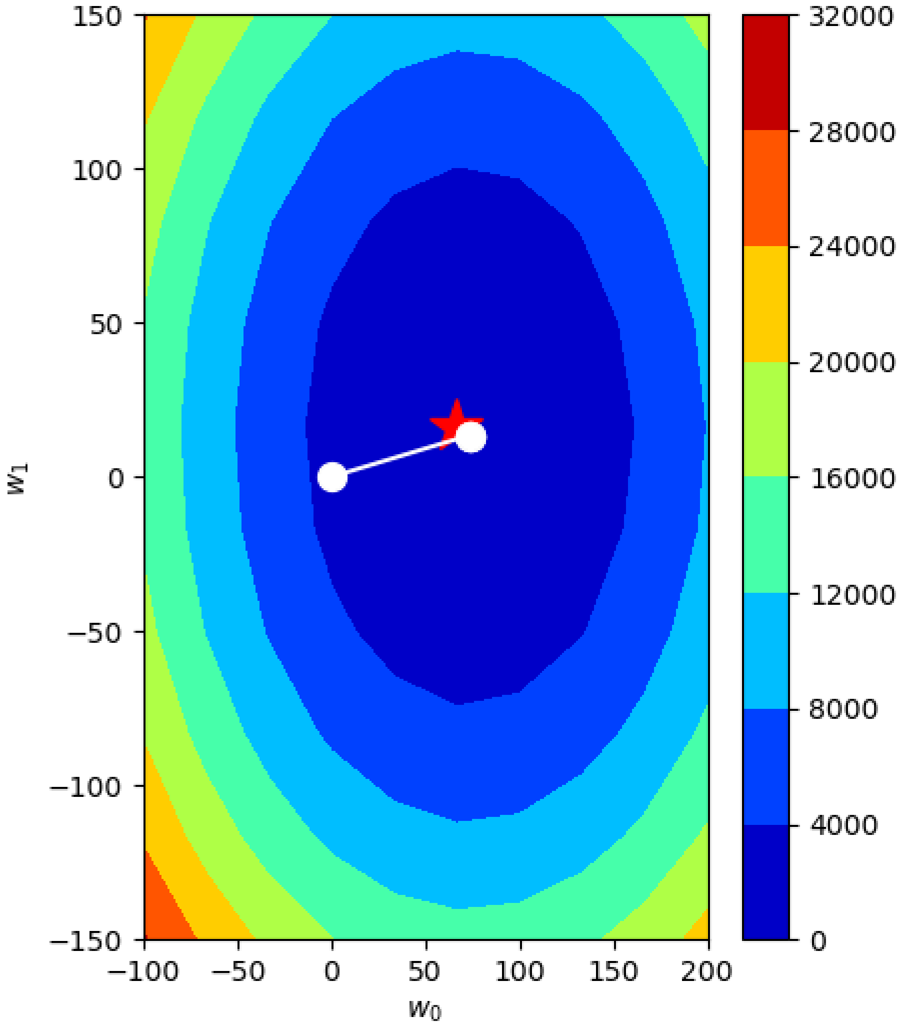
150
100
50
0
−50
−100
−150
w_1
−100
−50
0
50
100
150
200
w_0
32000
28000
24000
20000
16000
12000
8000
4000
0



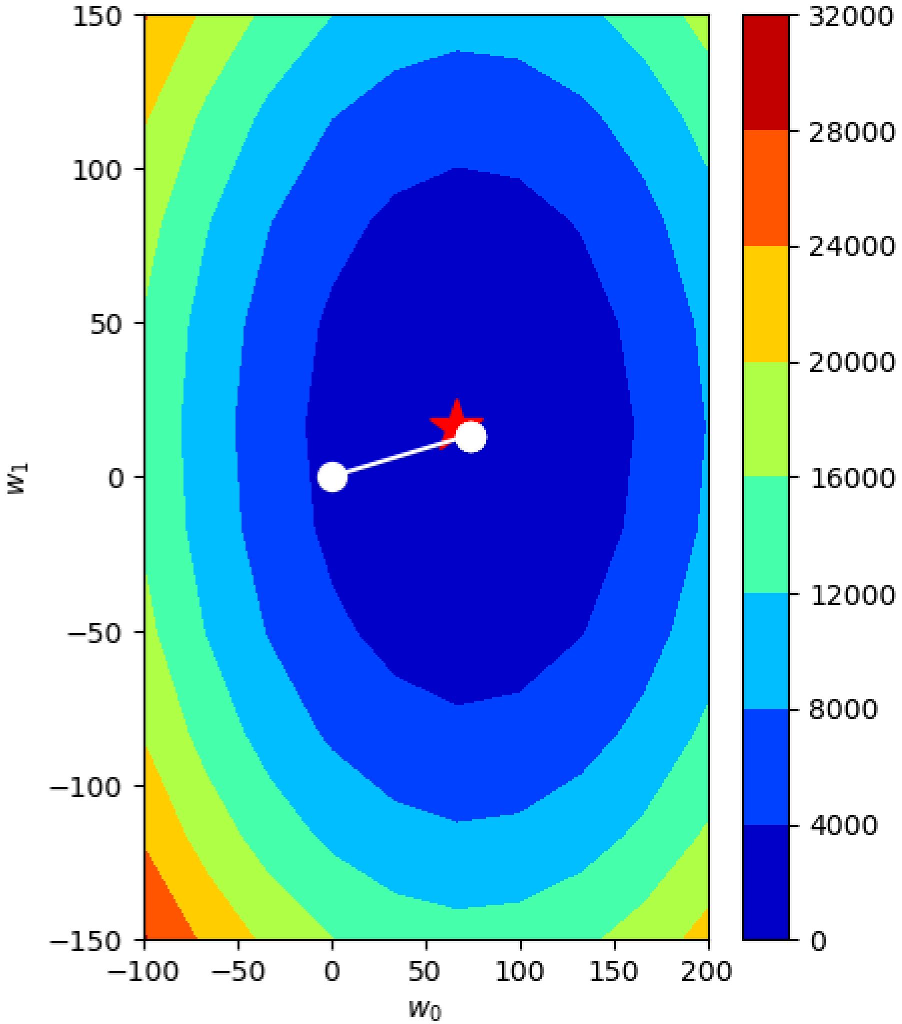
150
100
50
0
−50
−100
−150
w_1
−100
−50
0
50
100
150
200
w_0
32000
28000
24000
20000
16000
12000
8000
4000
0



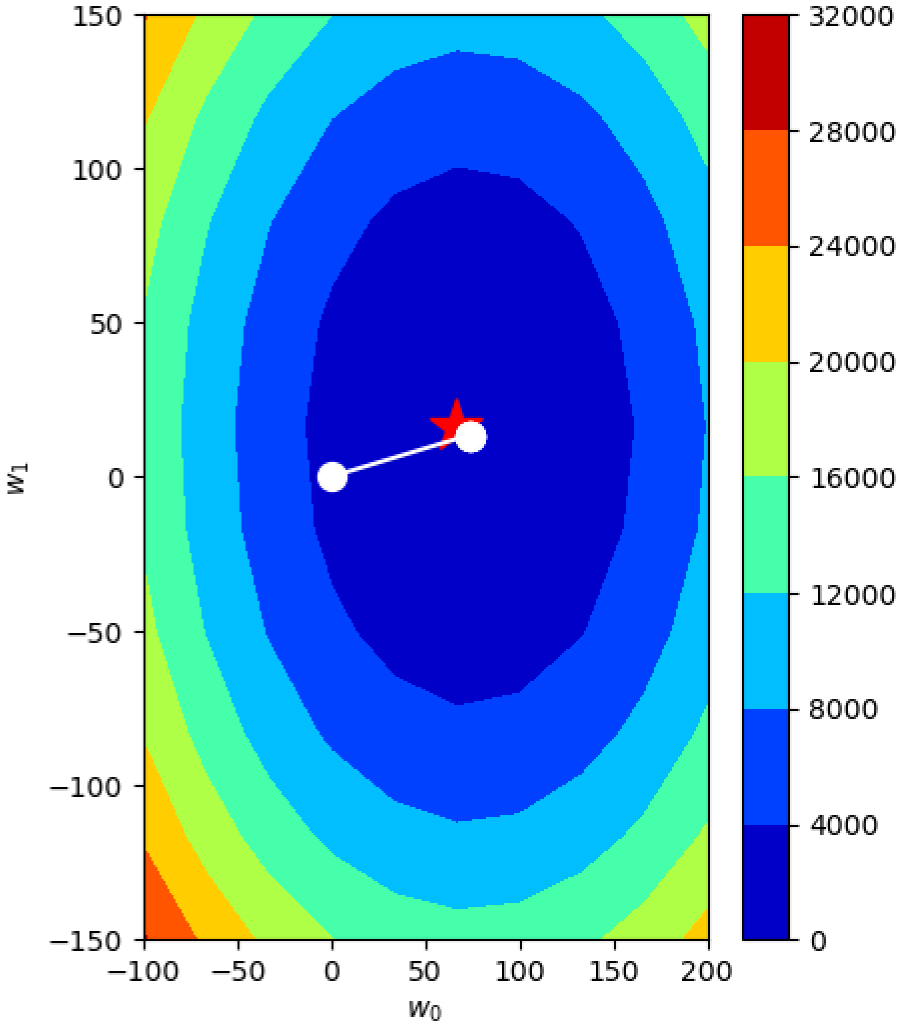
150
100
50
0
−50
−100
−150
w_1
−100
−50
0
50
100
150
200
w_0
32000
28000
24000
20000
16000
12000
8000
4000
0

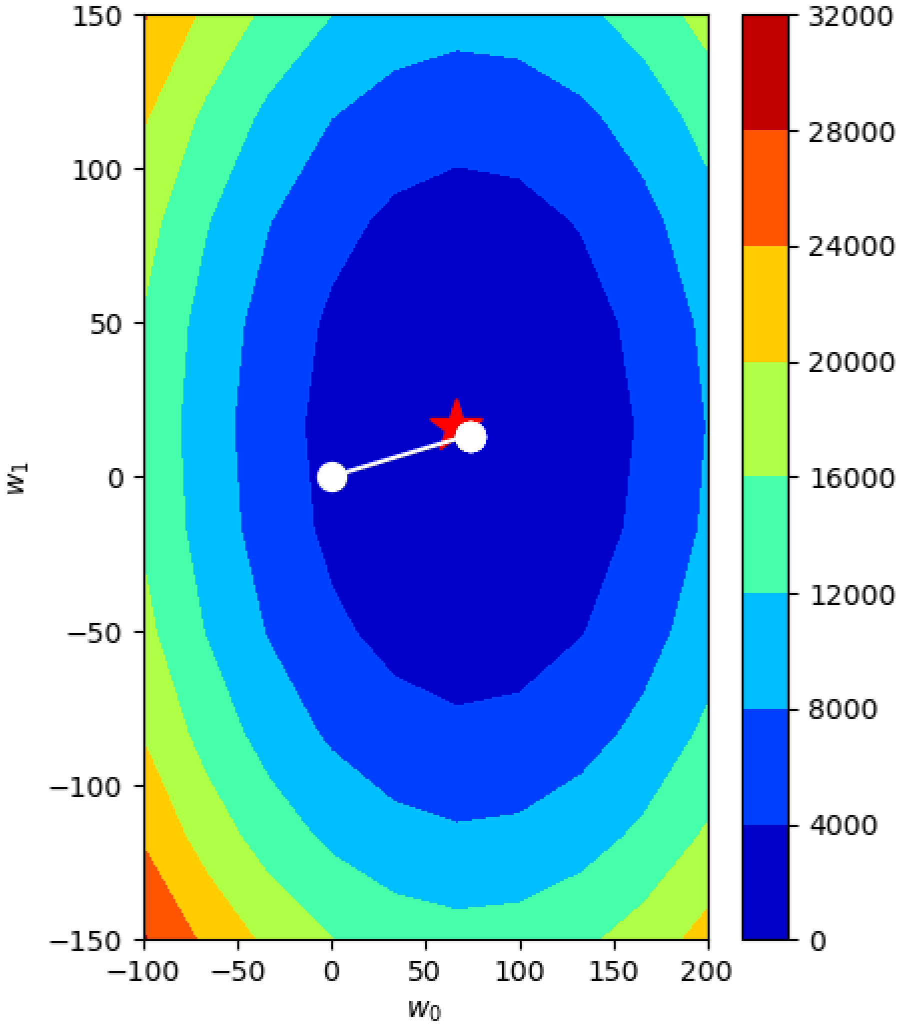


150
100
50
0
−50
−100
−150
w_1
−100
−50
0
50
100
150
200
w_0
32000
28000
24000
20000
16000
12000
8000
4000
0

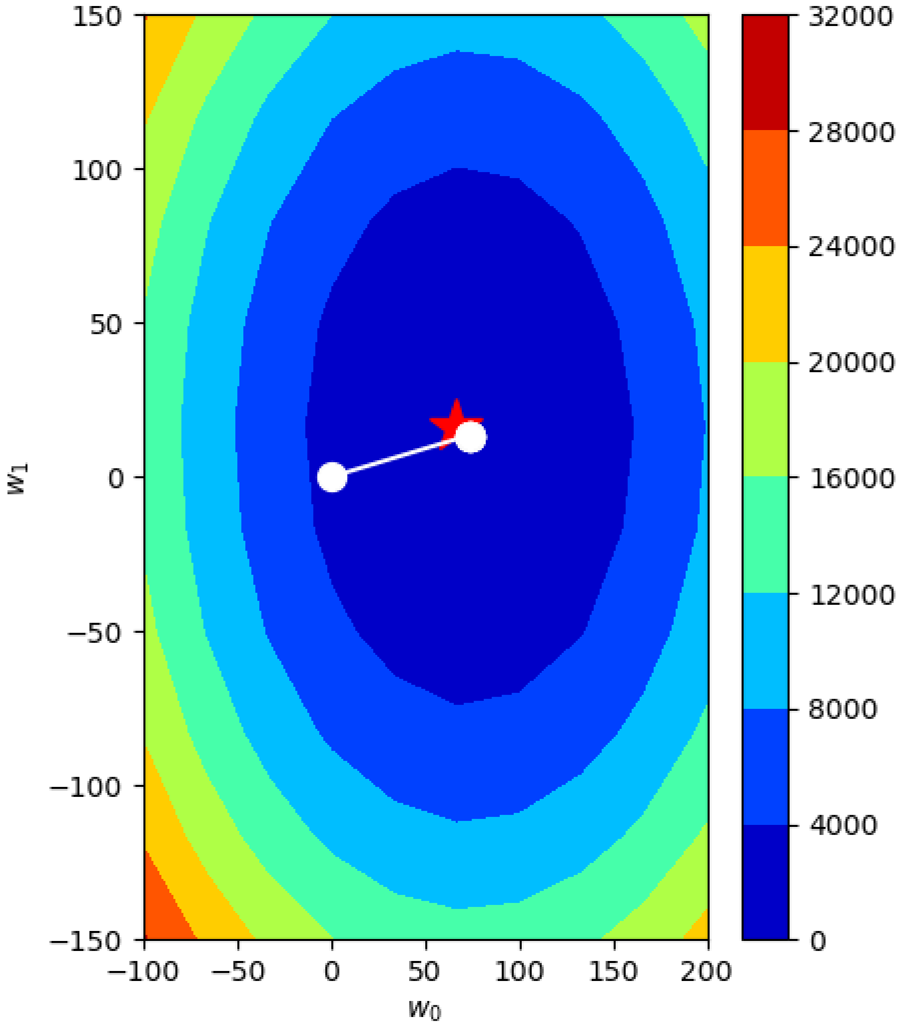


150
100
50
0
−50
−100
−150
−100
−50
0
50
100
150
200
w_0
w_1
32000
28000
24000
20000
16000
12000
8000
4000
0

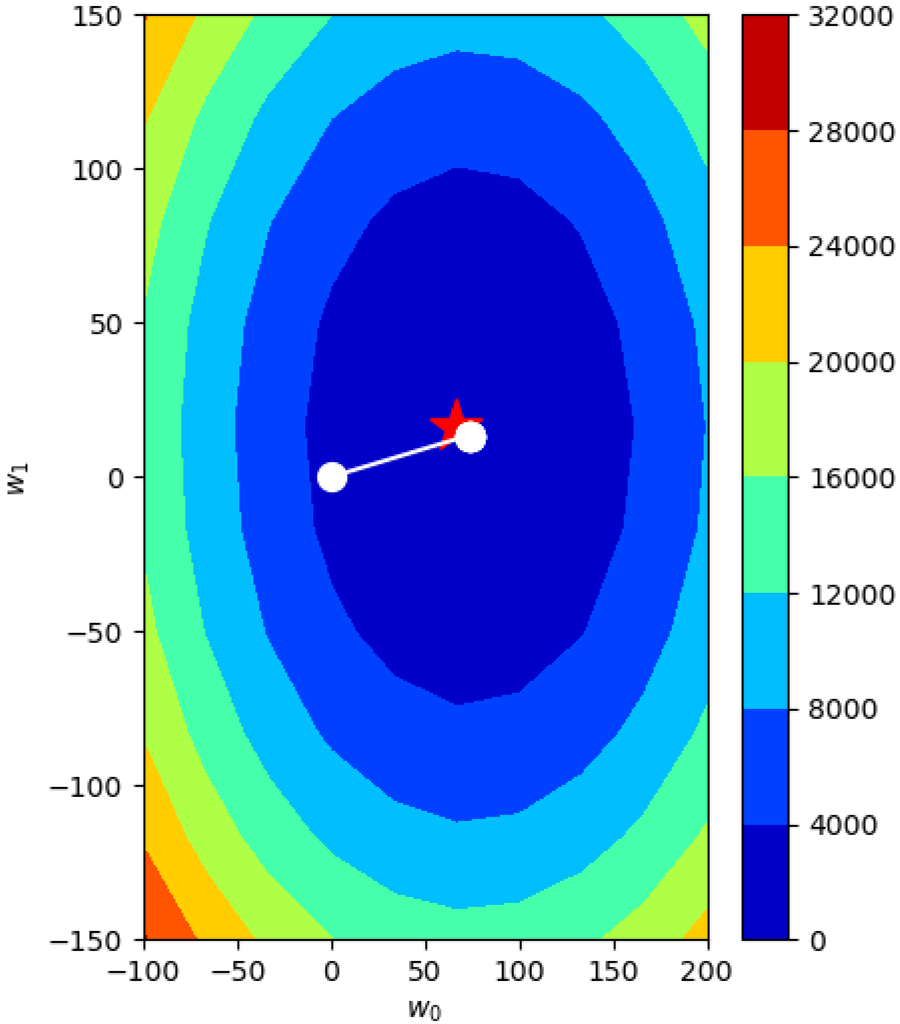

150
100
50
0
−50
−100
−150
w_1
−100
−50
0
50
100
150
200
w_0
32000
28000
24000
20000
16000
12000
8000
4000
0

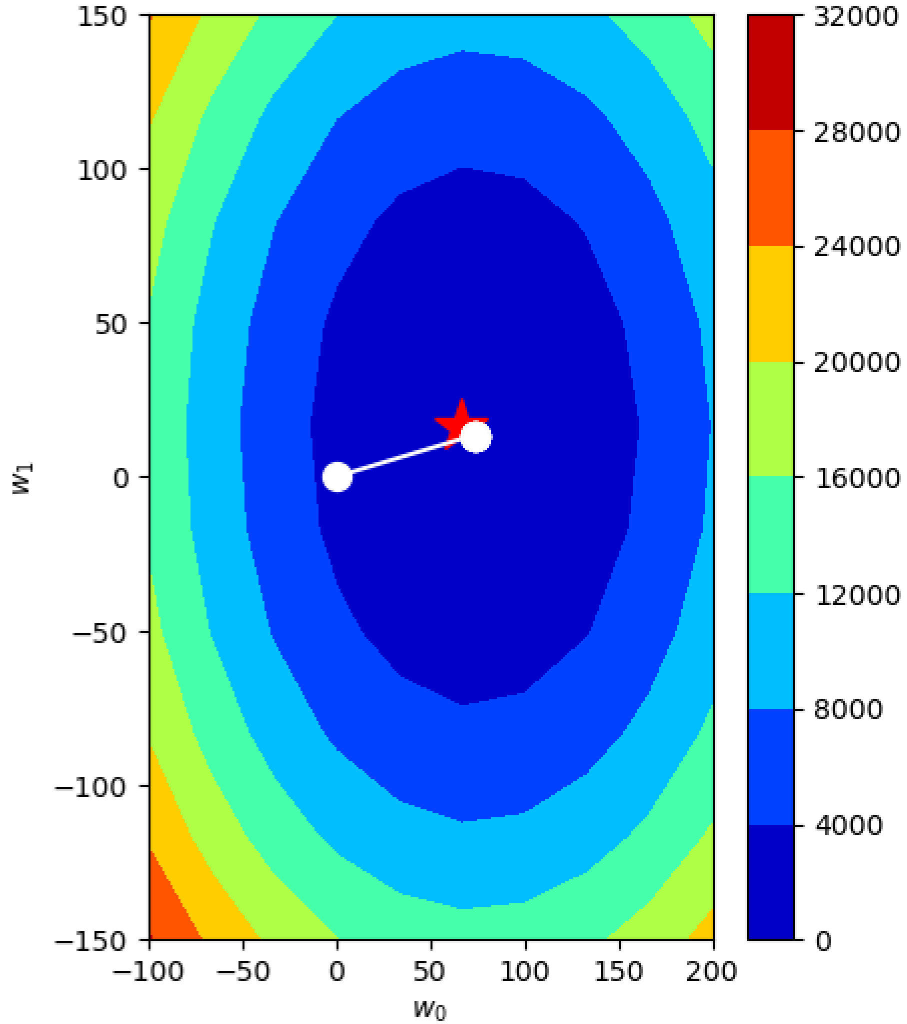


150
100
50
0
−50
−100
−150
w_1
−100
−50
0
50
100
150
200
w_0
32000
28000
24000
20000
16000
12000
8000
4000
0

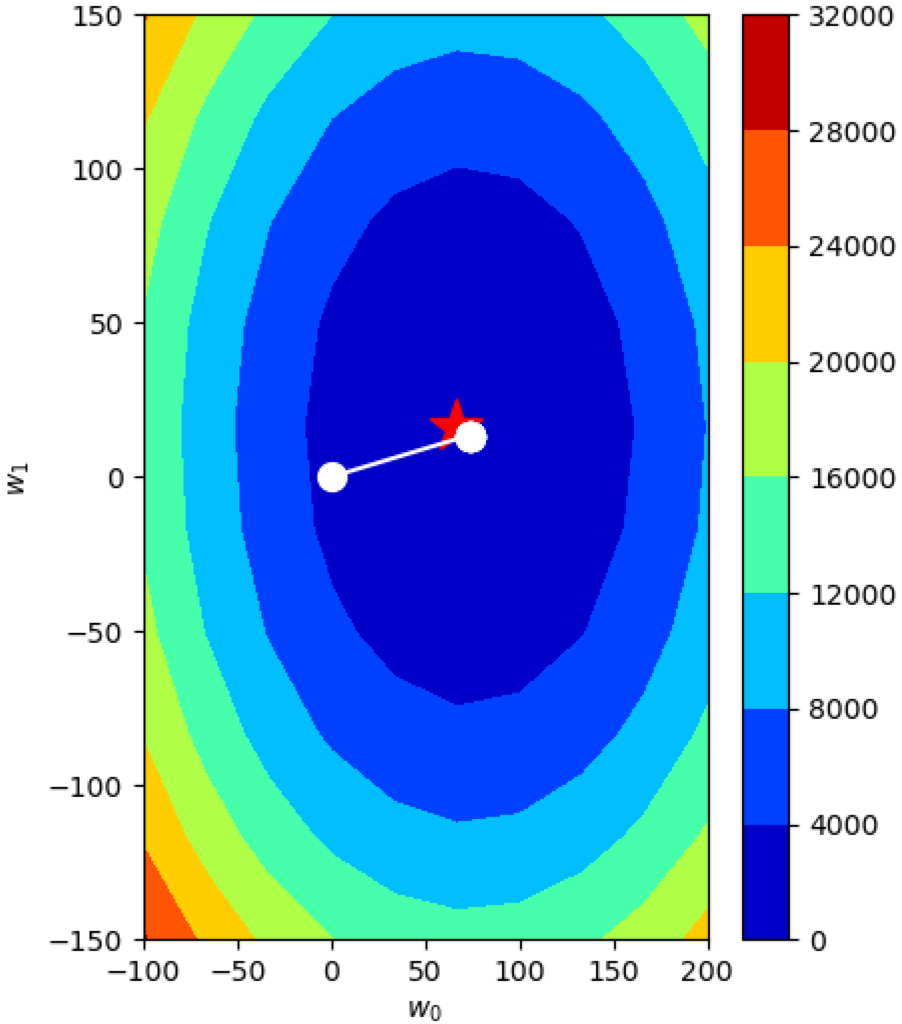


150
100
50
0
−50
−100
−150
w_1
−100
−50
0
50
100
150
200
w_0
32000
28000
24000
20000
16000
12000
8000
4000
0

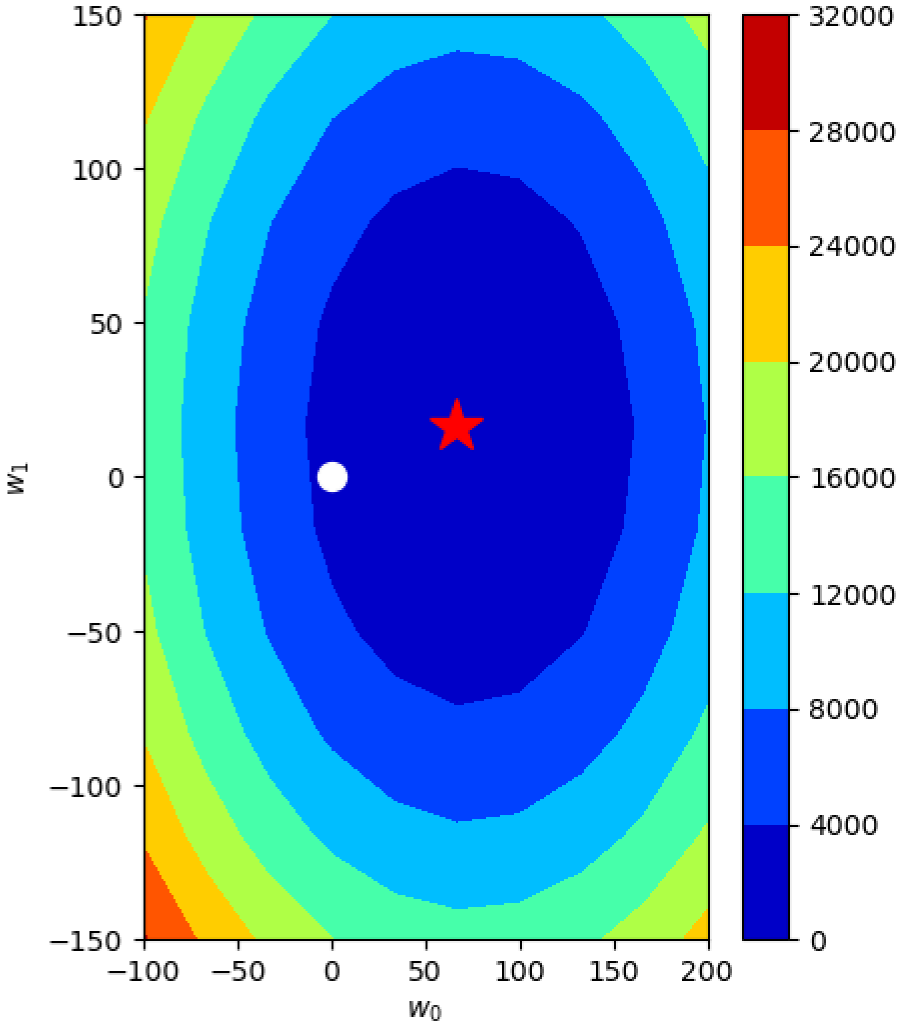

150
100
50
0
−50
−100
−150
w_1
−100
−50
0
50
100
150
200
w_0
32000
28000
24000
20000
16000
12000
8000
4000
0



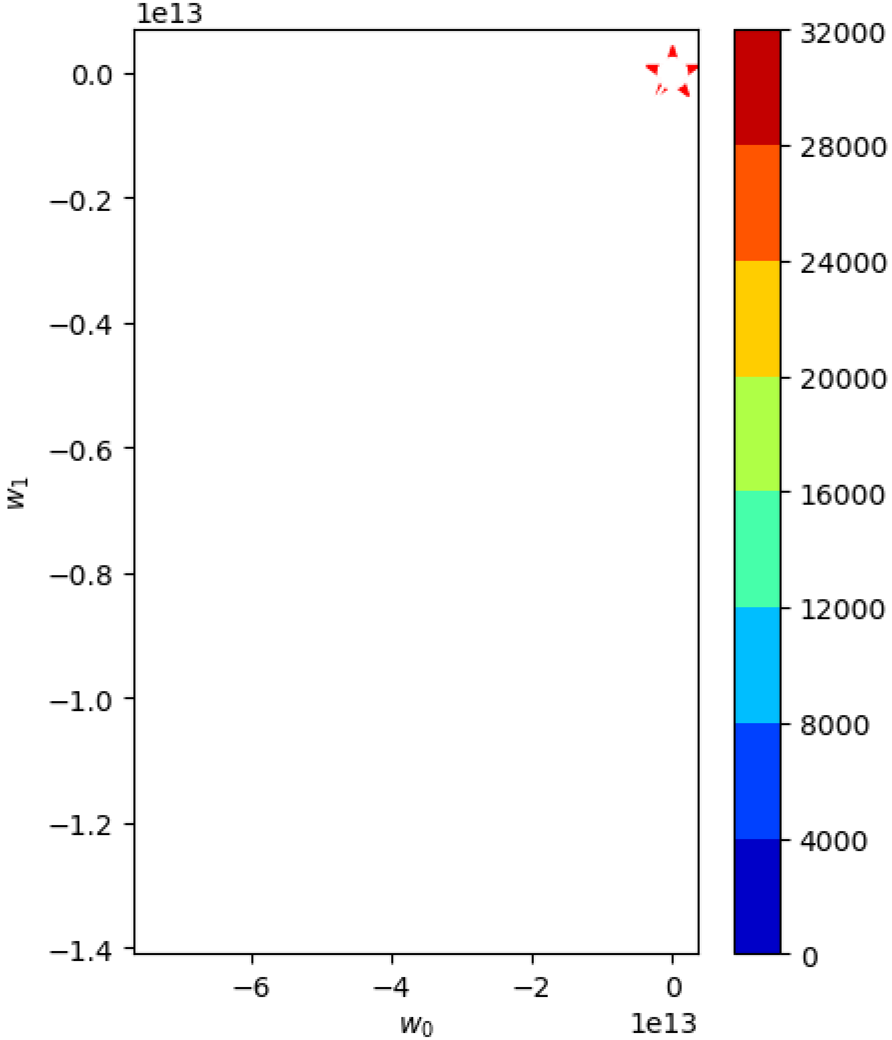
150
100
50
0
−50
−100
−150
w_1
−100
−50
0
50
100
150
200
w_0
32000
28000
24000
20000
16000
12000
8000
4000
0

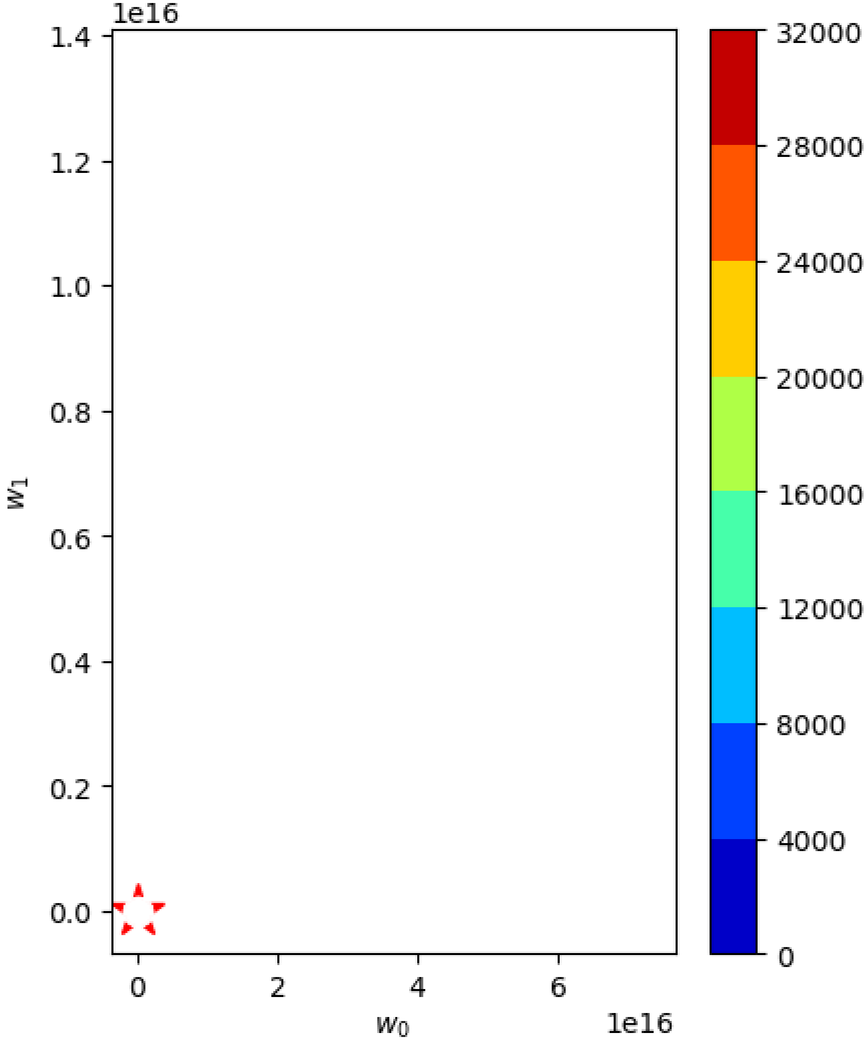


150
100
50
0
−50
−100
−150
w_1
−100
−50
0
50
100
150
200
w_0
32000
28000
24000
20000
16000
12000
8000
4000
0

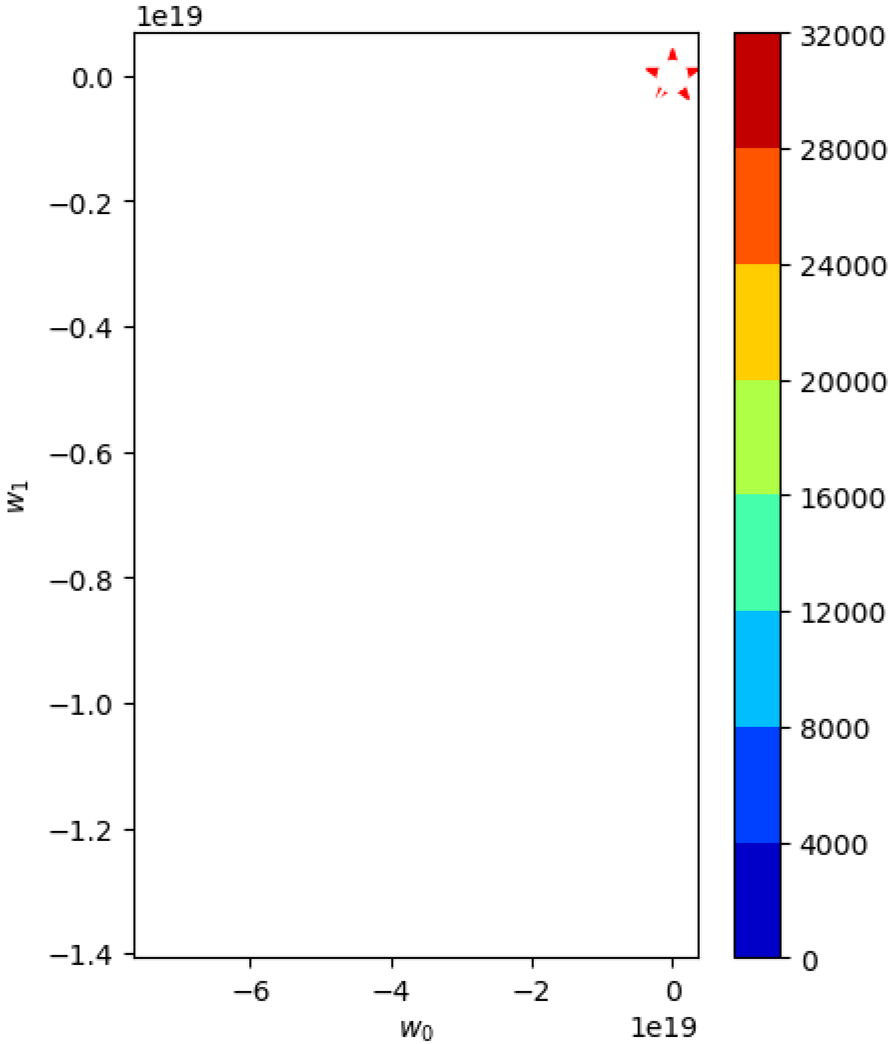


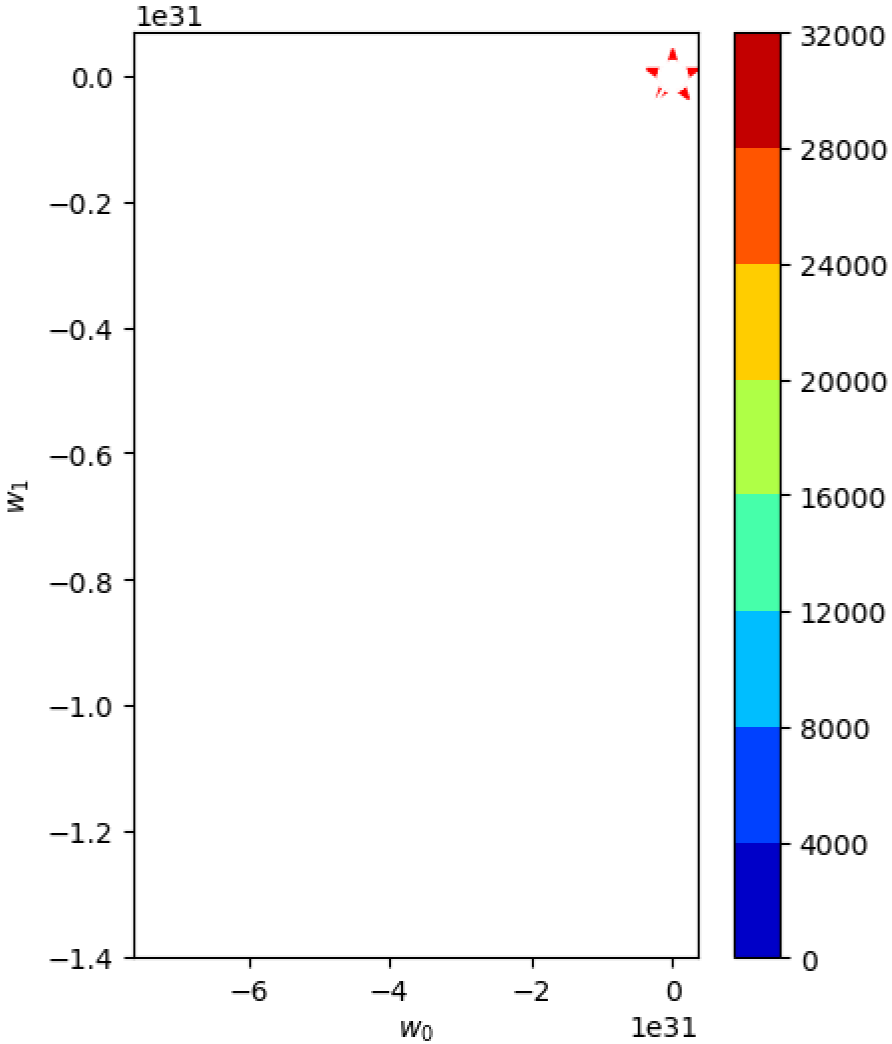
150
100
50
0
−50
−100
−150
w_1
−100
−50
0
50
100
150
200
w_0
32000
28000
24000
20000
16000
12000
8000
4000
0

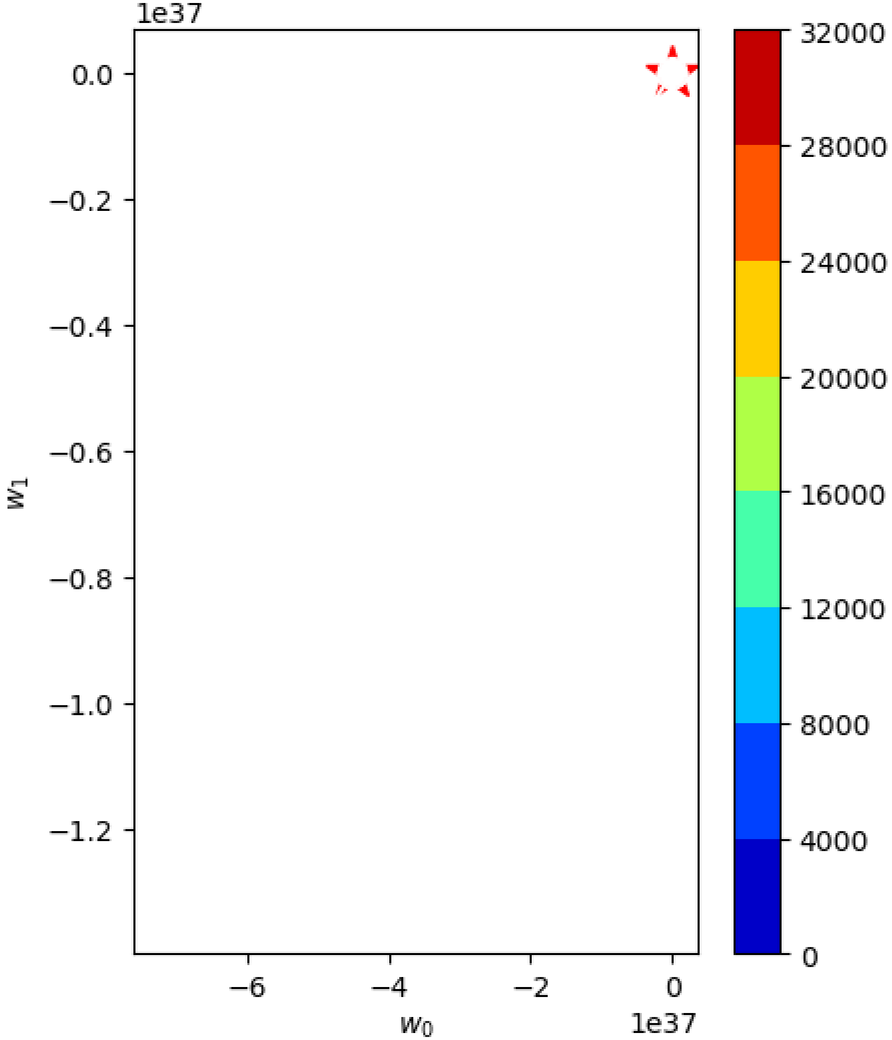

1e13
0.0
−0.2
−0.4
−0.6
−0.8
−1.0
−1.2
−1.4
w_1
−6
−4
−2
0
w_0
1e13
32000
28000
24000
20000
16000
12000
8000
4000
0



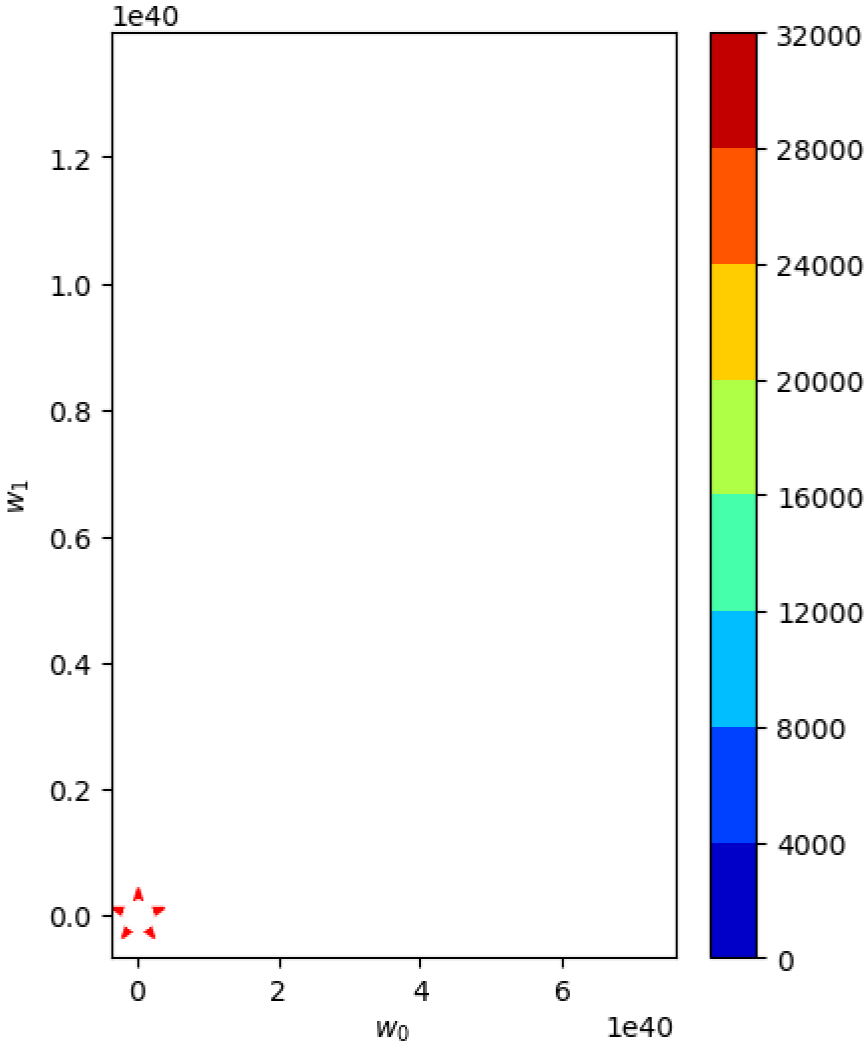
1e16
1.4
1.2
1.0
0.8
0.6
0.4
0.2
0.0
w_1
0
2
4
6
w_0
1e16
32000
28000
24000
20000
16000
12000
8000
4000
0


1e19
0.0
−0.2
−0.4
−0.6
−0.8
−1.0
−1.2
−1.4
w_1
−6
−4
−2
0
1e19
w_0
32000
28000
24000
20000
16000
12000
8000
4000
0

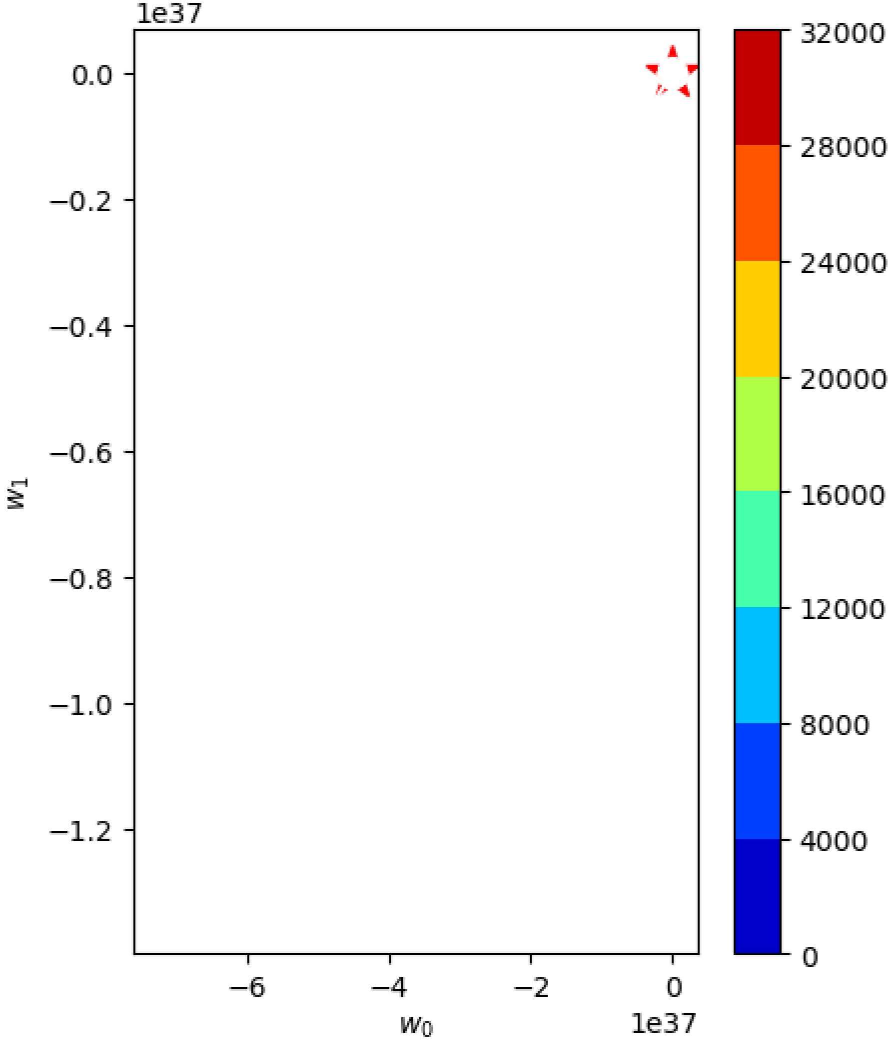

1e31
0.0
−0.2
−0.4
−0.6
−0.8
−1.0
−1.2
−1.4
w_1
−6
−4
−2
0
w_0
1e31
32000
28000
24000
20000
16000
12000
8000
4000
0

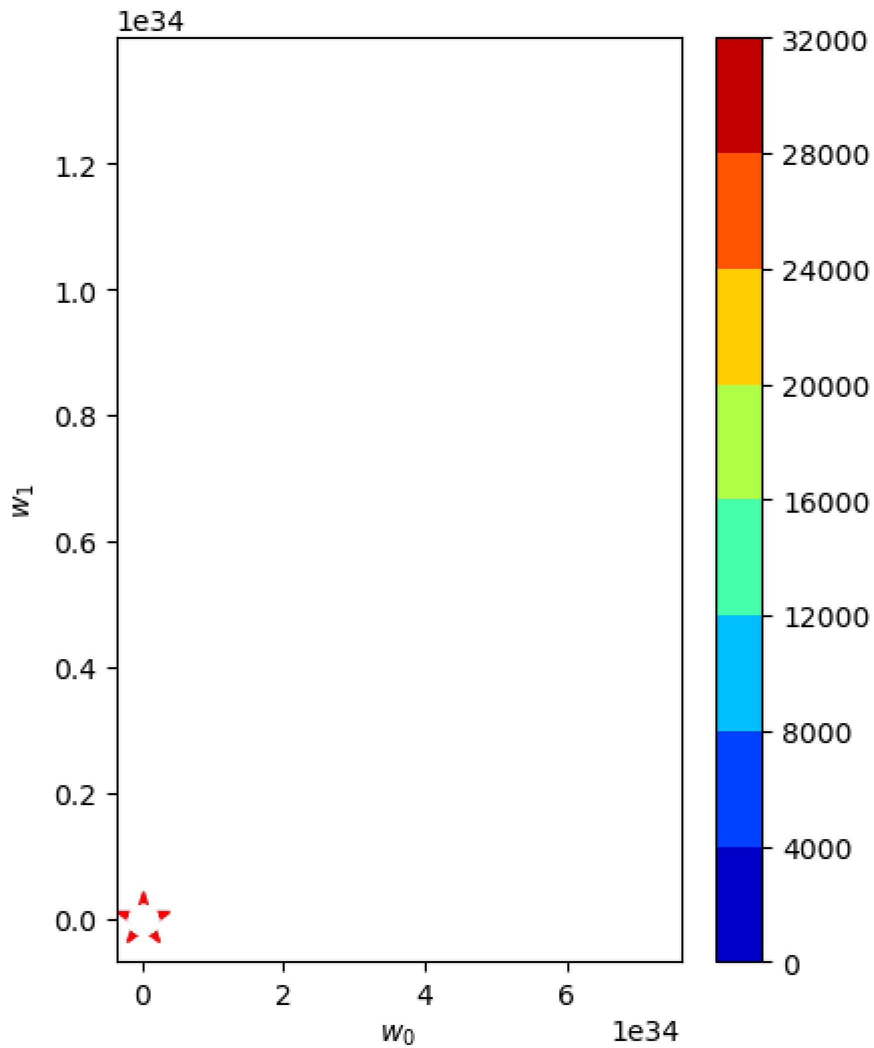


1e37
0.0
−0.2
−0.4
−0.6
−0.8
−1.0
−1.2
w_1
−6
−4
−2
0
1e37
w_0
32000
28000
24000
20000
16000
12000
8000
4000
0

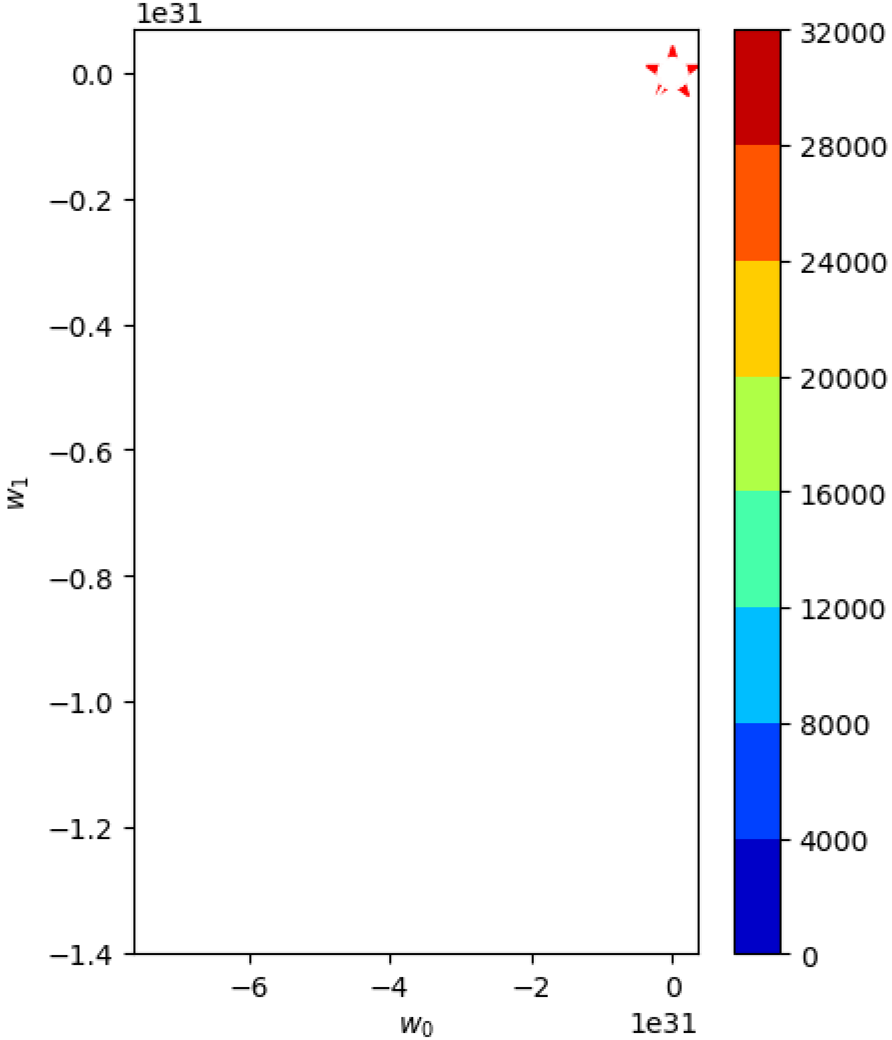


1e40
1.2
1.0
0.8
0.6
0.4
0.2
0.0
w_1
0
2
4
6
w_0
1e40
32000
28000
24000
20000
16000
12000
8000
4000
0

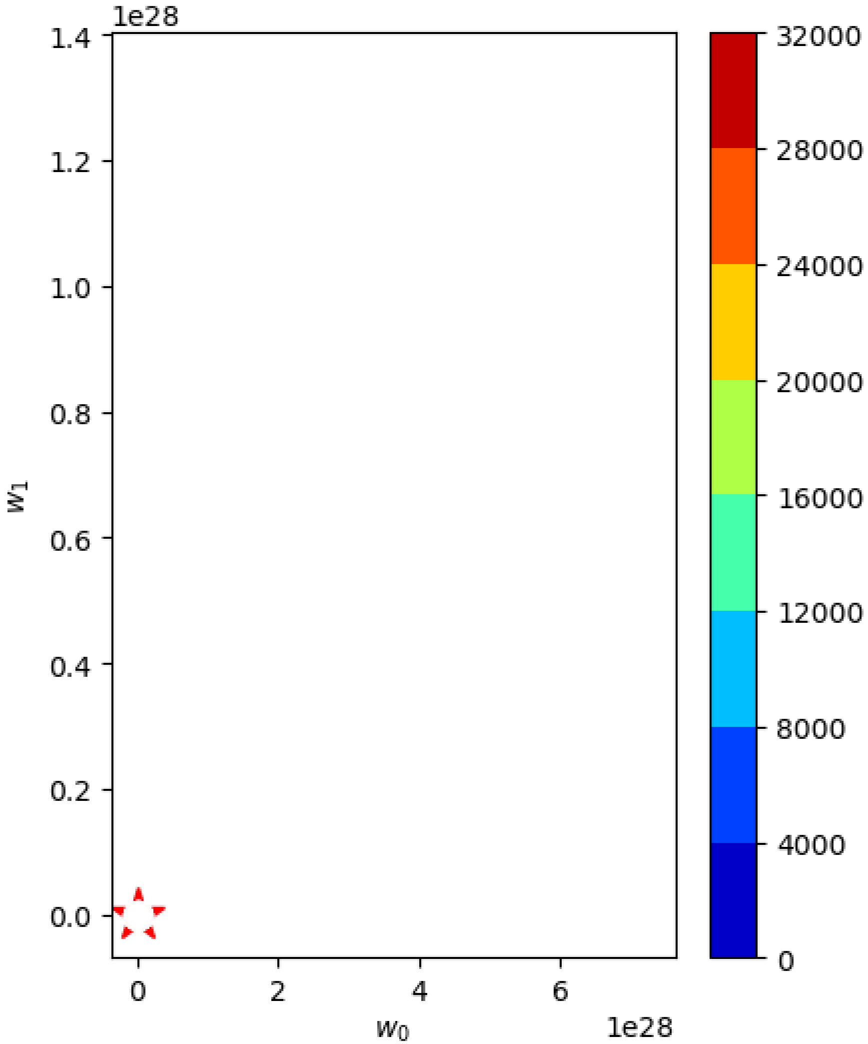

1e37
0.0
−0.2
−0.4
−0.6
−0.8
−1.0
−1.2
w_1
−6
−4
−2
0
1e37
w_0
32000
28000
24000
20000
16000
12000
8000
4000
0



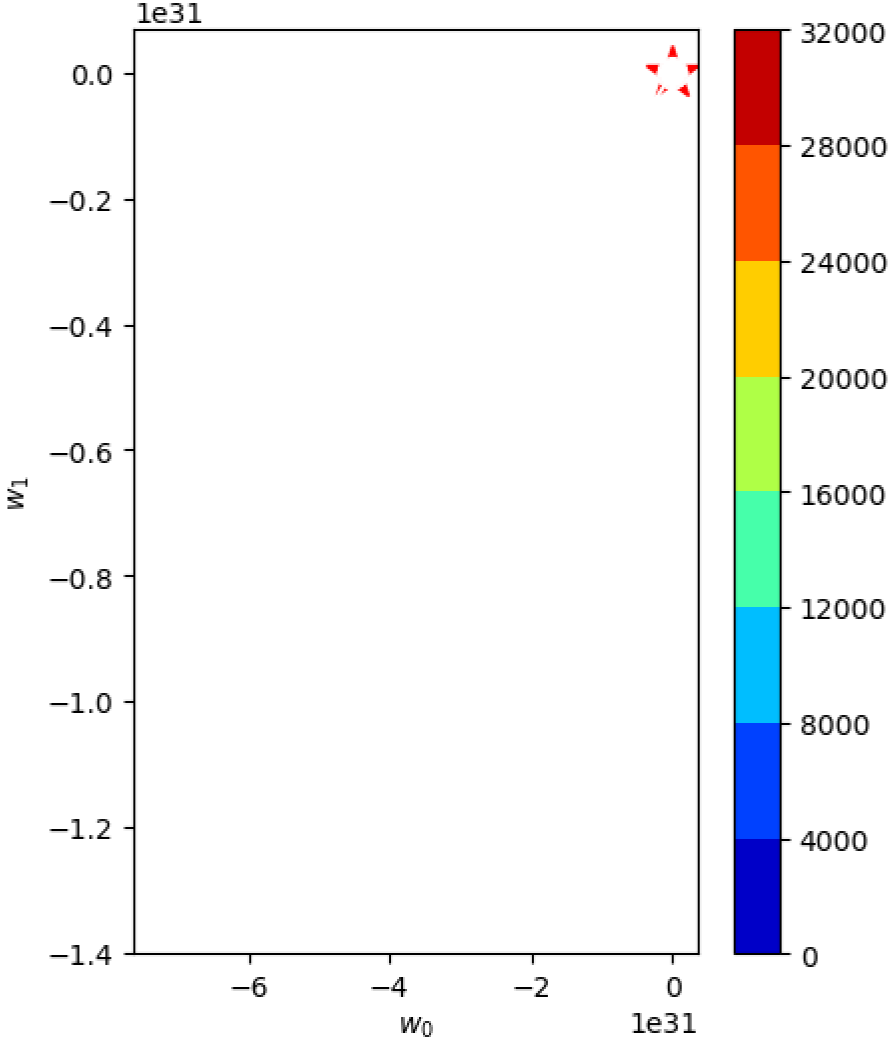
1e34
1.2
1.0
0.8
0.6
0.4
0.2
0.0
w_1
0
2
4
6
w_0
1e34
32000
28000
24000
20000
16000
12000
8000
4000
0



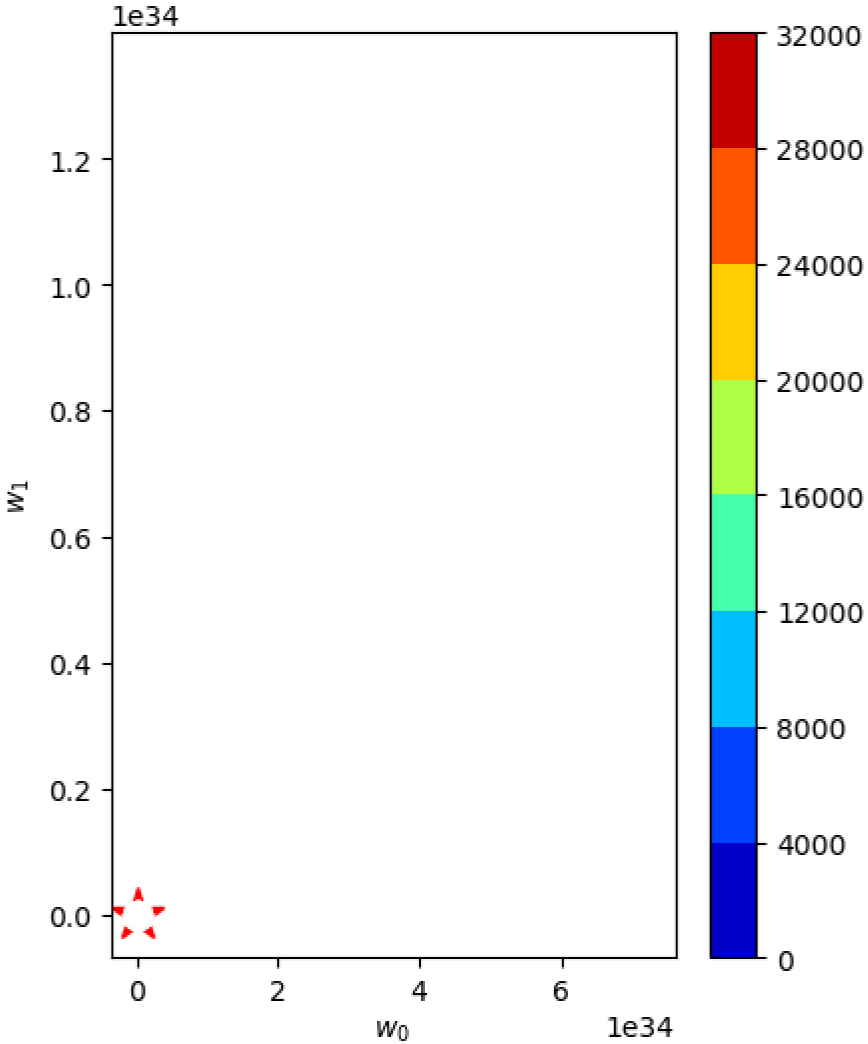
1e31
0.0
−0.2
−0.4
−0.6
−0.8
−1.0
−1.2
−1.4
w_1
−6
−4
−2
0
w_0
1e31
32000
28000
24000
20000
16000
12000
8000
4000
0

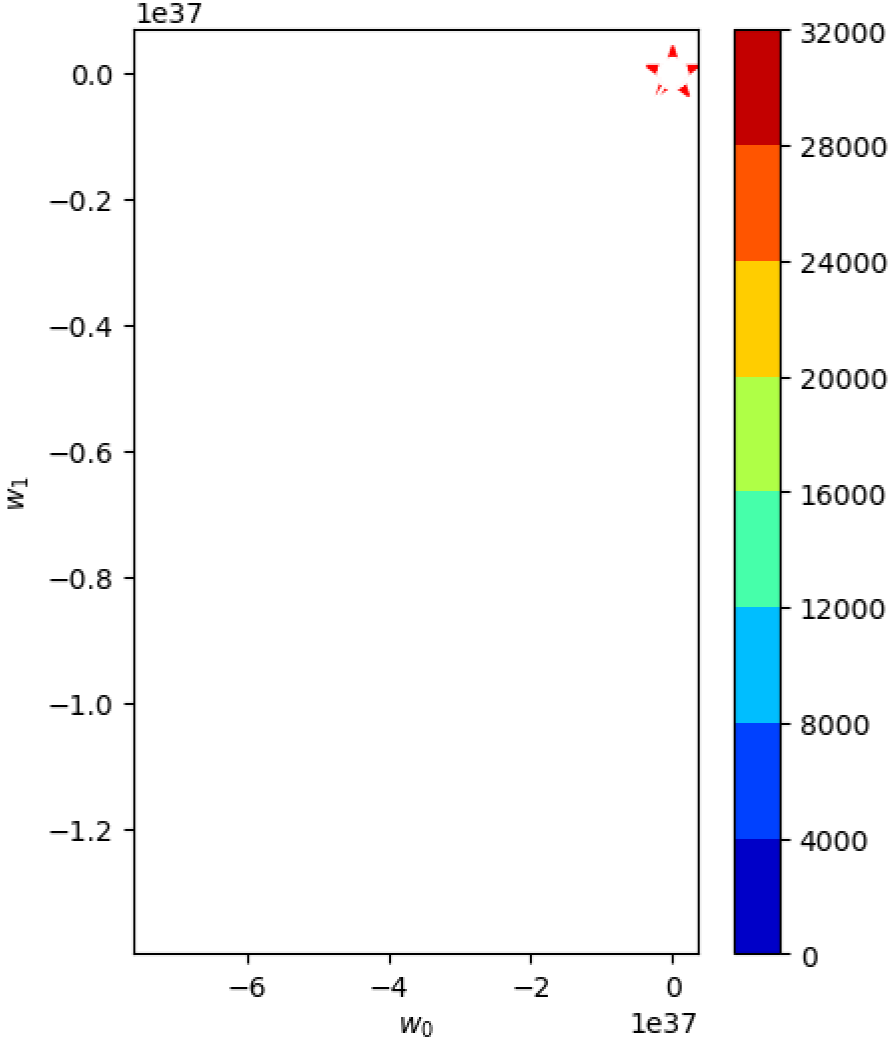


1e28
1.4
1.2
1.0
0.8
0.6
0.4
0.2
0.0
w_1
0
2
4
6
w_0
1e28
32000
28000
24000
20000
16000
12000
8000
4000
0

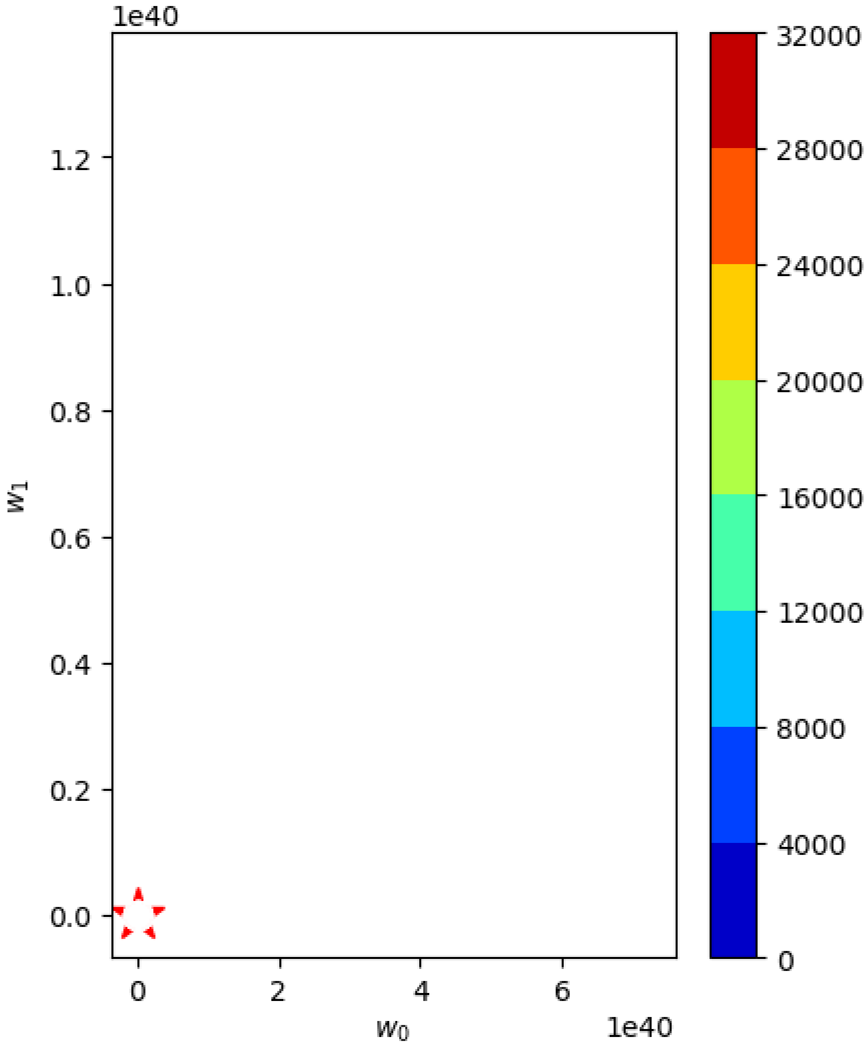


1e31
0.0
−0.2
−0.4
−0.6
−0.8
−1.0
−1.2
−1.4
w_1
−6
−4
−2
0
1e31
w_0
32000
28000
24000
20000
16000
12000
8000
4000
0

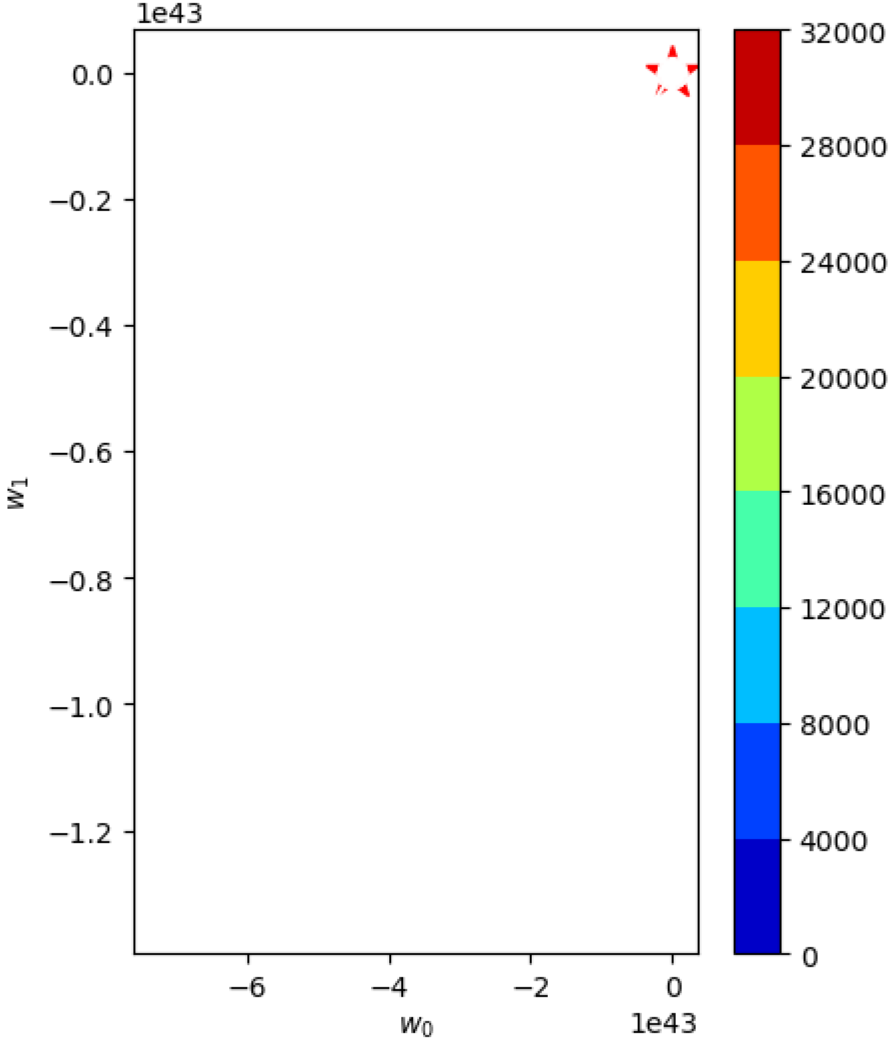

1e34
1.2
1.0
0.8
0.6
0.4
0.2
0.0
w_1
0
2
4
6
w_0
1e34
32000
28000
24000
20000
16000
12000
8000
4000
0



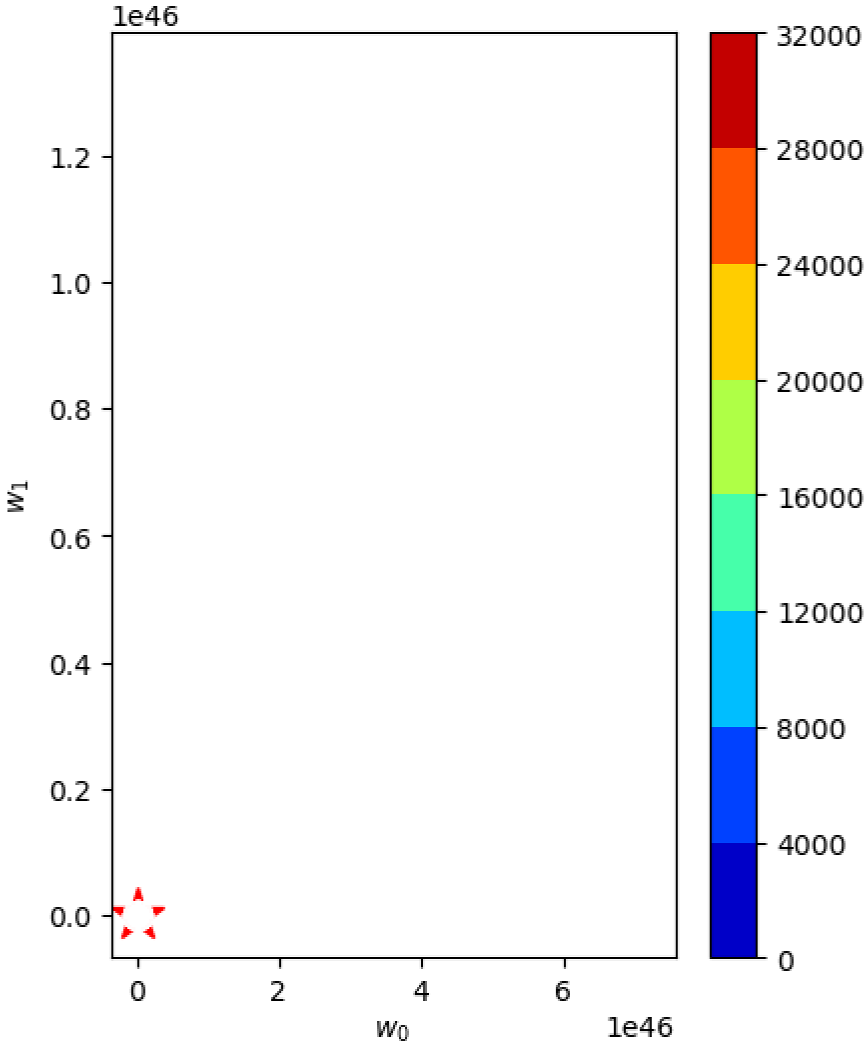
1e37
0.0
−0.2
−0.4
−0.6
−0.8
−1.0
−1.2
w_1
−6
−4
−2
0
1e37
w_0
32000
28000
24000
20000
16000
12000
8000
4000
0



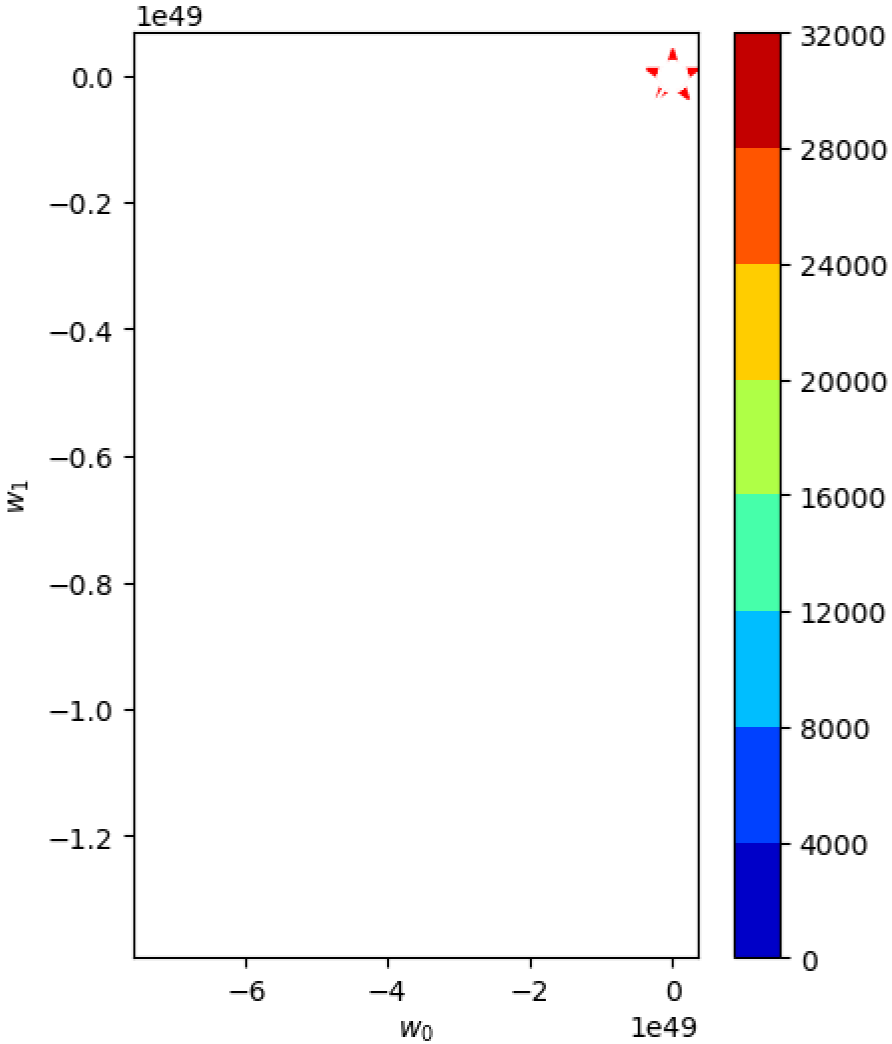
1e40
1.2
1.0
0.8
0.6
0.4
0.2
0.0
w_1
0
2
4
6
w_0
1e40
32000
28000
24000
20000
16000
12000
8000
4000
0



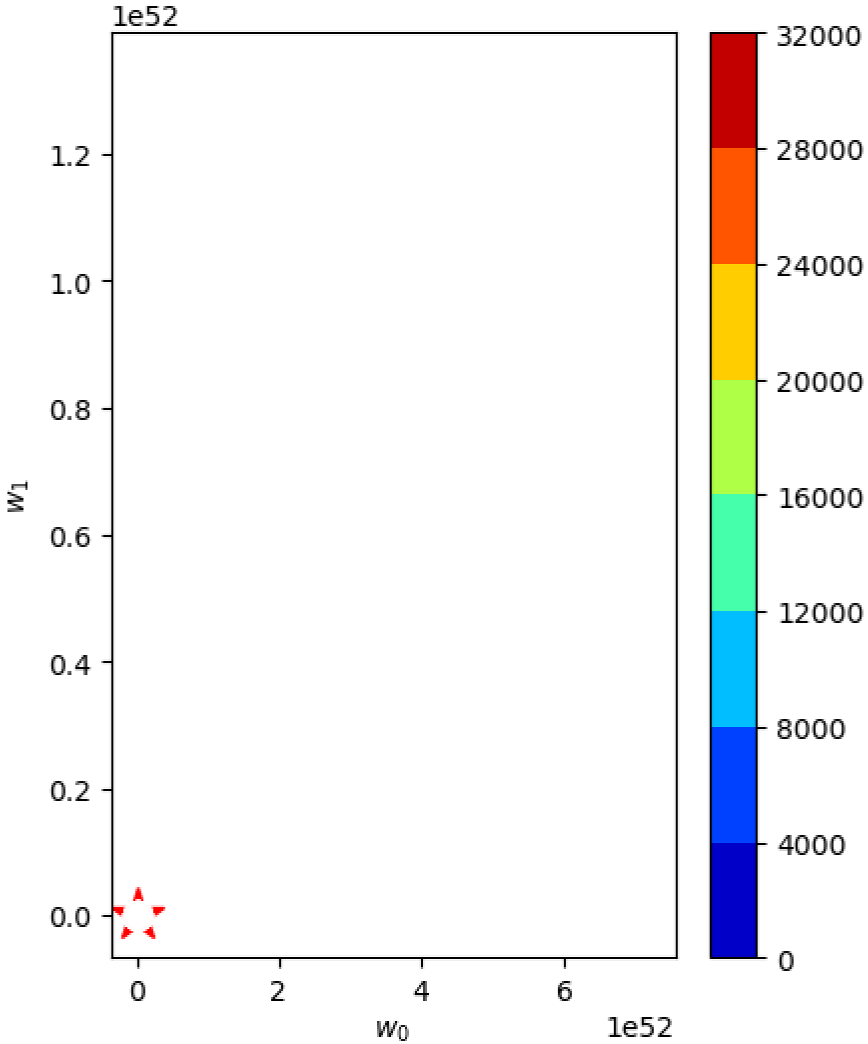
1e43
0.0
−0.2
−0.4
−0.6
−0.8
−1.0
−1.2
w_1
−6
−4
−2
0
1e43
w_0
32000
28000
24000
20000
16000
12000
8000
4000
0



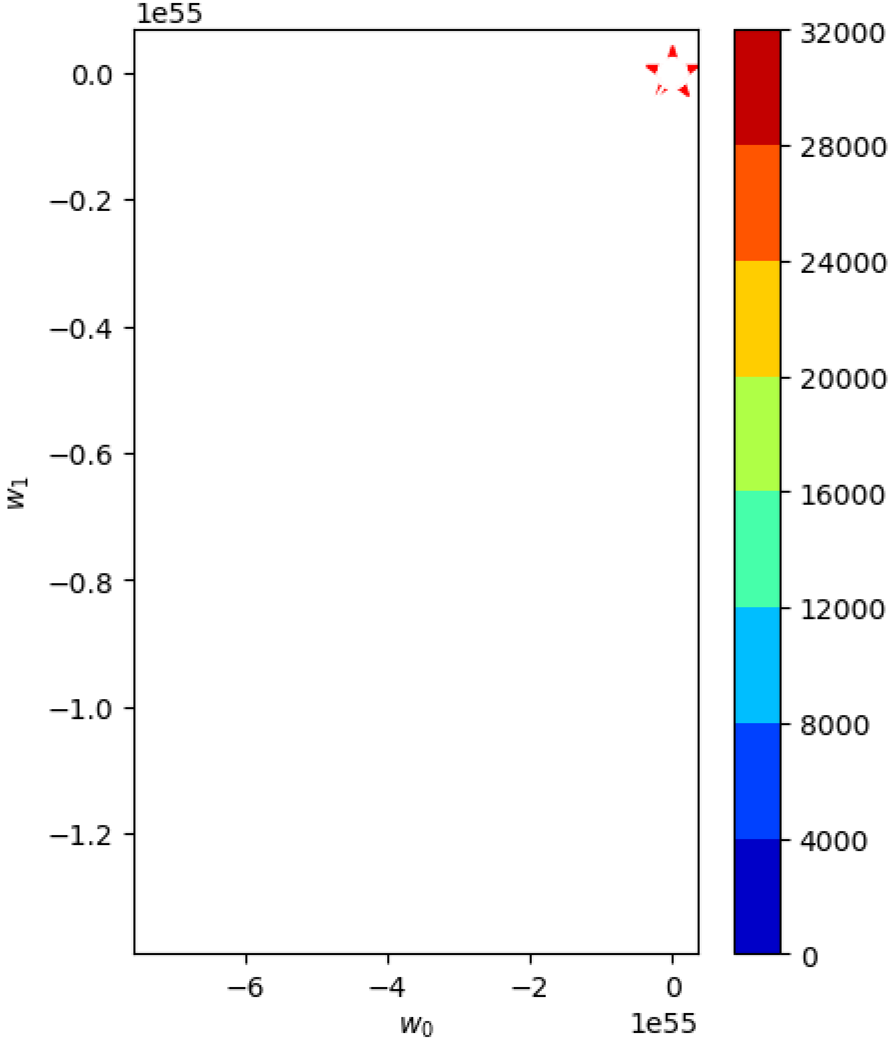
1e46
1.2
1.0
0.8
0.6
0.4
0.2
0.0
w_1
0
2
4
6
w_0
1e46
32000
28000
24000
20000
16000
12000
8000
4000
0

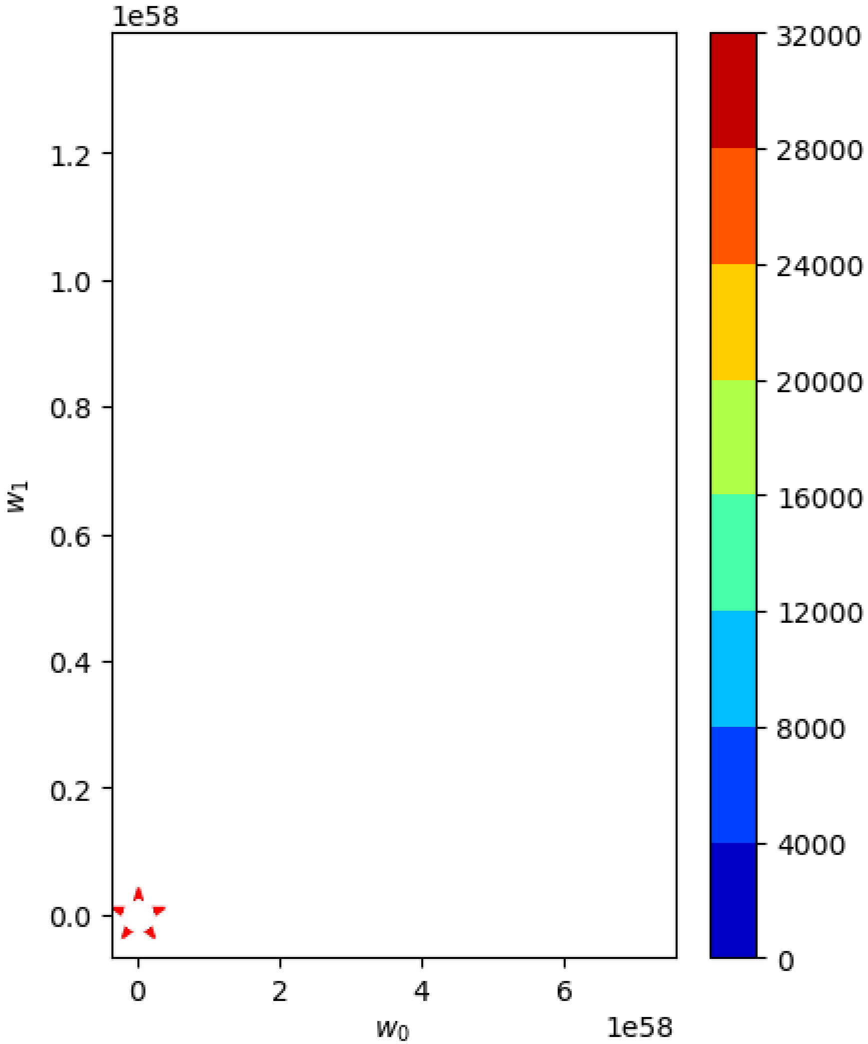


1e49
0.0
−0.2
−0.4
−0.6
−0.8
−1.0
−1.2
w_1
−6
−4
−2
0
1e49
w_0
32000
28000
24000
20000
16000
12000
8000
4000
0

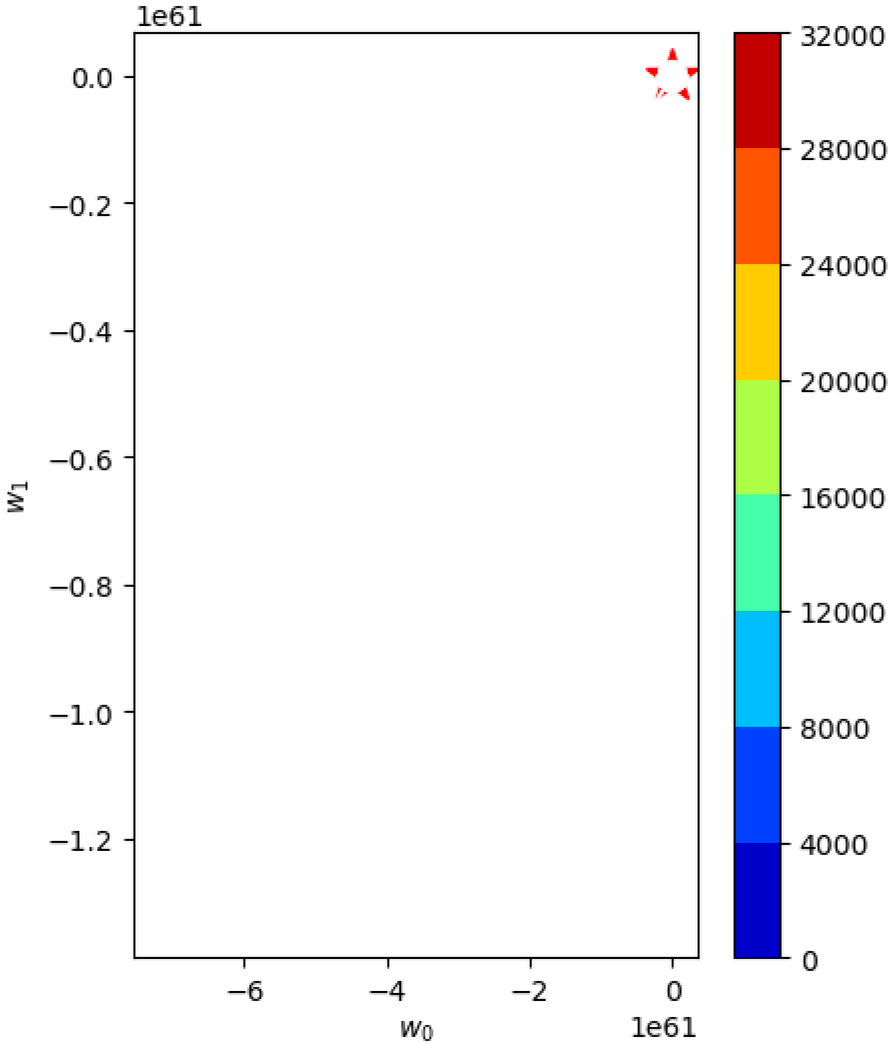


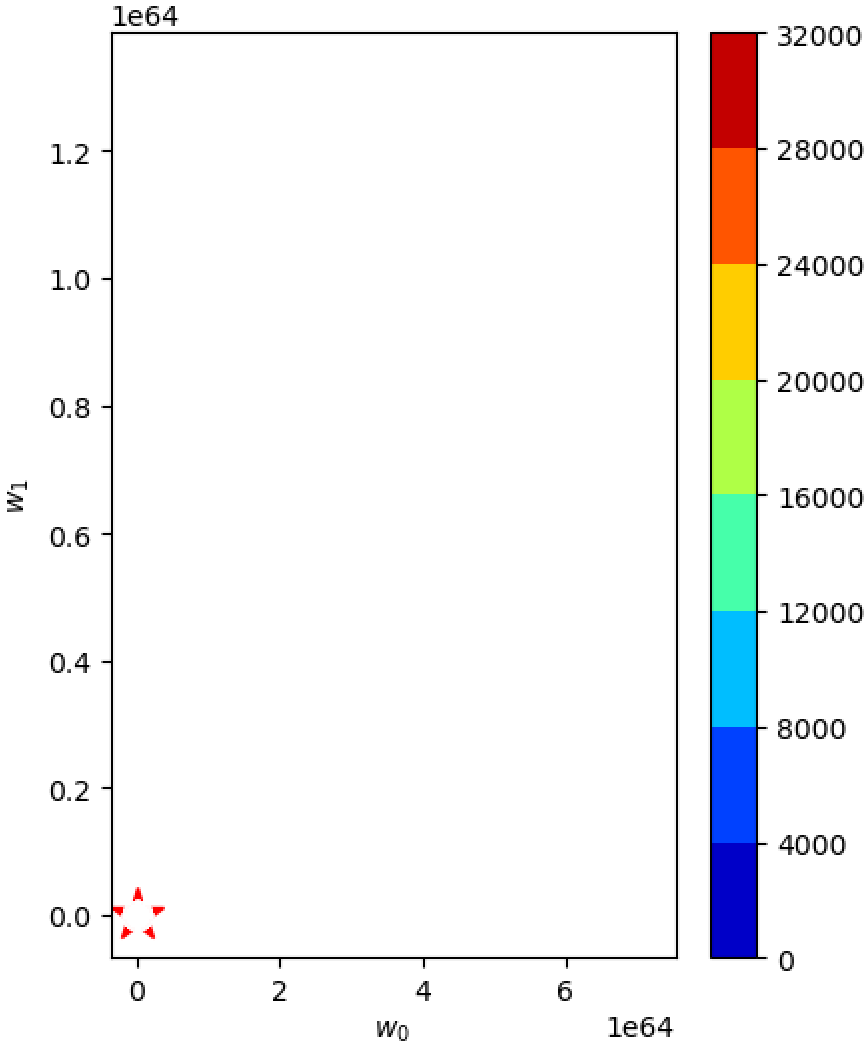
1e52
1.2
1.0
0.8
0.6
0.4
0.2
0.0
w_1
0
2
4
6
w_0
1e52
32000
28000
24000
20000
16000
12000
8000
4000
0


1e55
0.0
−0.2
−0.4
−0.6
−0.8
−1.0
−1.2
w_1
−6
−4
−2
0
1e55
w_0
32000
28000
24000
20000
16000
12000
8000
4000
0

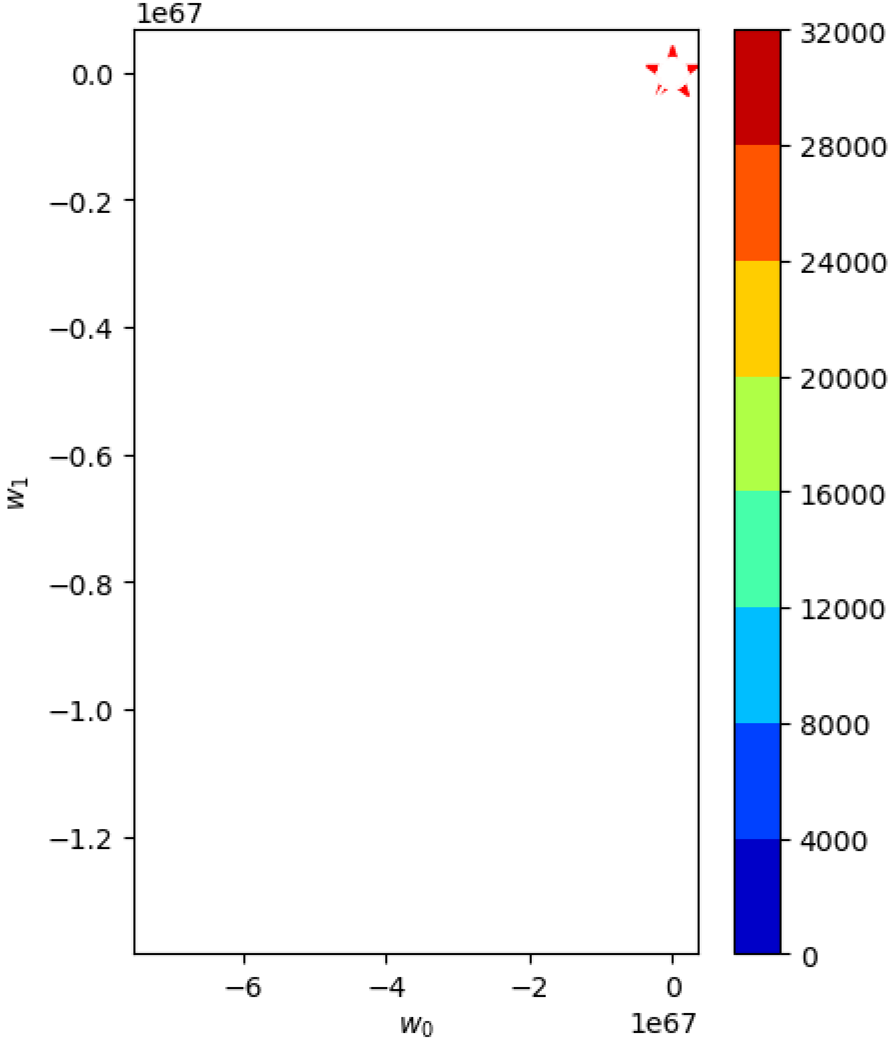


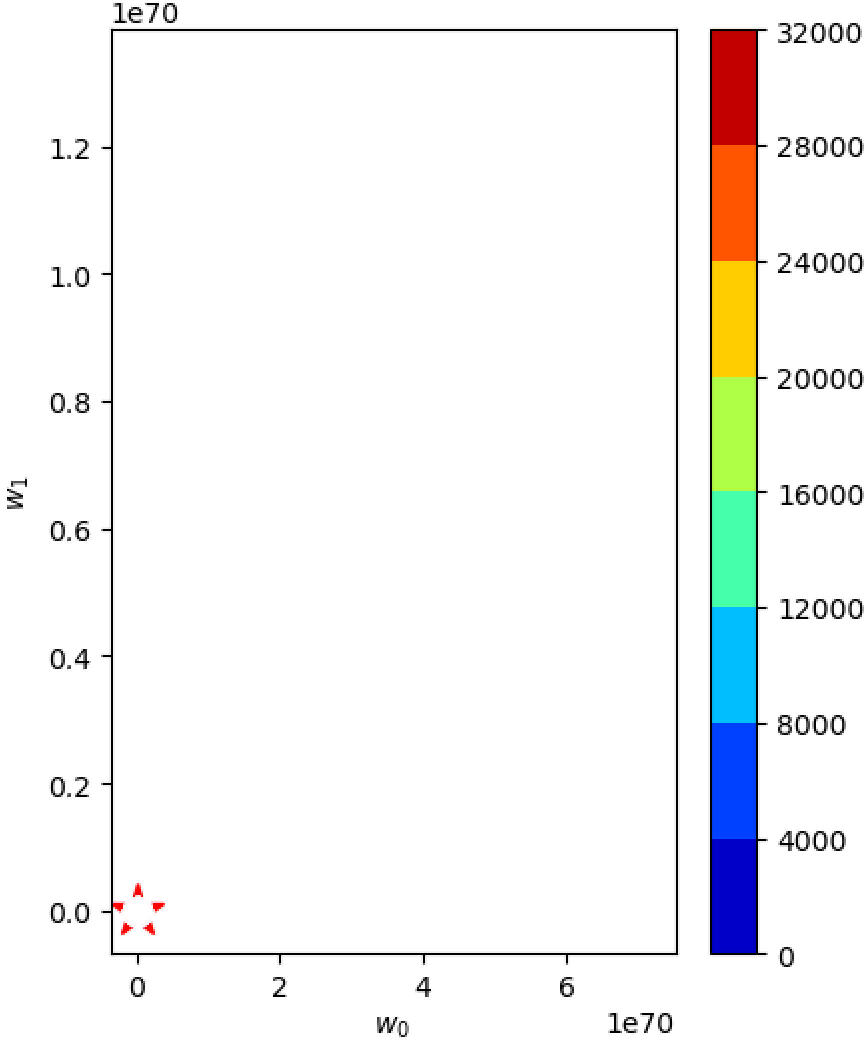
1e58
1.2
1.0
0.8
0.6
0.4
0.2
0.0
w_1
0
2
4
6
w_0
1e58
32000
28000
24000
20000
16000
12000
8000
4000
0


1e61
0.0
−0.2
−0.4
−0.6
−0.8
−1.0
−1.2
w_1
−6
−4
−2
0
w_0
1e61
32000
28000
24000
20000
16000
12000
8000
4000
0

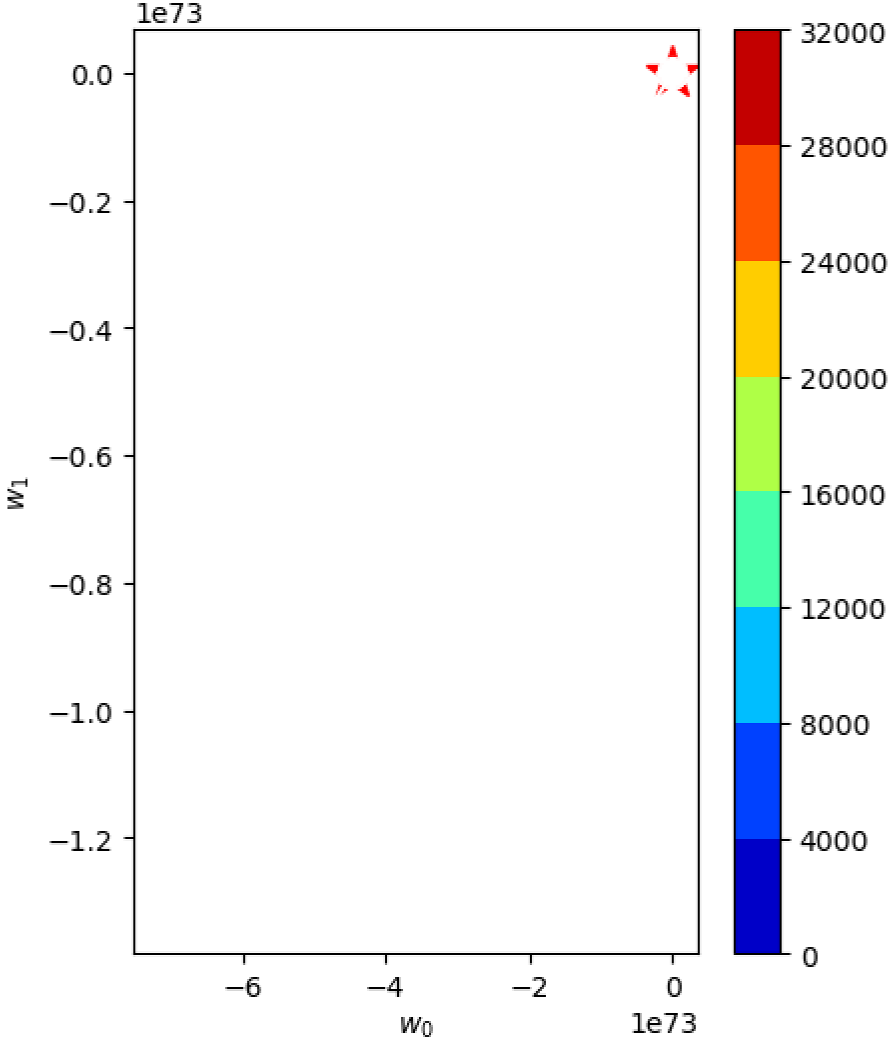


1e64
1.2
1.0
0.8
0.6
0.4
0.2
0.0
w_1
0
2
4
6
w_0
1e64
32000
28000
24000
20000
16000
12000
8000
4000
0

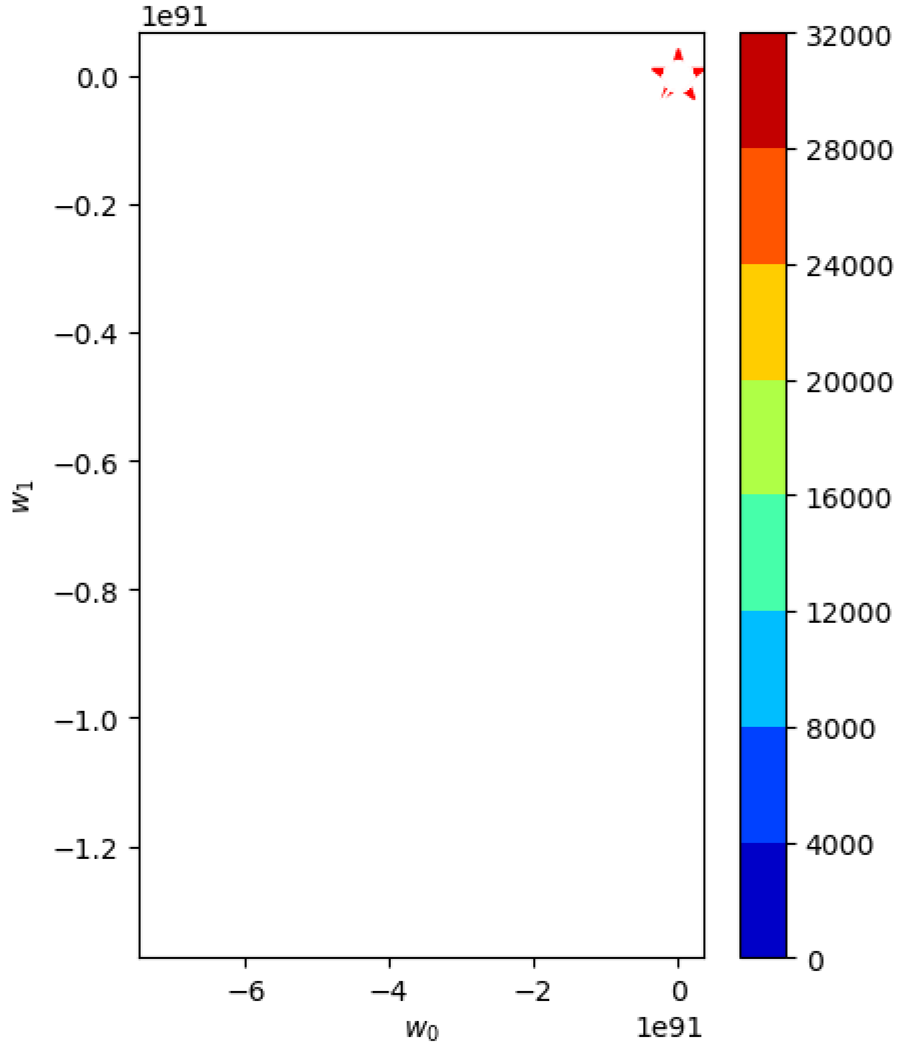

1e67
0.0
−0.2
−0.4
−0.6
−0.8
−1.0
−1.2
w_1
−6
−4
−2
0
1e67
w_0
32000
28000
24000
20000
16000
12000
8000
4000
0



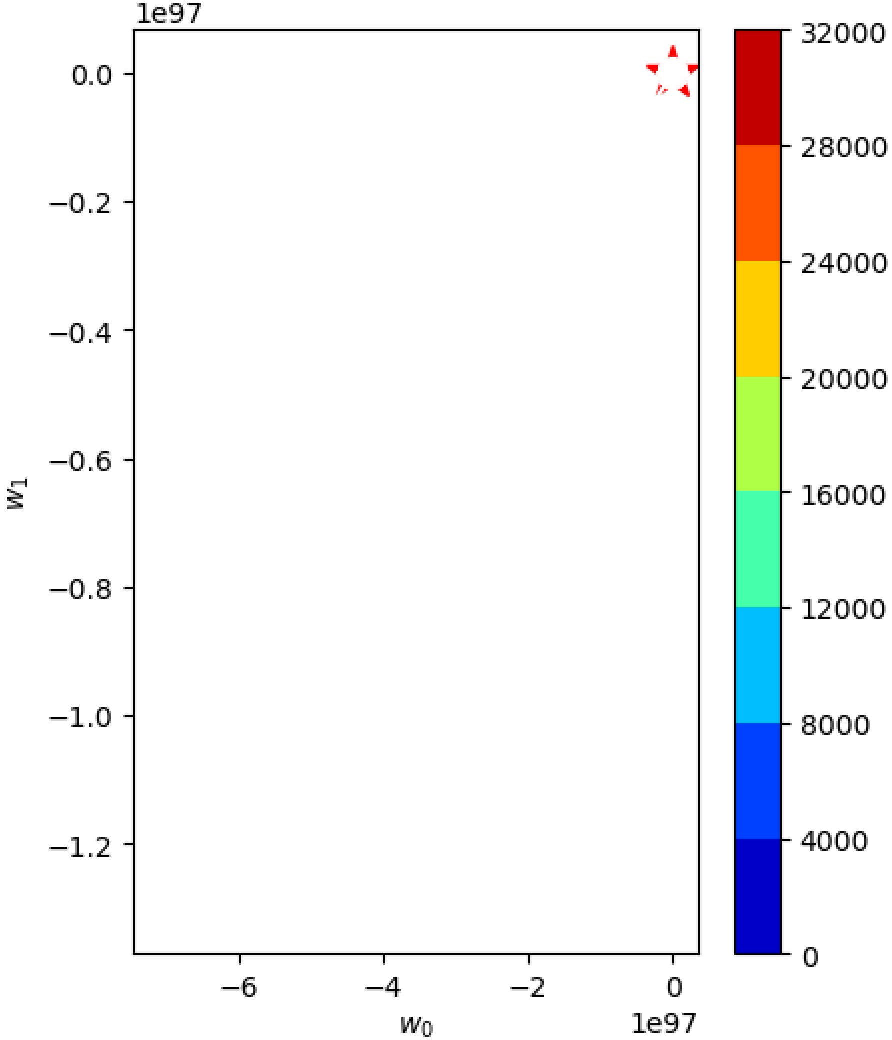
1e70
1.2
1.0
0.8
0.6
0.4
0.2
0.0
w_1
0
2
4
6
w_0
1e70
32000
28000
24000
20000
16000
12000
8000
4000
0



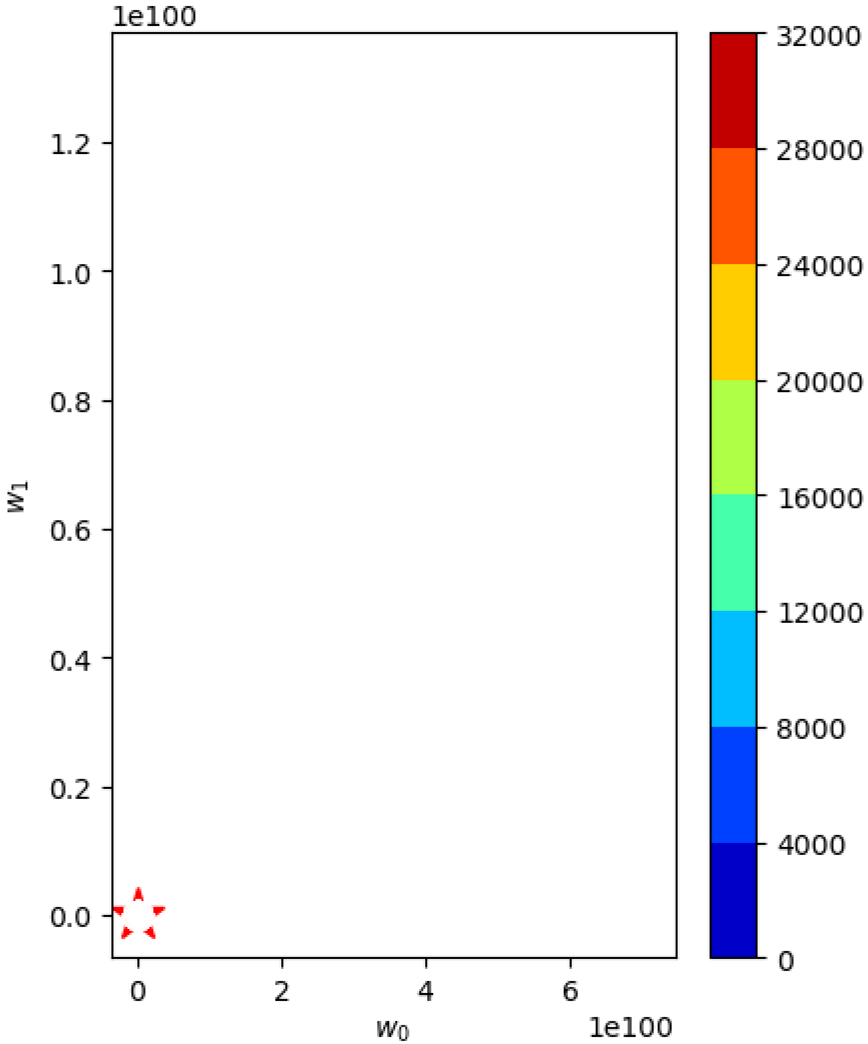
1e73
0.0
−0.2
−0.4
−0.6
−0.8
−1.0
−1.2
w_1
−6
−4
−2
0
1e73
w_0
32000
28000
24000
20000
16000
12000
8000
4000
0



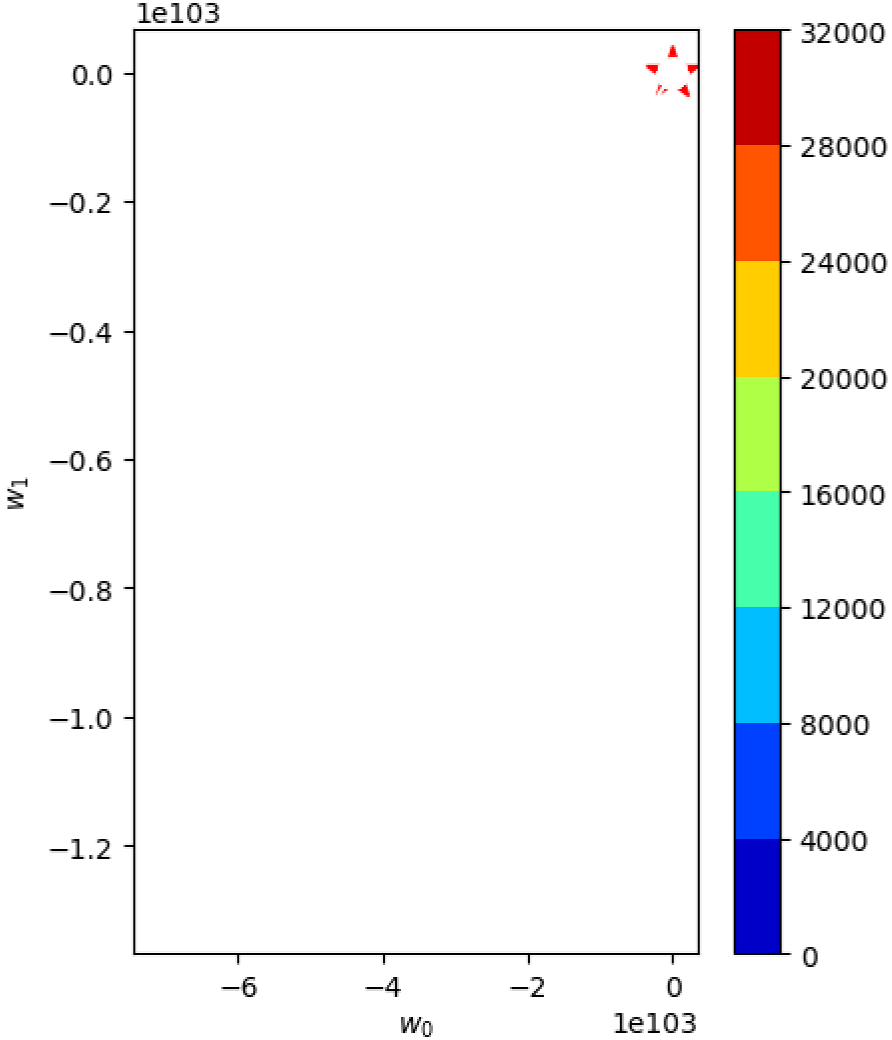
1e91
0.0
−0.2
−0.4
−0.6
−0.8
−1.0
−1.2
w_1
−6
−4
−2
0
1e91
w_0
32000
28000
24000
20000
16000
12000
8000
4000
0



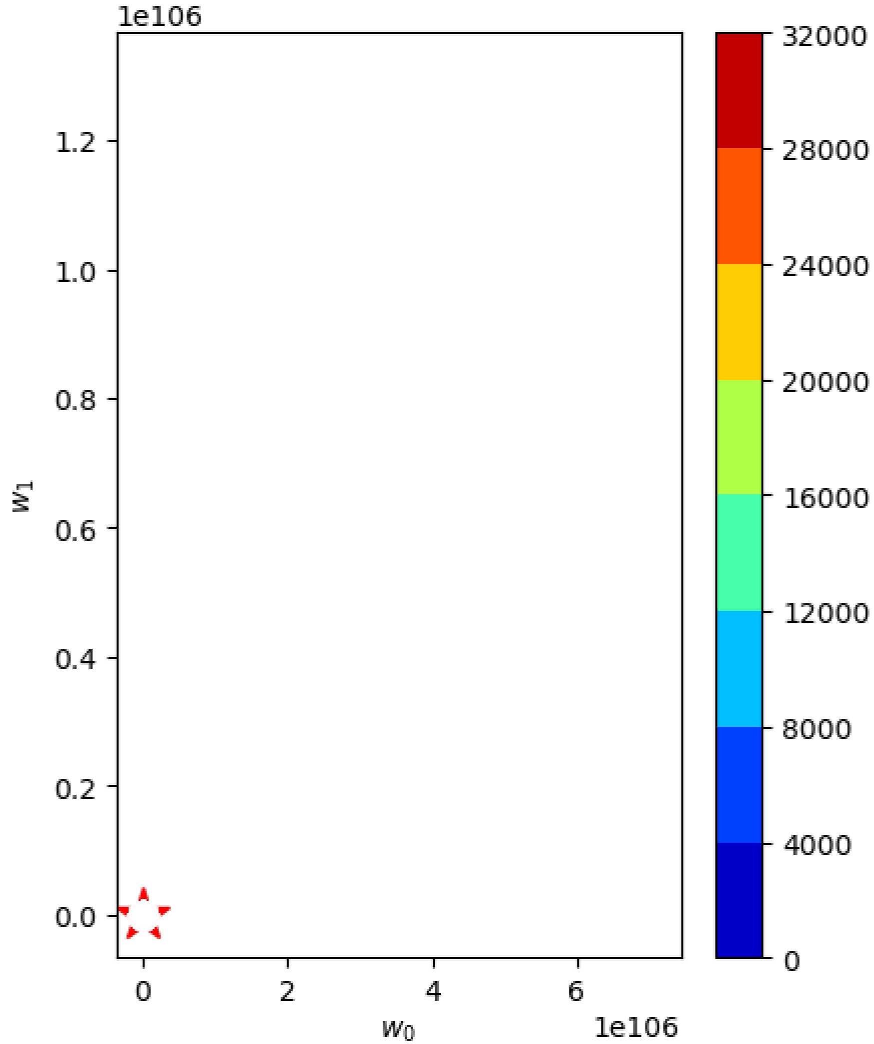
1e97
0.0
−0.2
−0.4
−0.6
−0.8
−1.0
−1.2
w_1
−6
−4
−2
0
w_0
1e97
32000
28000
24000
20000
16000
12000
8000
4000
0



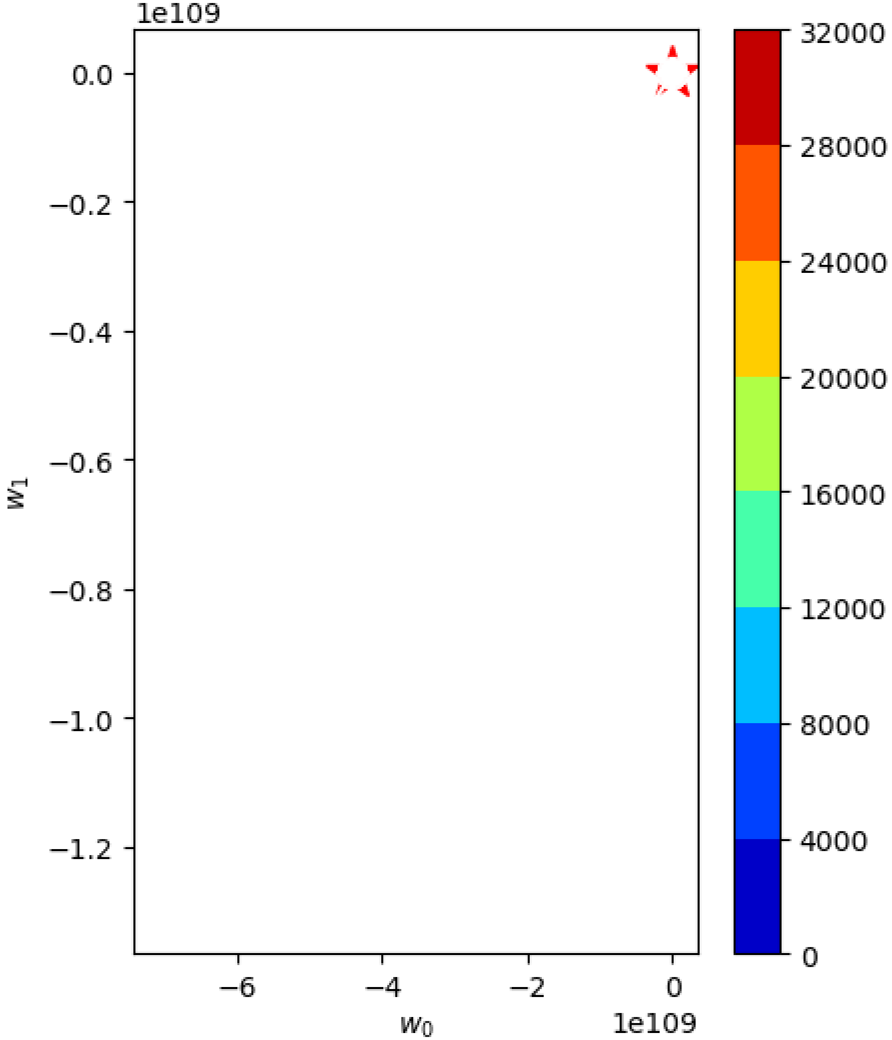
1e100
1.2
1.0
0.8
0.6
0.4
0.2
0.0
w_1
0
2
4
6
w_0
1e100
32000
28000
24000
20000
16000
12000
8000
4000
0

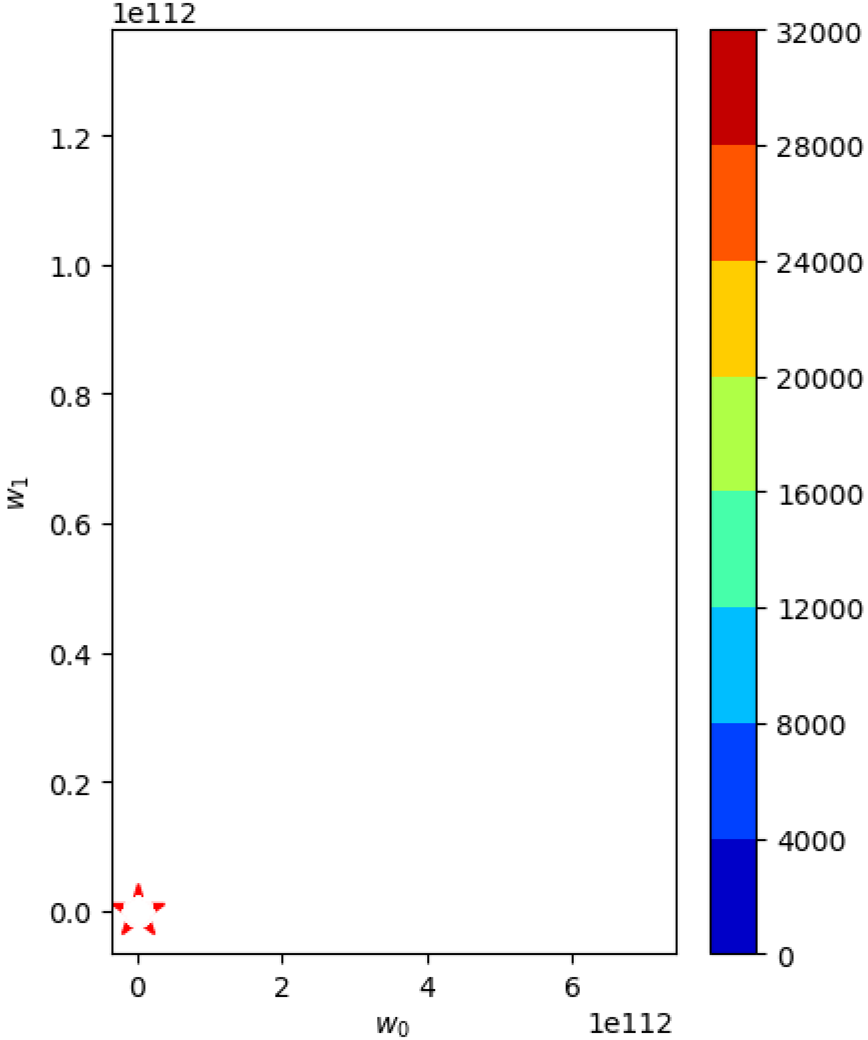


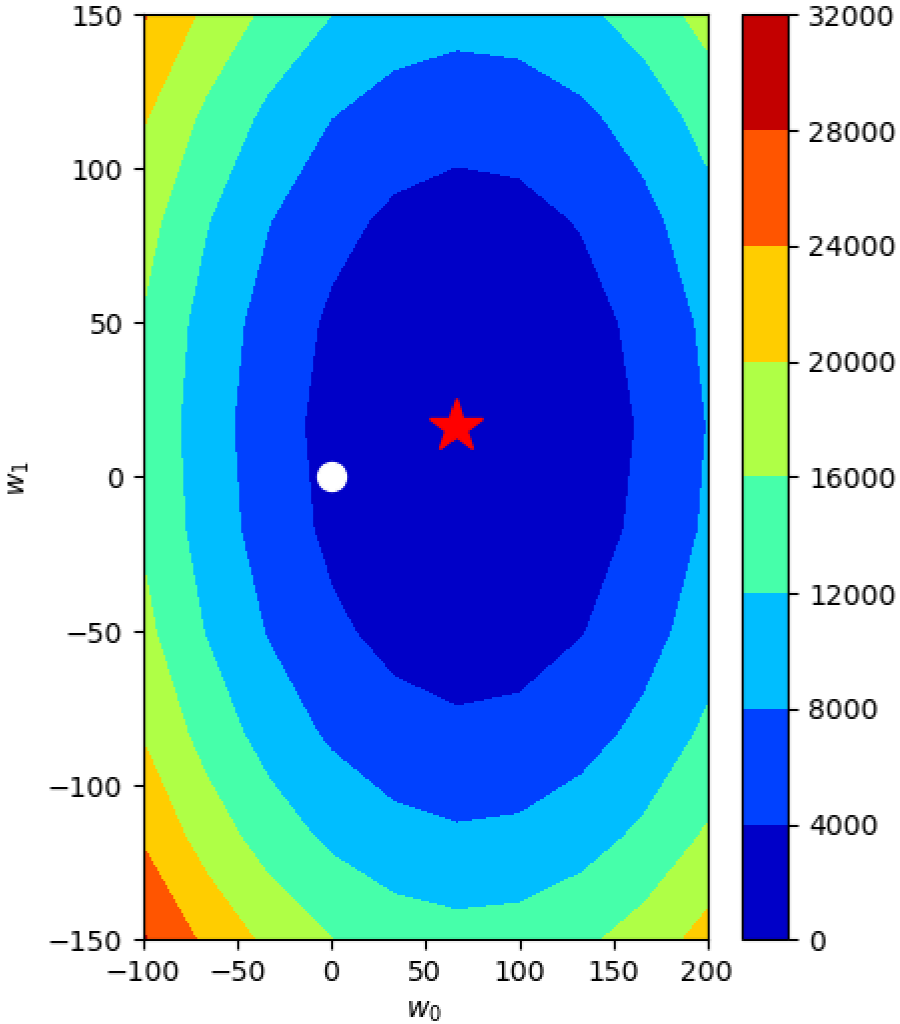
1e103
0.0
−0.2
−0.4
−0.6
−0.8
−1.0
−1.2
w_1
−6
−4
−2
0
1e103
w_0
32000
28000
24000
20000
16000
12000
8000
4000
0



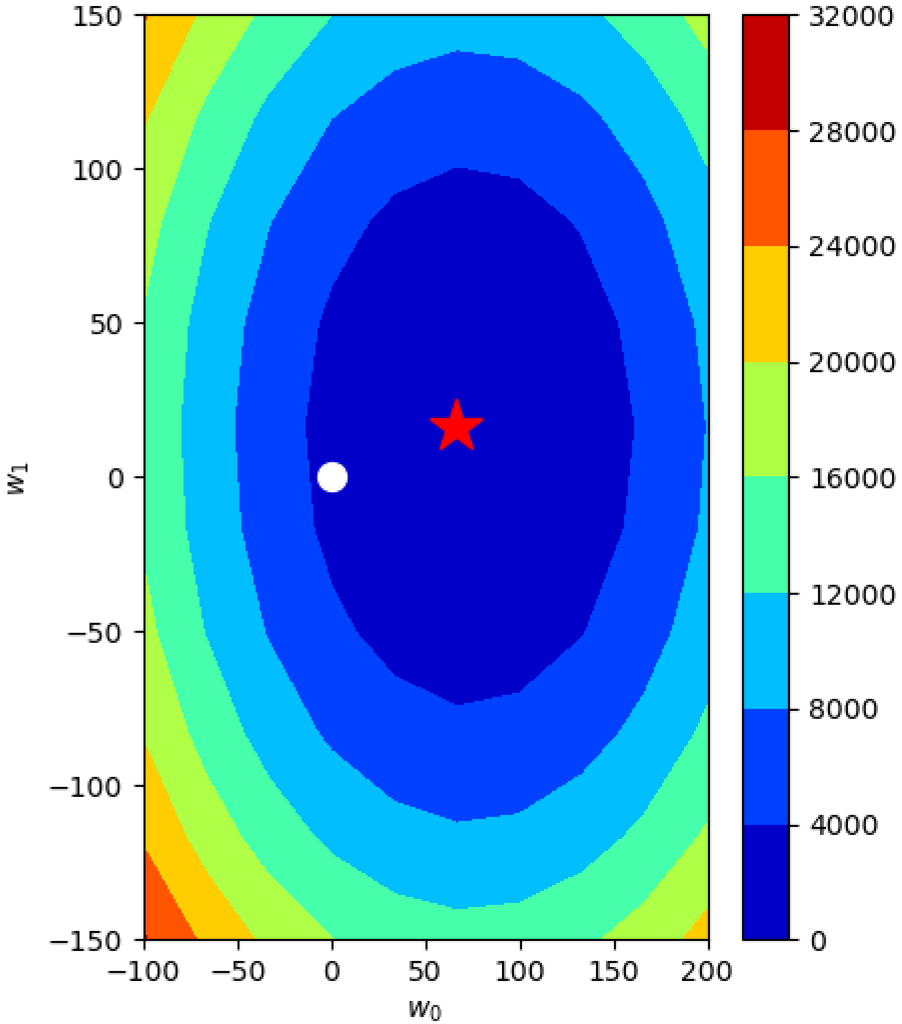
1e106
1.2
1.0
0.8
0.6
0.4
0.2
0.0
w_1
0
2
4
6
w_0
1e106
32000
28000
24000
20000
16000
12000
8000
4000
0

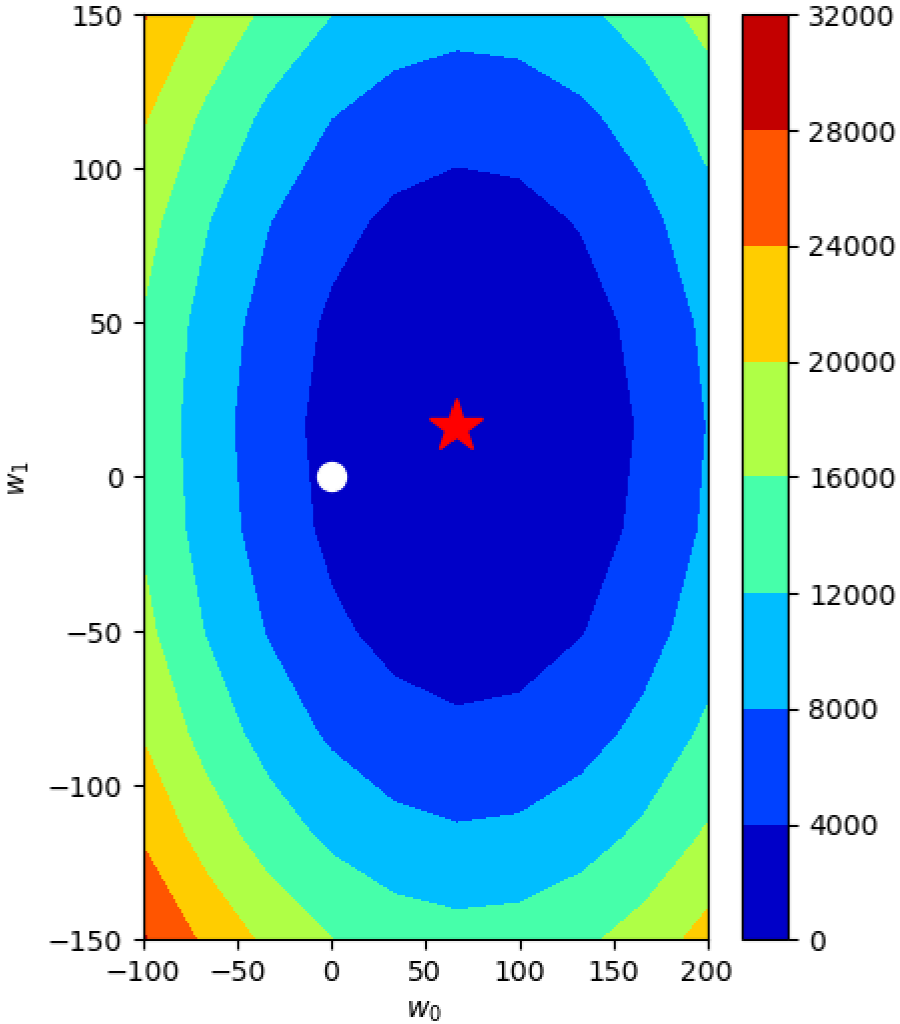

1e109
0.0
−0.2
−0.4
−0.6
−0.8
−1.0
−1.2
w_1
−6
−4
−2
0
w_0
1e109
32000
28000
24000
20000
16000
12000
8000
4000
0


1e112
1.2
1.0
0.8
0.6
0.4
0.2
0.0
w_1
0
2
4
6
w_0
1e112
32000
28000
24000
20000
16000
12000
8000
4000
0

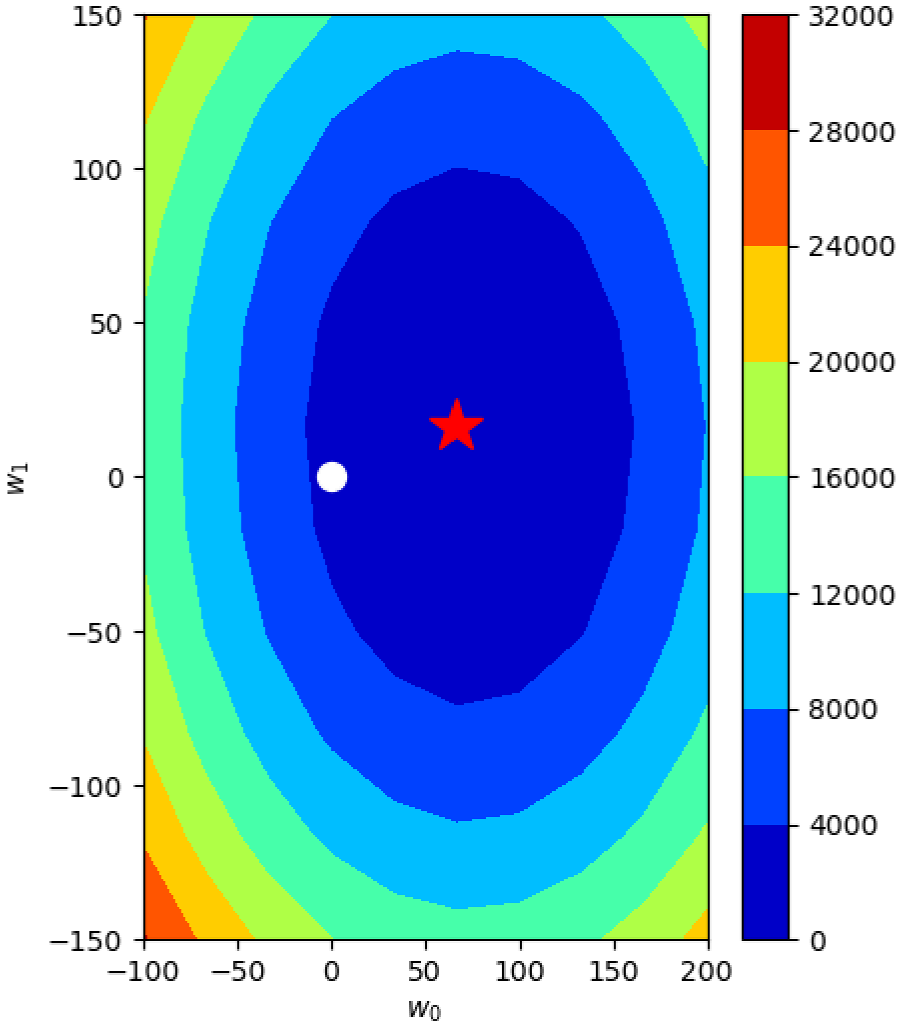


150
100
50
0
−50
−100
−150
w_1
−100
−50
0
50
100
150
200
w_0
32000
28000
24000
20000
16000
12000
8000
4000
0

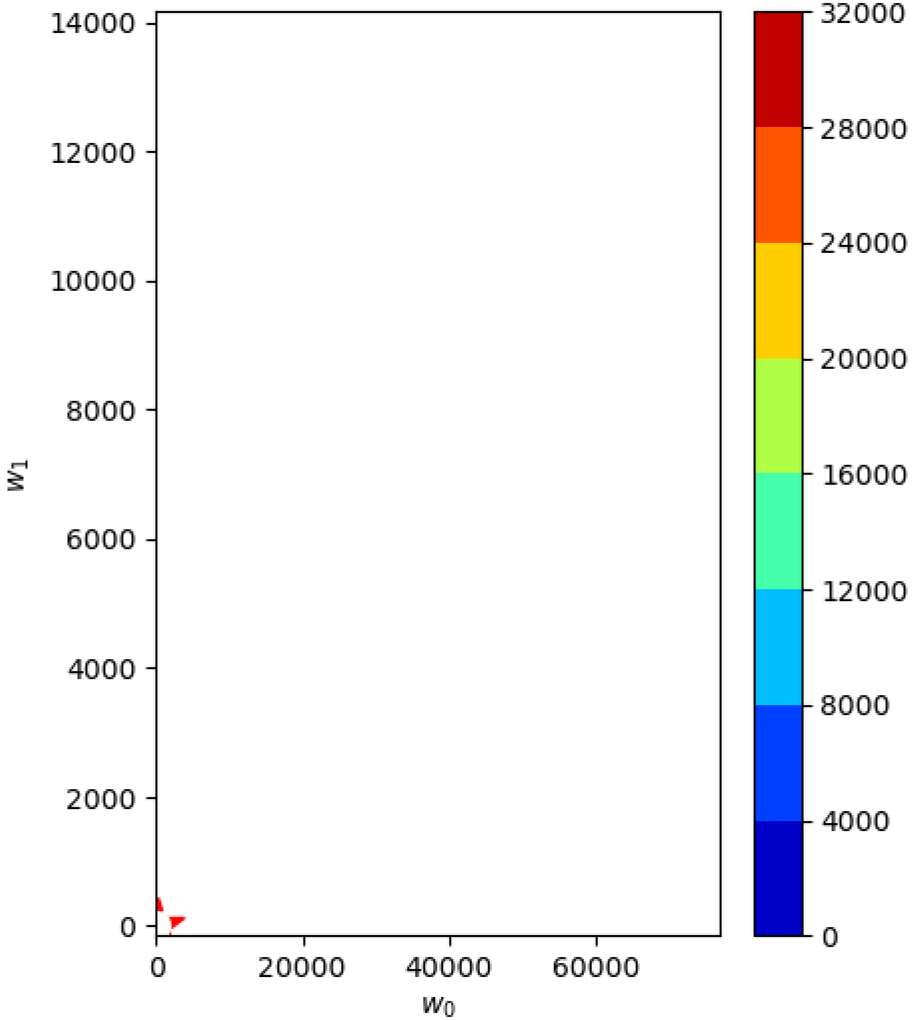

150
100
50
0
−50
−100
−150
w_1
−100
−50
0
50
100
150
200
w_0
32000
28000
24000
20000
16000
12000
8000
4000
0

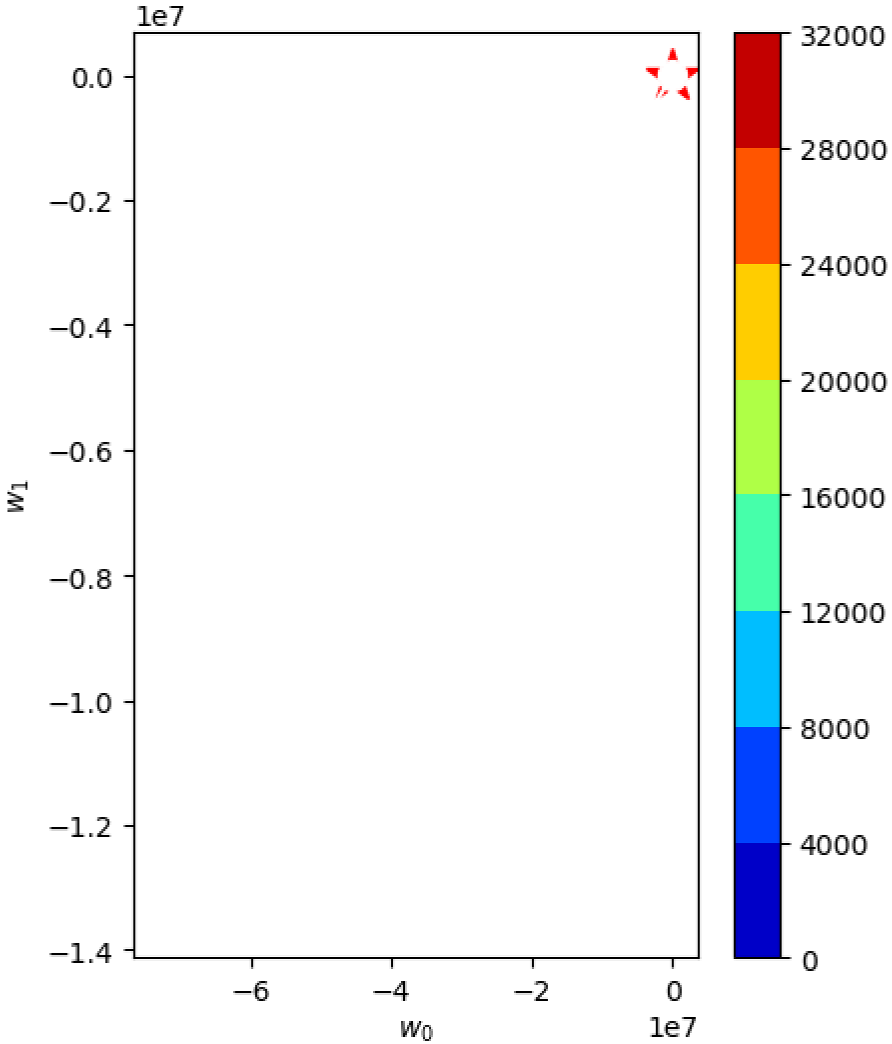


150
100
50
0
−50
−100
−150
w_1
−100
−50
0
50
100
150
200
w_0
32000
28000
24000
20000
16000
12000
8000
4000
0

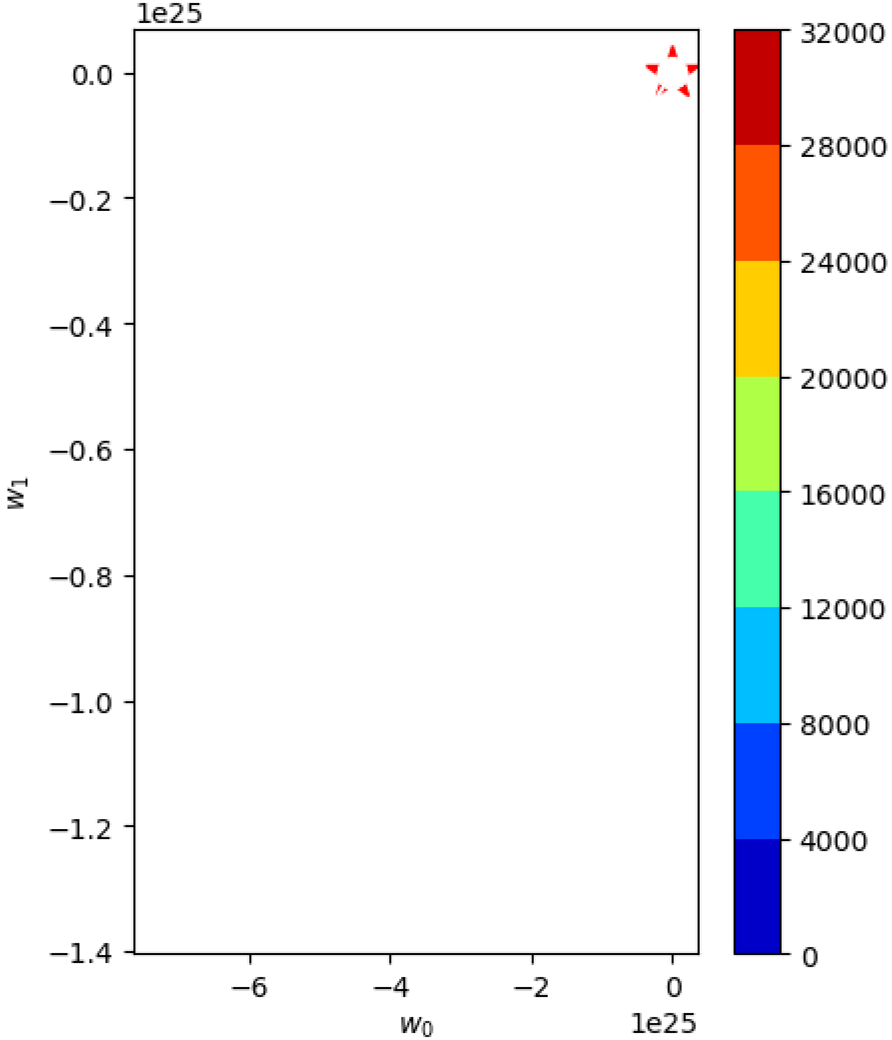


150
100
50
0
−50
−100
−150
w_1
−100
−50
0
50
100
150
200
w_0
32000
28000
24000
20000
16000
12000
8000
4000
0

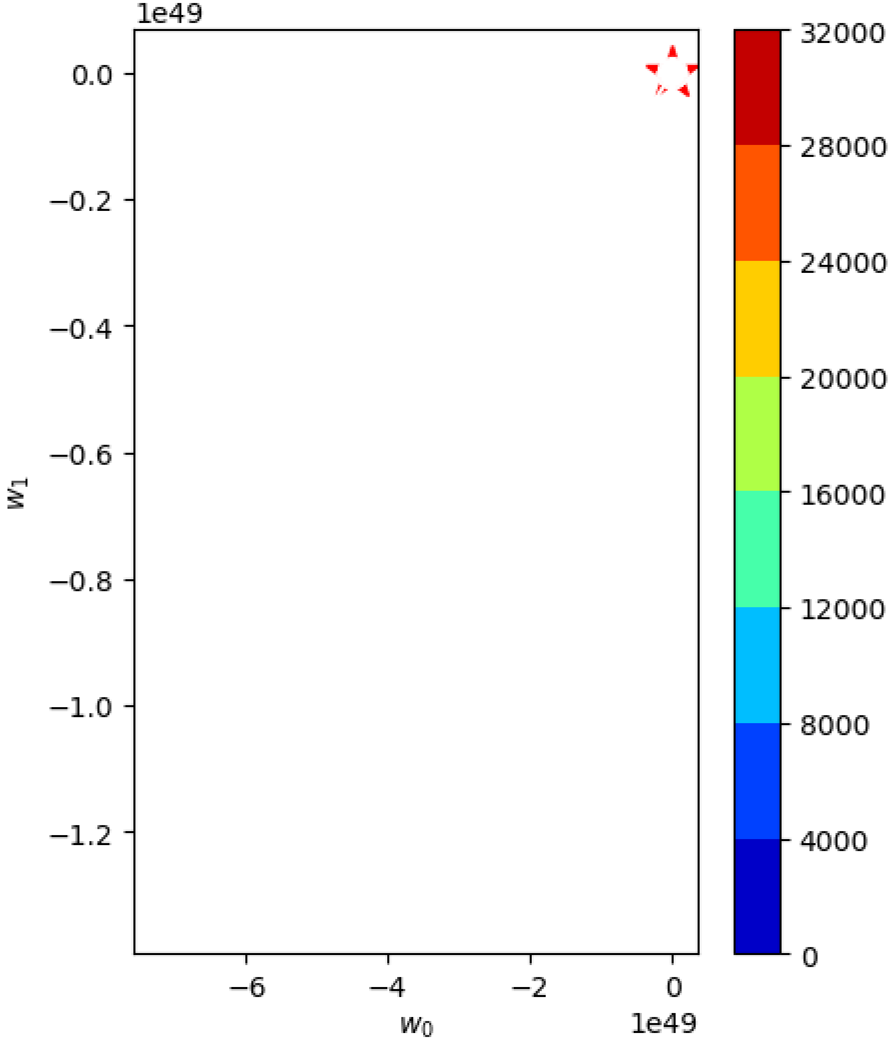

14000
12000
10000
8000
6000
4000
2000
0
w_1
0
20000
40000
60000
w_0
32000
28000
24000
20000
16000
12000
8000
4000
0



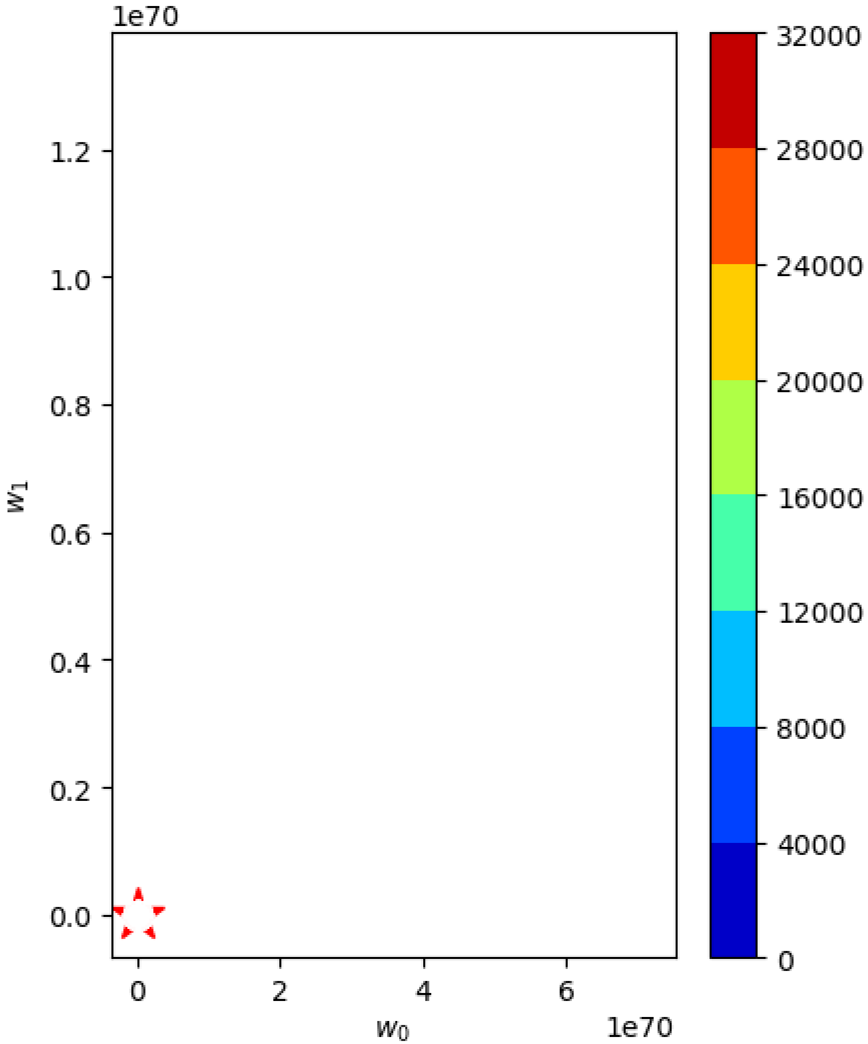
1e7
0.0
−0.2
−0.4
−0.6
−0.8
−1.0
−1.2
−1.4
w_1
−6
−4
−2
0
1e7
w_0
32000
28000
24000
20000
16000
12000
8000
4000
0

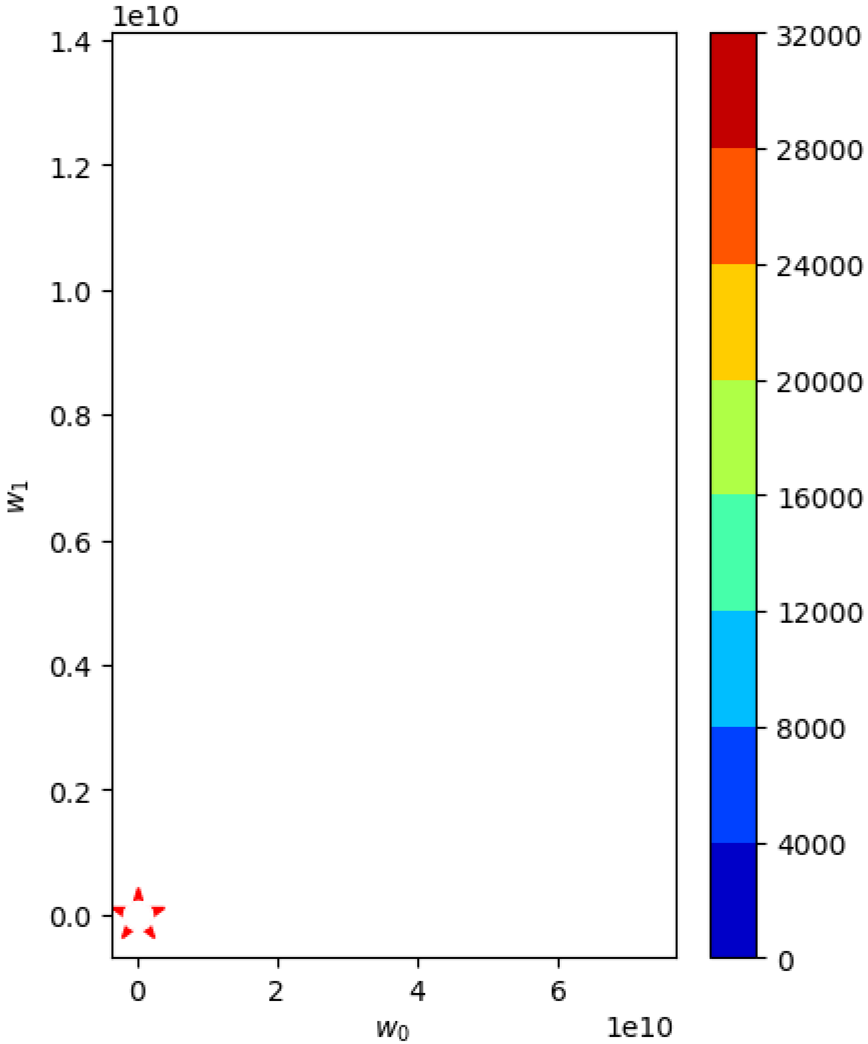


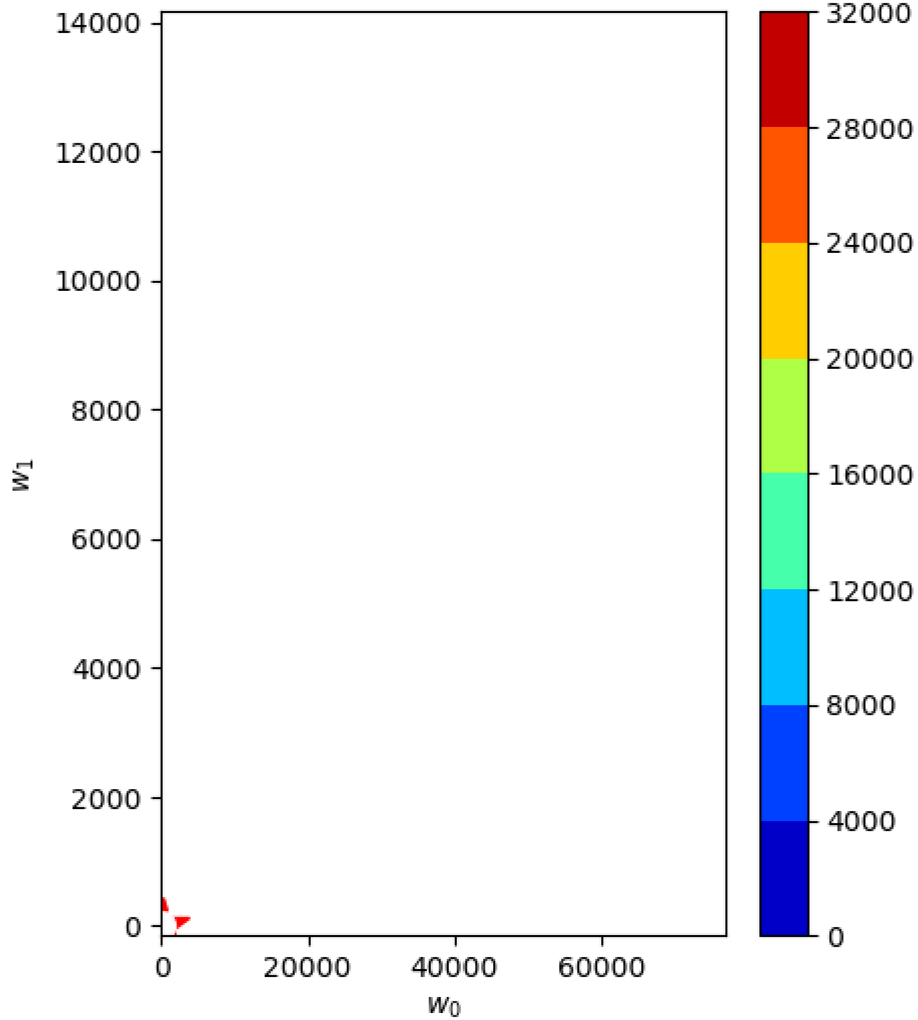
1e25
0.0
−0.2
−0.4
−0.6
−0.8
−1.0
−1.2
−1.4
w_1
−6
−4
−2
0
1e25
w_0
32000
28000
24000
20000
16000
12000
8000
4000
0



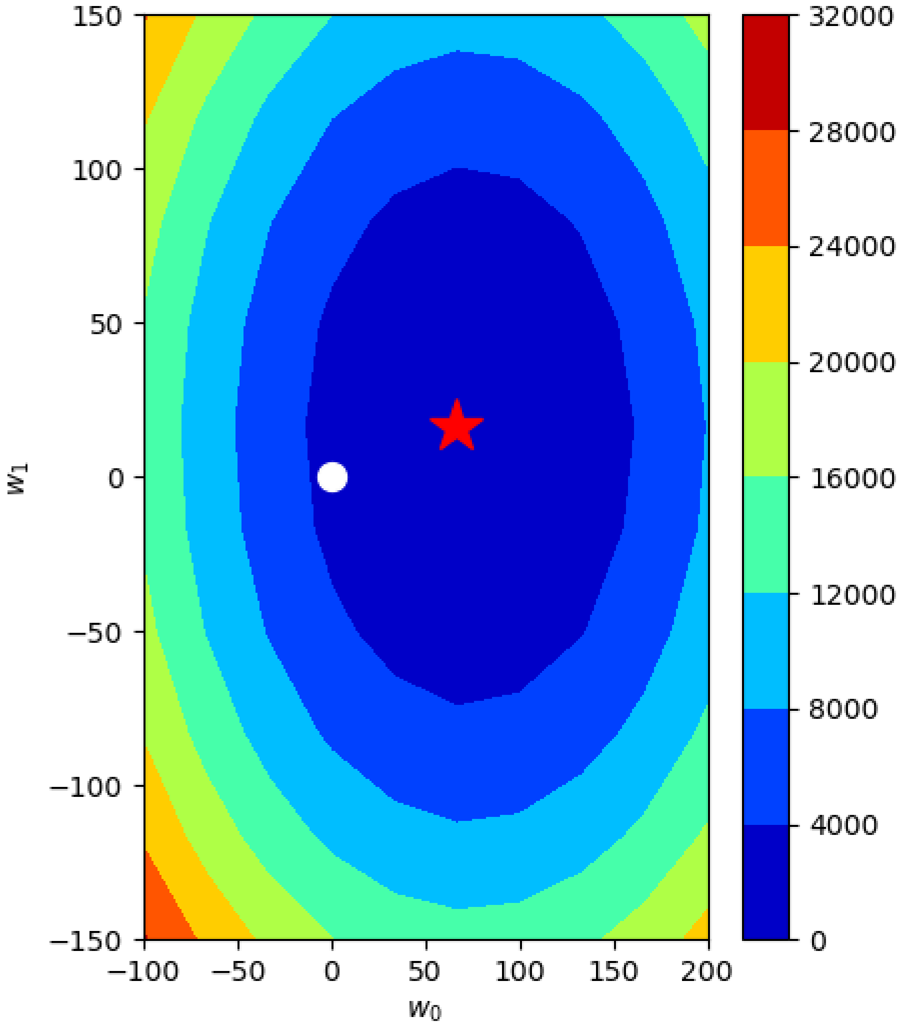
1e49
0.0
−0.2
−0.4
−0.6
−0.8
−1.0
−1.2
w_1
−6
−4
−2
0
1e49
w_0
32000
28000
24000
20000
16000
12000
8000
4000
0


1e70
1.2
1.0
0.8
0.6
0.4
0.2
0.0
w_1
0
2
4
6
w_0
1e70
32000
28000
24000
20000
16000
12000
8000
4000
0

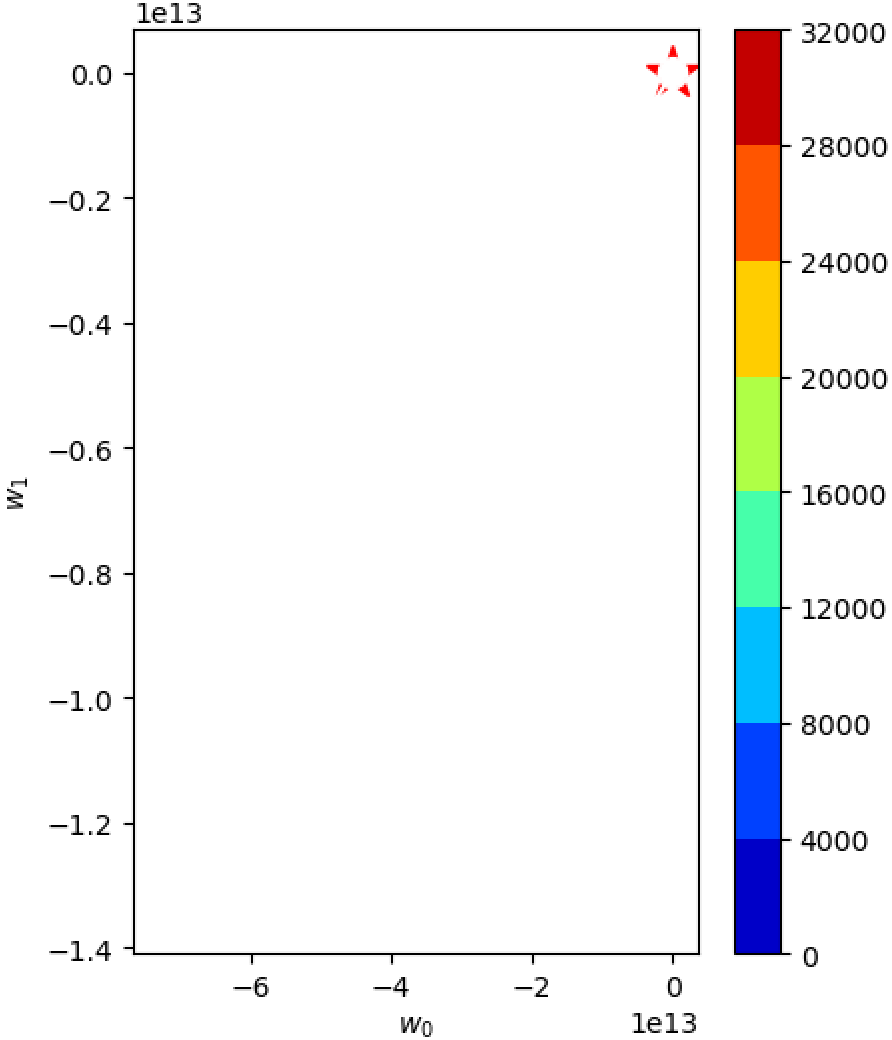

1e10
1.4
1.2
1.0
0.8
0.6
0.4
0.2
0.0
w_1
0
2
4
6
w_0
1e10
32000
28000
24000
20000
16000
12000
8000
4000
0



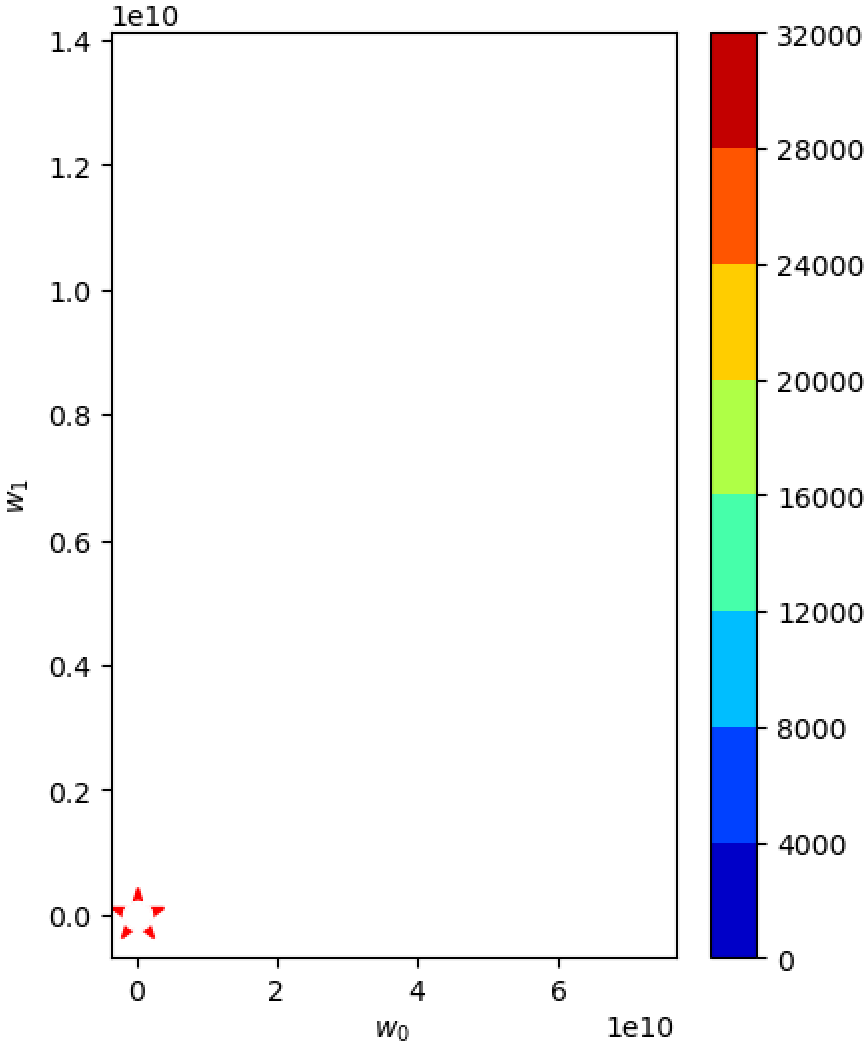
14000
12000
10000
8000
6000
4000
2000
0
w_1
0
20000
40000
60000
w_0
32000
28000
24000
20000
16000
12000
8000
4000
0



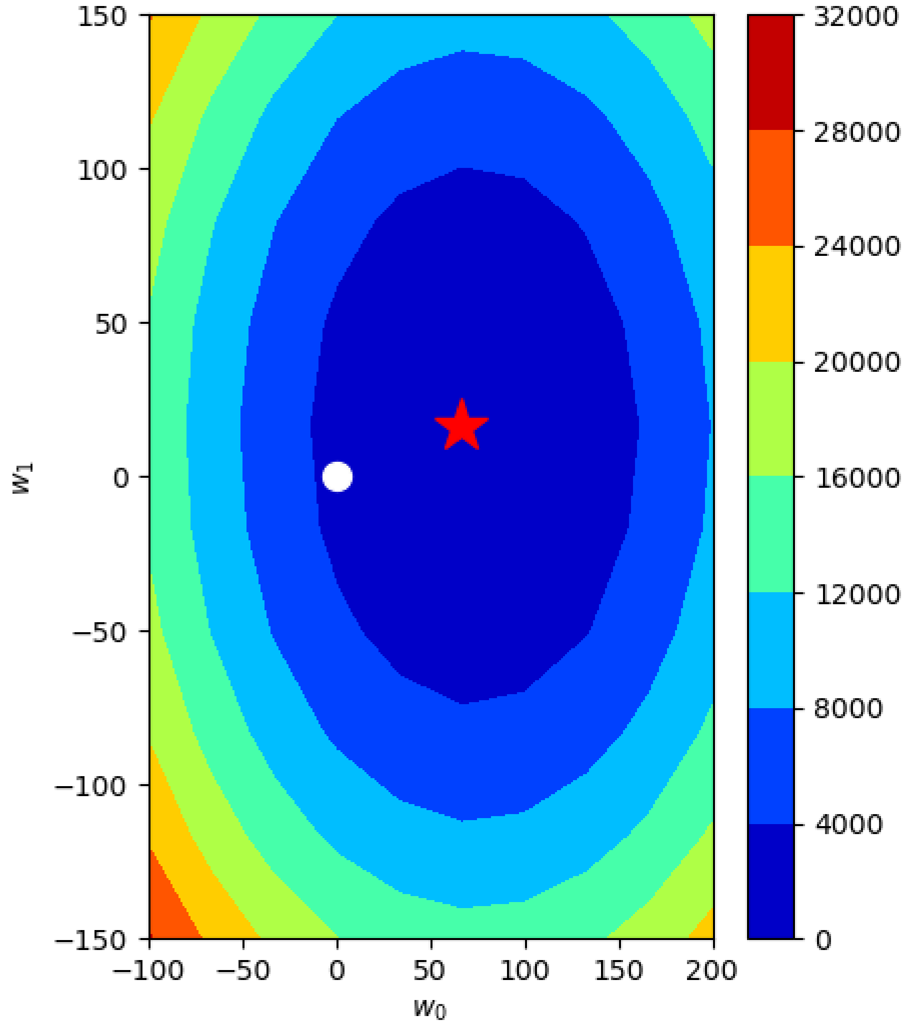
150
100
50
0
−50
−100
−150
w_1
−100
−50
0
50
100
150
200
w_0
32000
28000
24000
20000
16000
12000
8000
4000
0



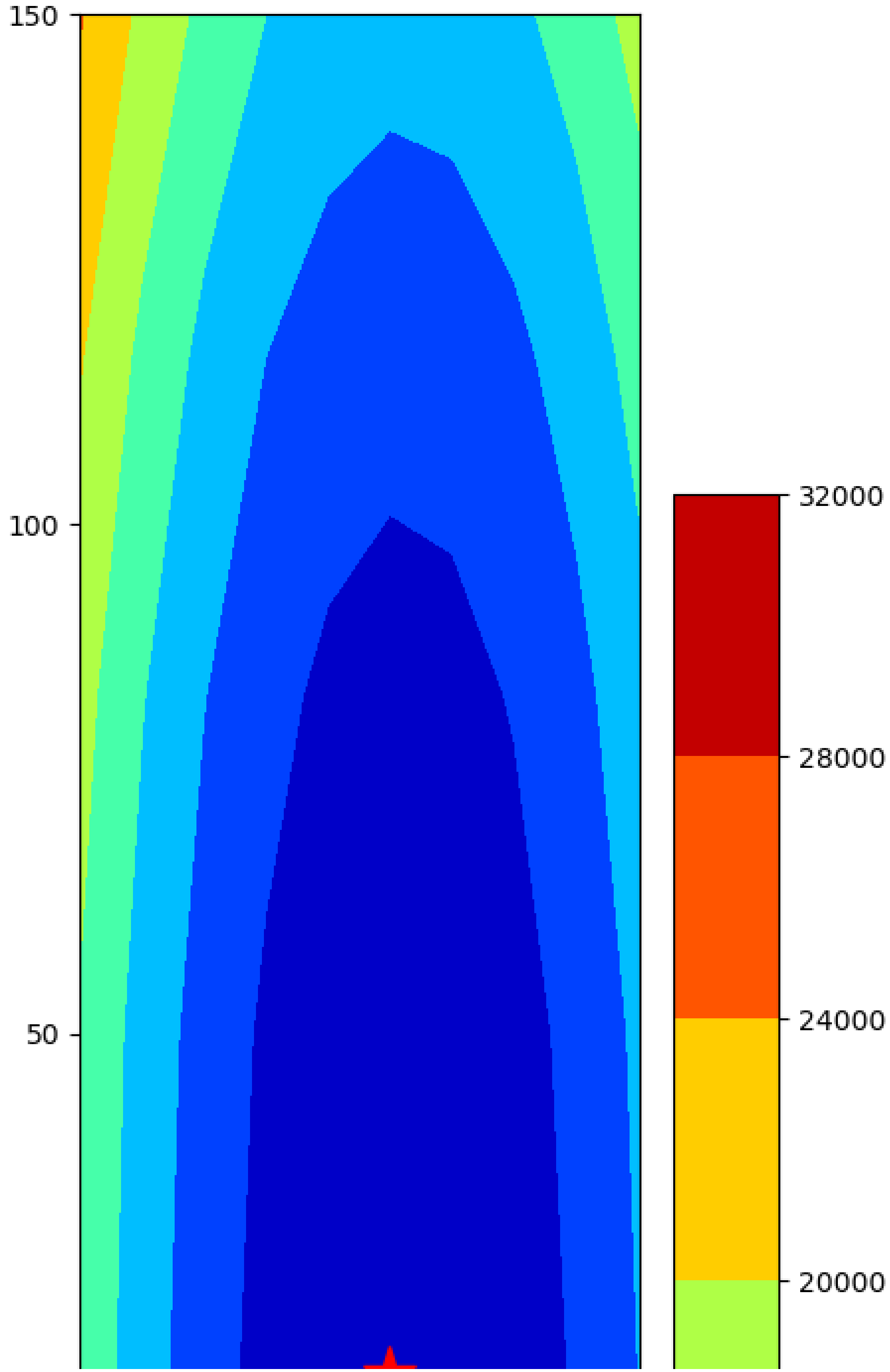
1e13
0.0
−0.2
−0.4
−0.6
−0.8
−1.0
−1.2
−1.4
w_1
−6
−4
−2
0
w_0
1e13
32000
28000
24000
20000
16000
12000
8000
4000
0

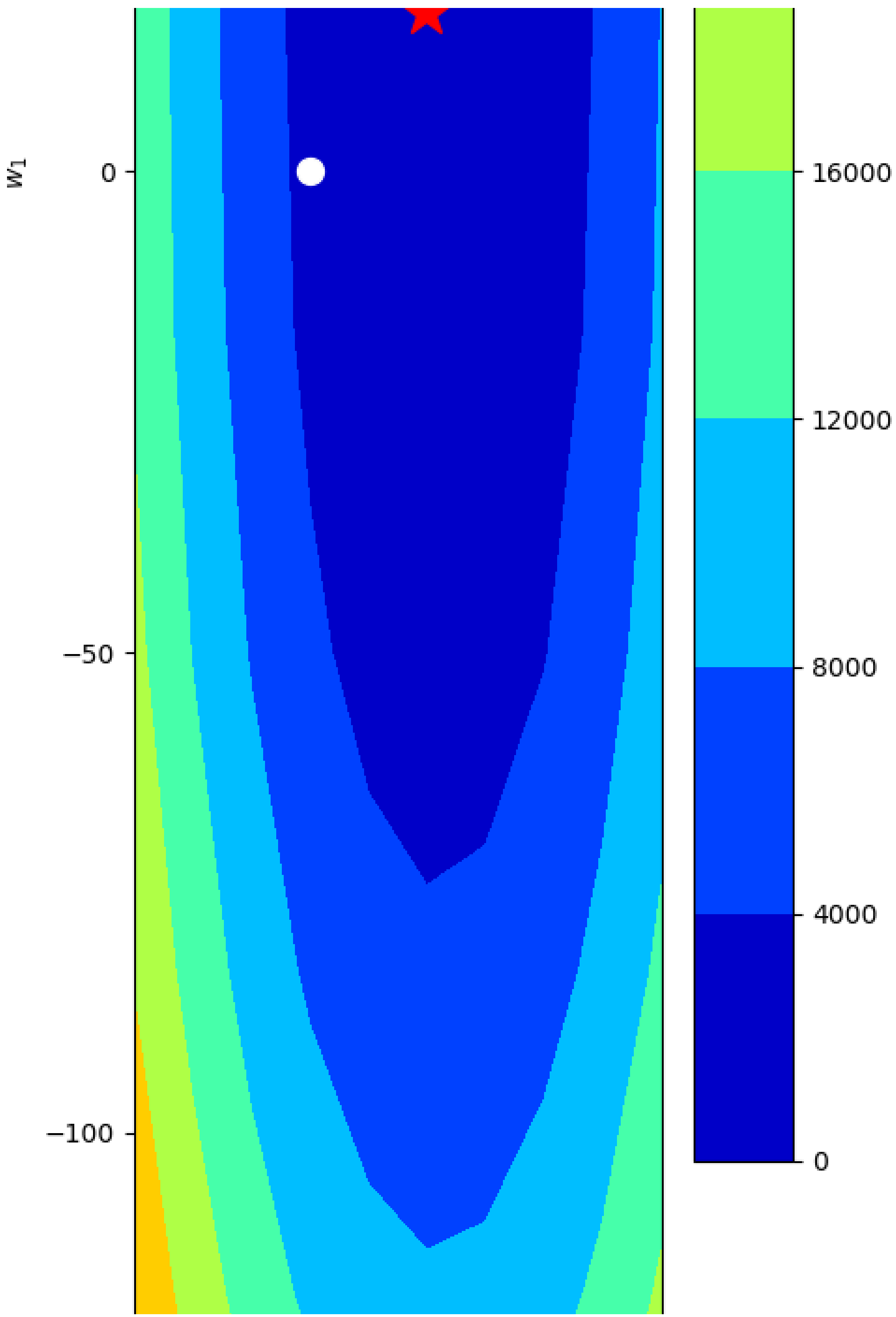


1e10
1.4
1.2
1.0
0.8
0.6
0.4
0.2
0.0
w_1
0
2
4
6
w_0
1e10
32000
28000
24000
20000
16000
12000
8000
4000
0

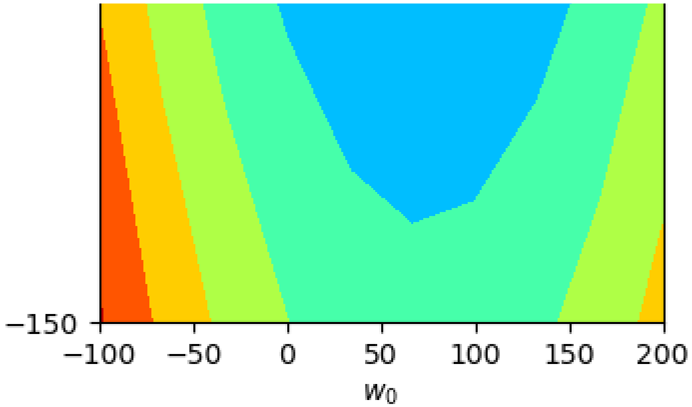


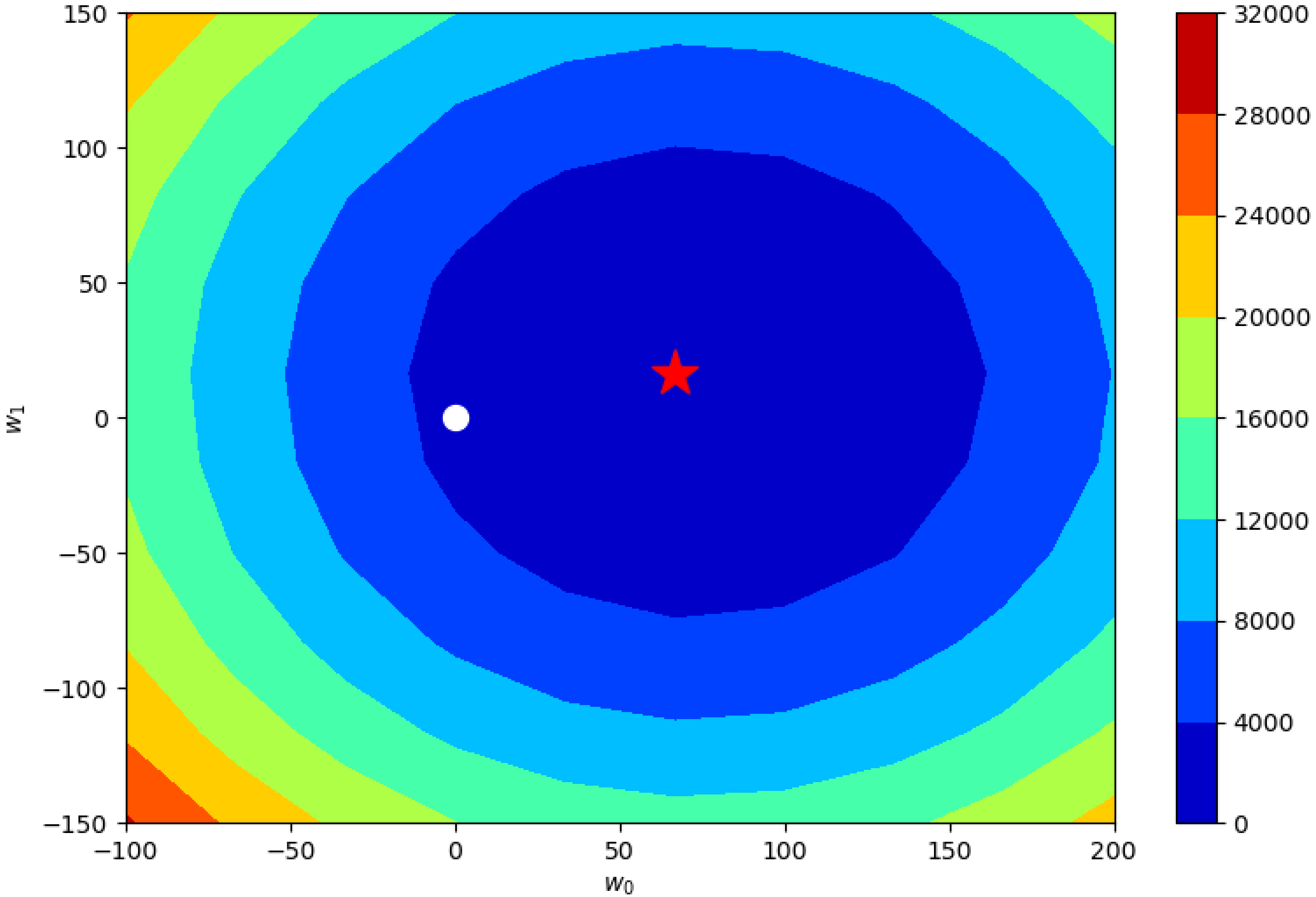
150
100
50
0
−50
−100
−150
w_1
−100
−50
0
50
100
150
200
w_0
32000
28000
24000
20000
16000
12000
8000
4000
0


150
100
50
32000
28000
24000
20000

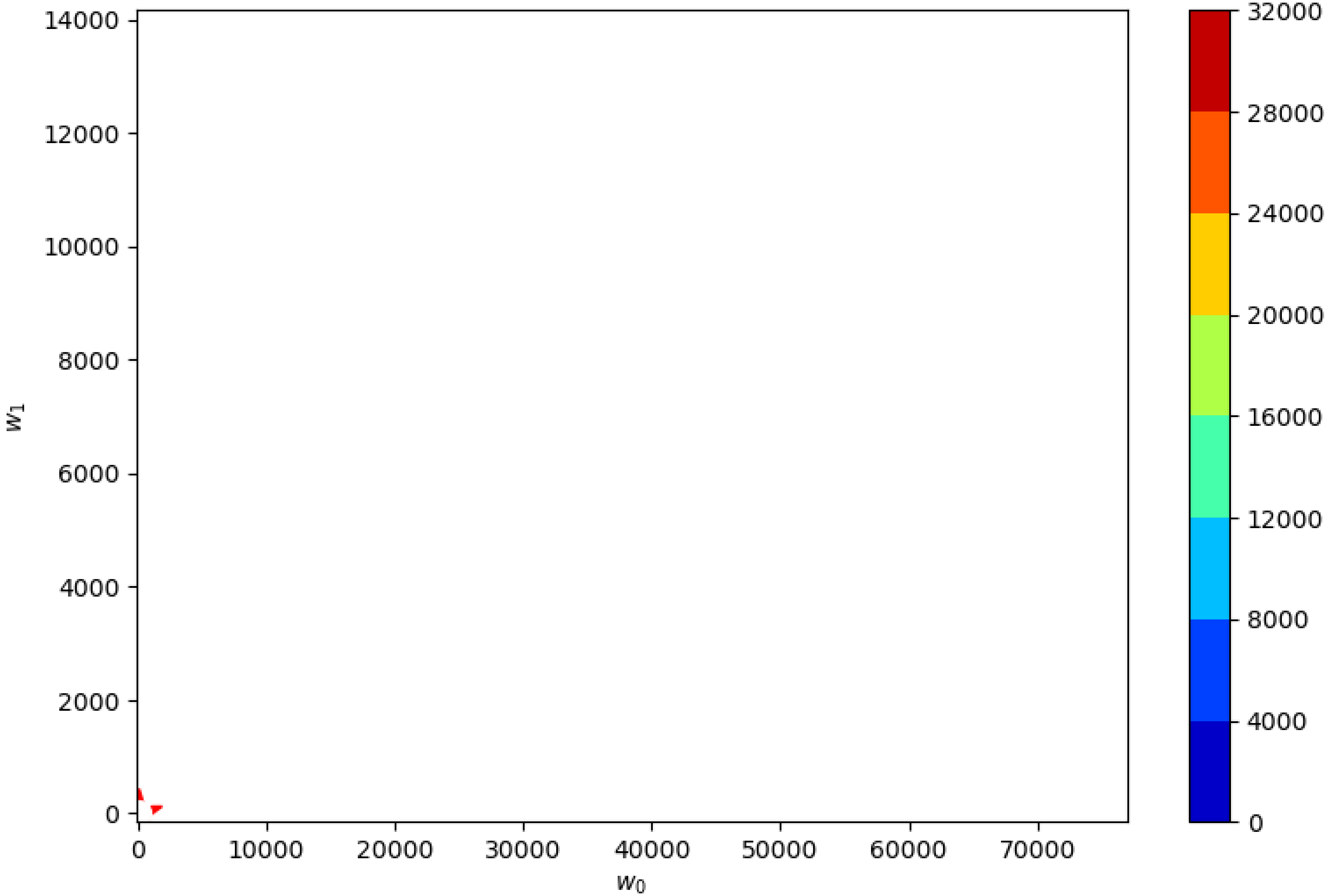


w_1
0
−50
−100
16000
12000
8000
4000
0

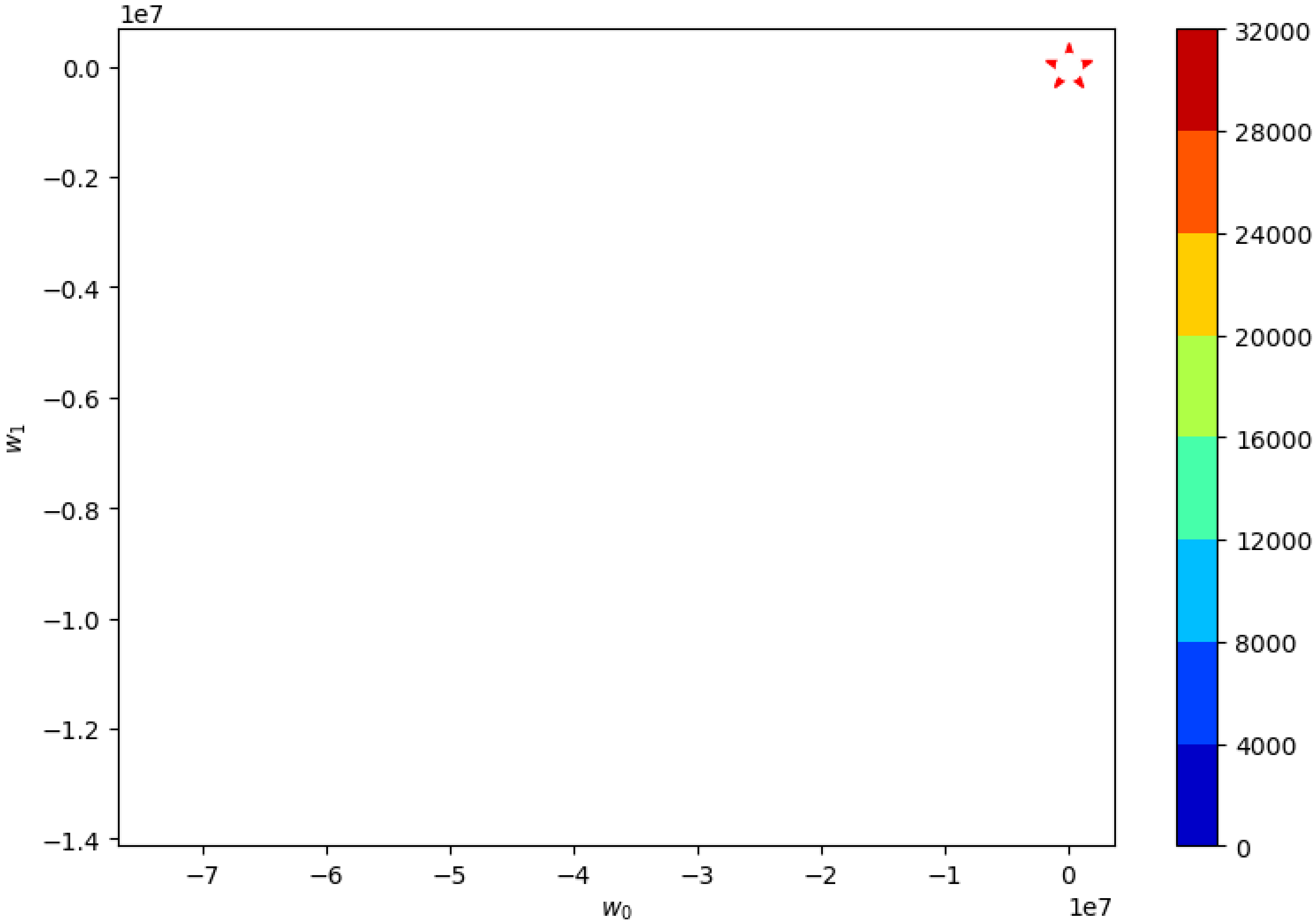

−150
−100
−50
0
50
100
150
200
w_0



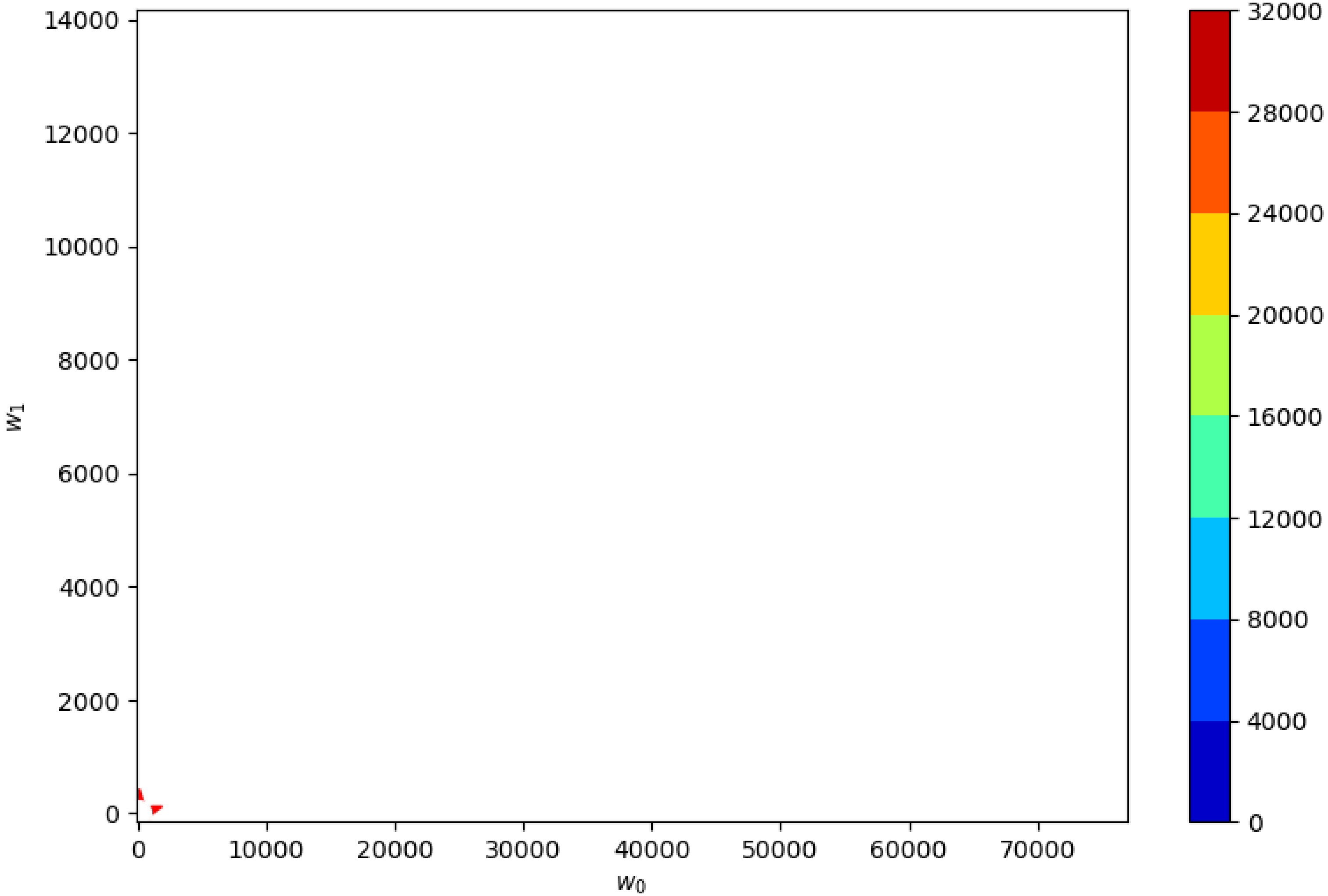
150
100
50
0
−50
−100
−150
−100
−50
0
50
100
150
200
w_0
w_1
32000
28000
24000
20000
16000
12000
8000
4000
0



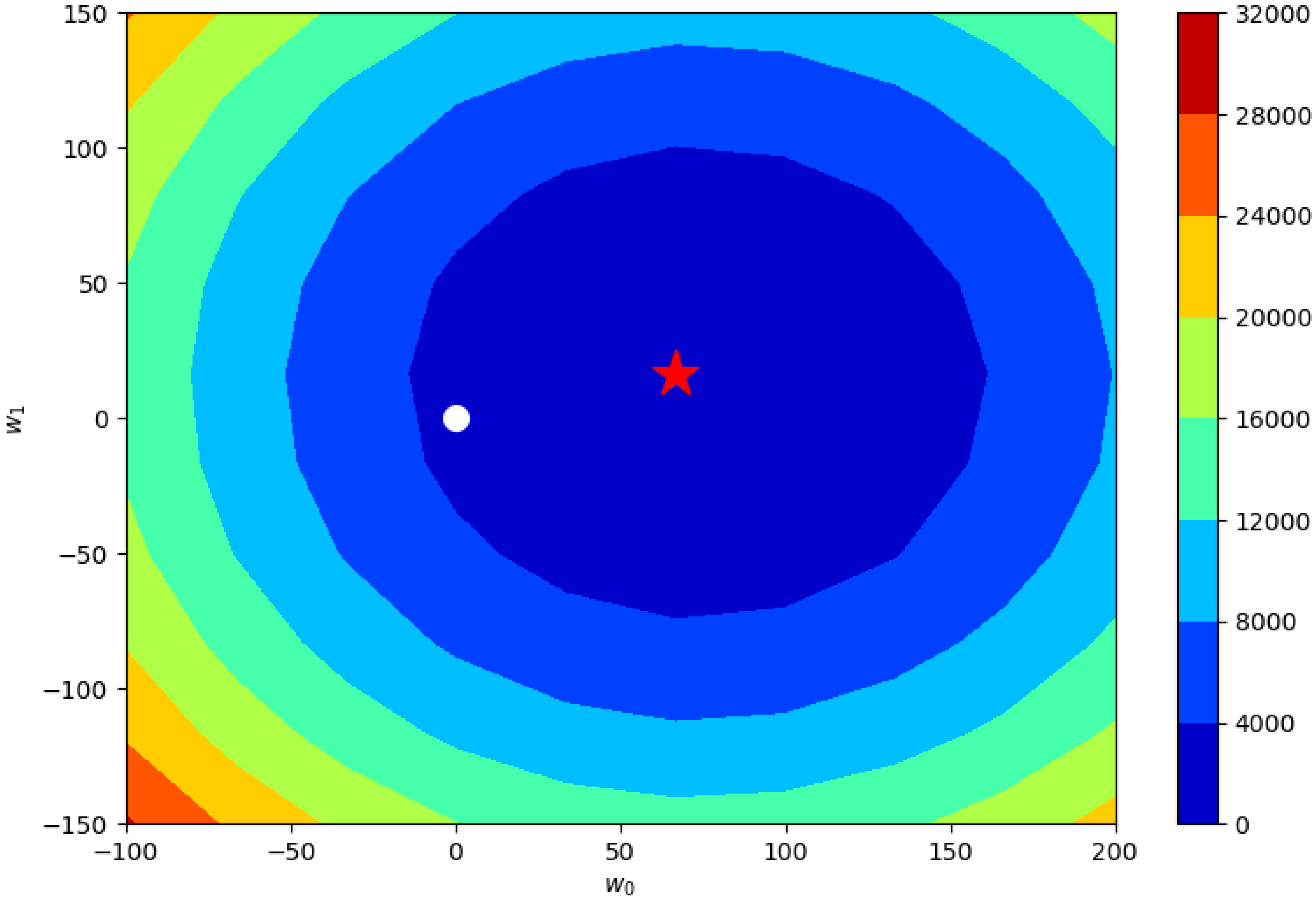
14000
12000
10000
8000
6000
4000
2000
0
w_1
0
10000
20000
30000
40000
50000
60000
70000
w_0
32000
28000
24000
20000
16000
12000
8000
4000
0



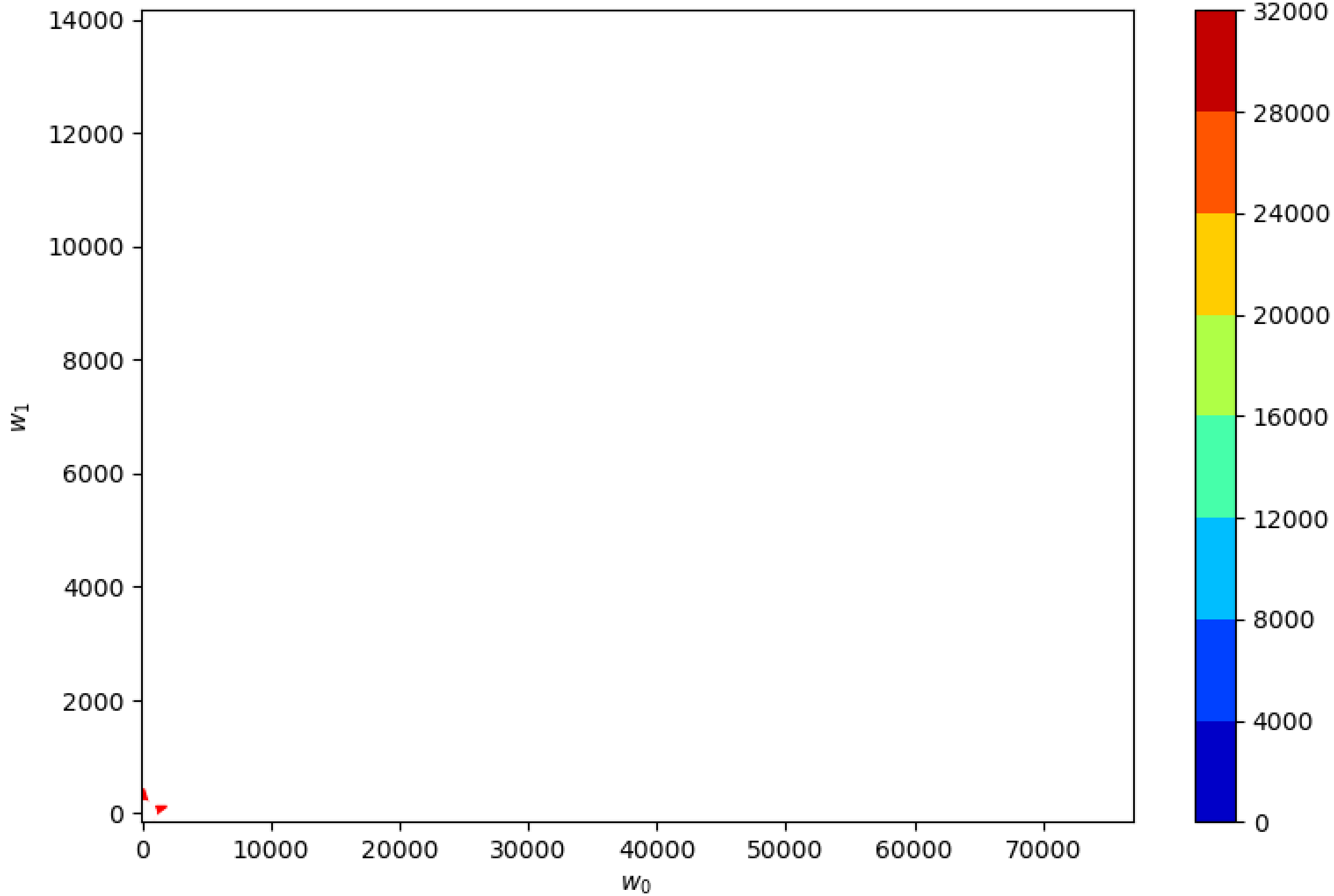
1e7
0.0
−0.2
−0.4
−0.6
−0.8
−1.0
−1.2
−1.4
w_1
−7
−6
−5
−4
−3
−2
−1
0
1e7
w_0
32000
28000
24000
20000
16000
12000
8000
4000
0



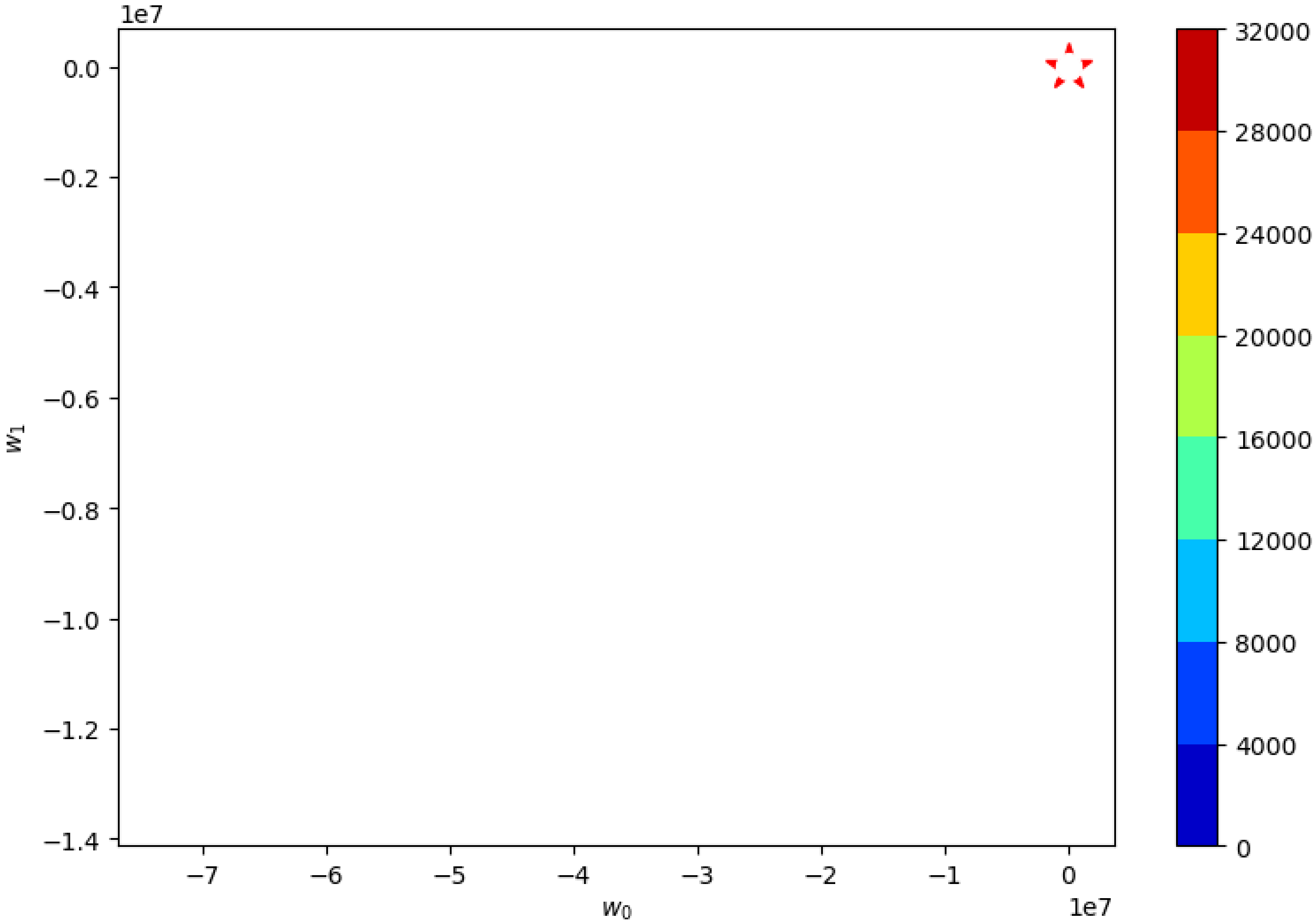
14000
12000
10000
8000
6000
4000
2000
0
w_1
0
10000
20000
30000
40000
50000
60000
70000
w_0
32000
28000
24000
20000
16000
12000
8000
4000
0

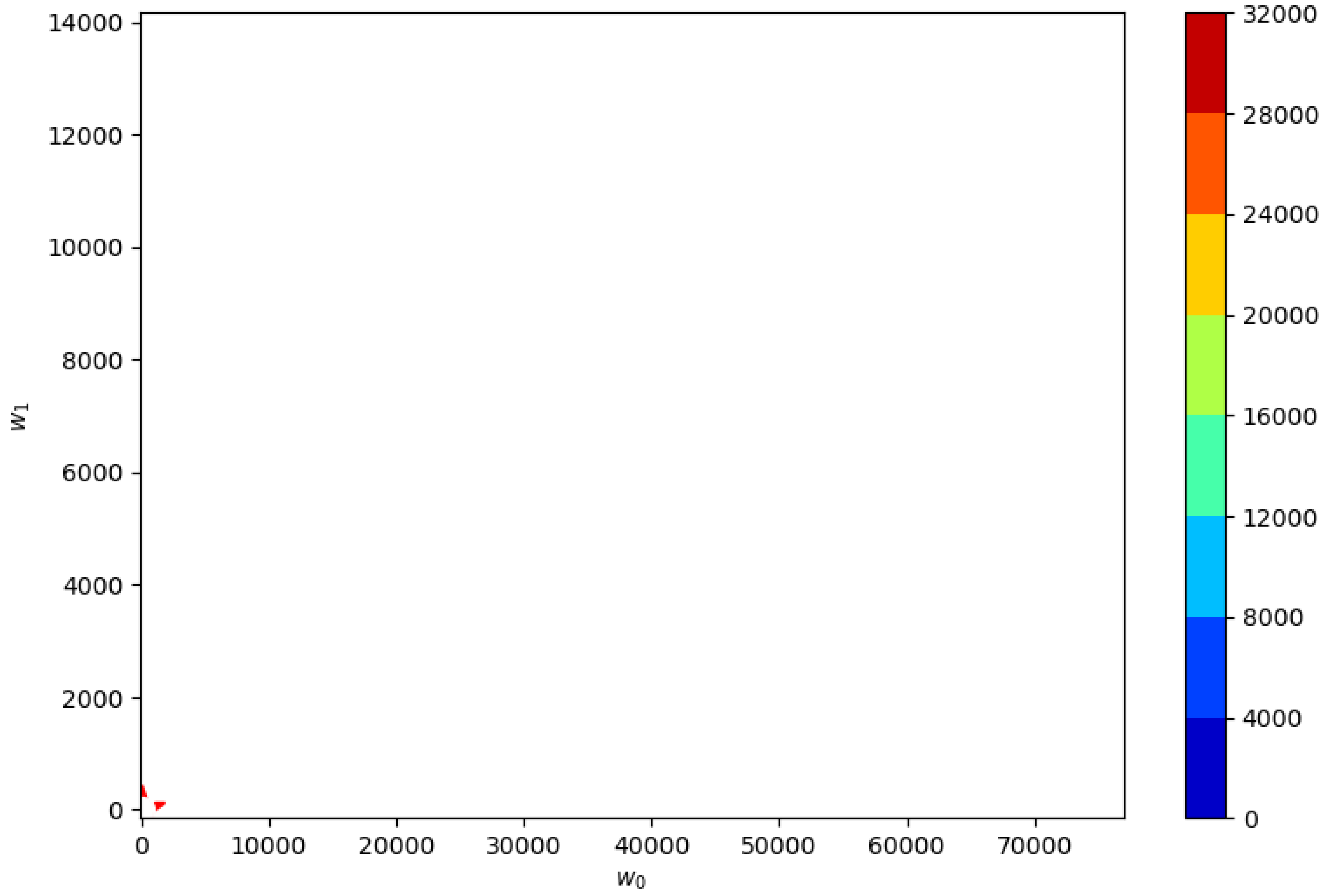


150
100
50
0
−50
−100
−150
w_1
−100
−50
0
50
100
150
200
w_0
32000
28000
24000
20000
16000
12000
8000
4000
0

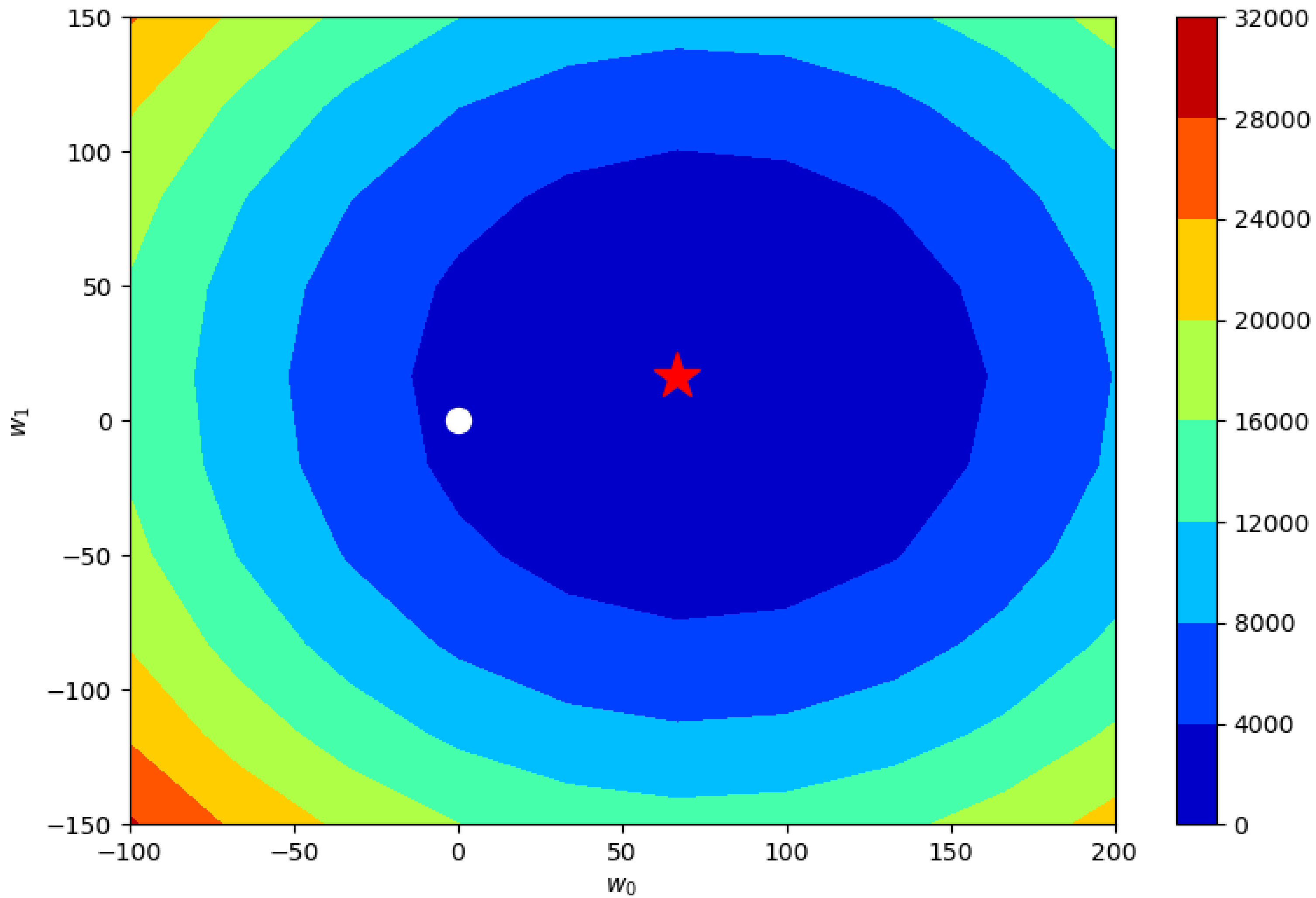


14000
12000
10000
8000
6000
4000
2000
0
w_1
0
10000
20000
30000
40000
50000
60000
70000
w_0
32000
28000
24000
20000
16000
12000
8000
4000
0

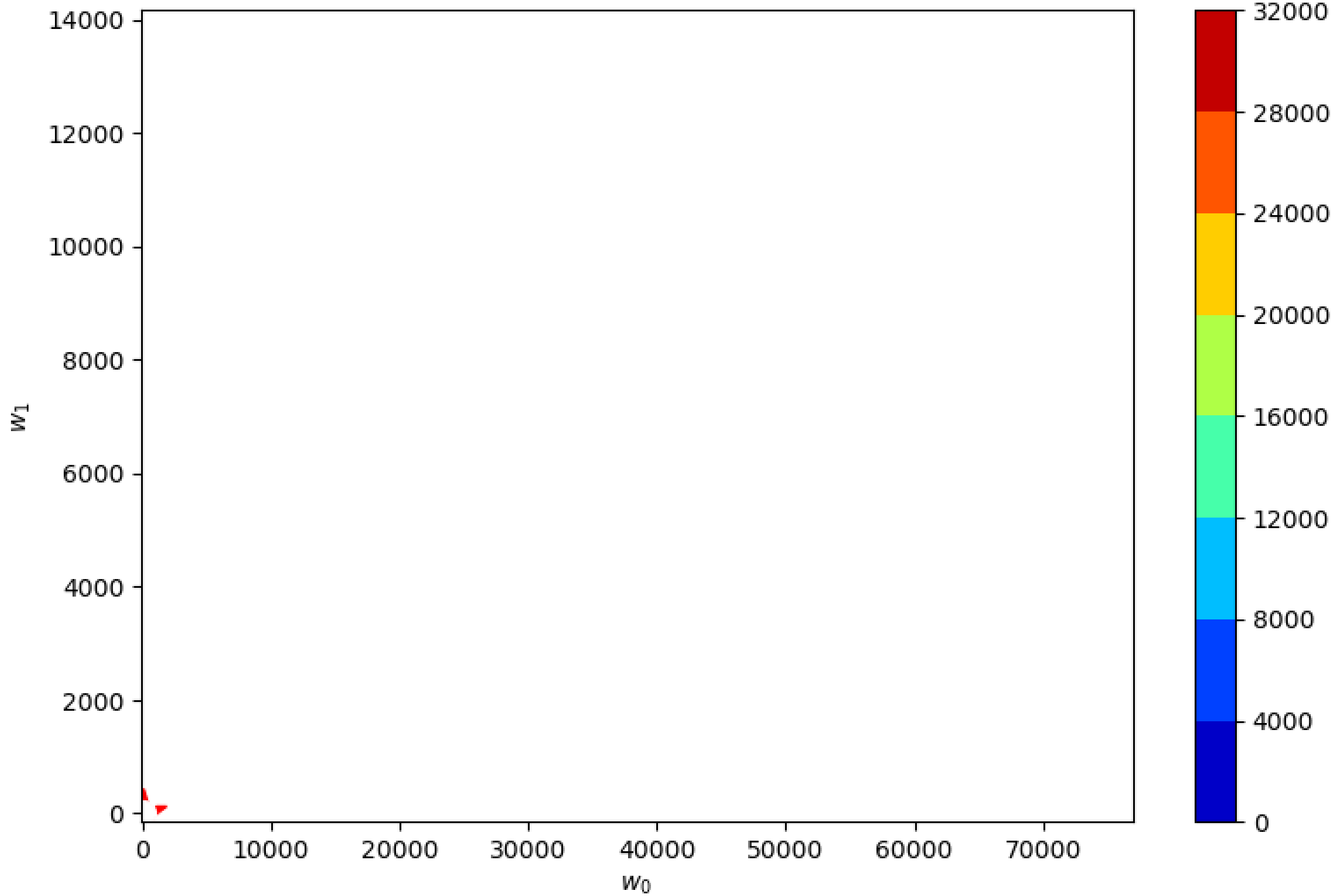

1e7
0.0
−0.2
−0.4
−0.6
−0.8
−1.0
−1.2
−1.4
w_1
−7
−6
−5
−4
−3
−2
−1
0
1e7
w_0
32000
28000
24000
20000
16000
12000
8000
4000
0

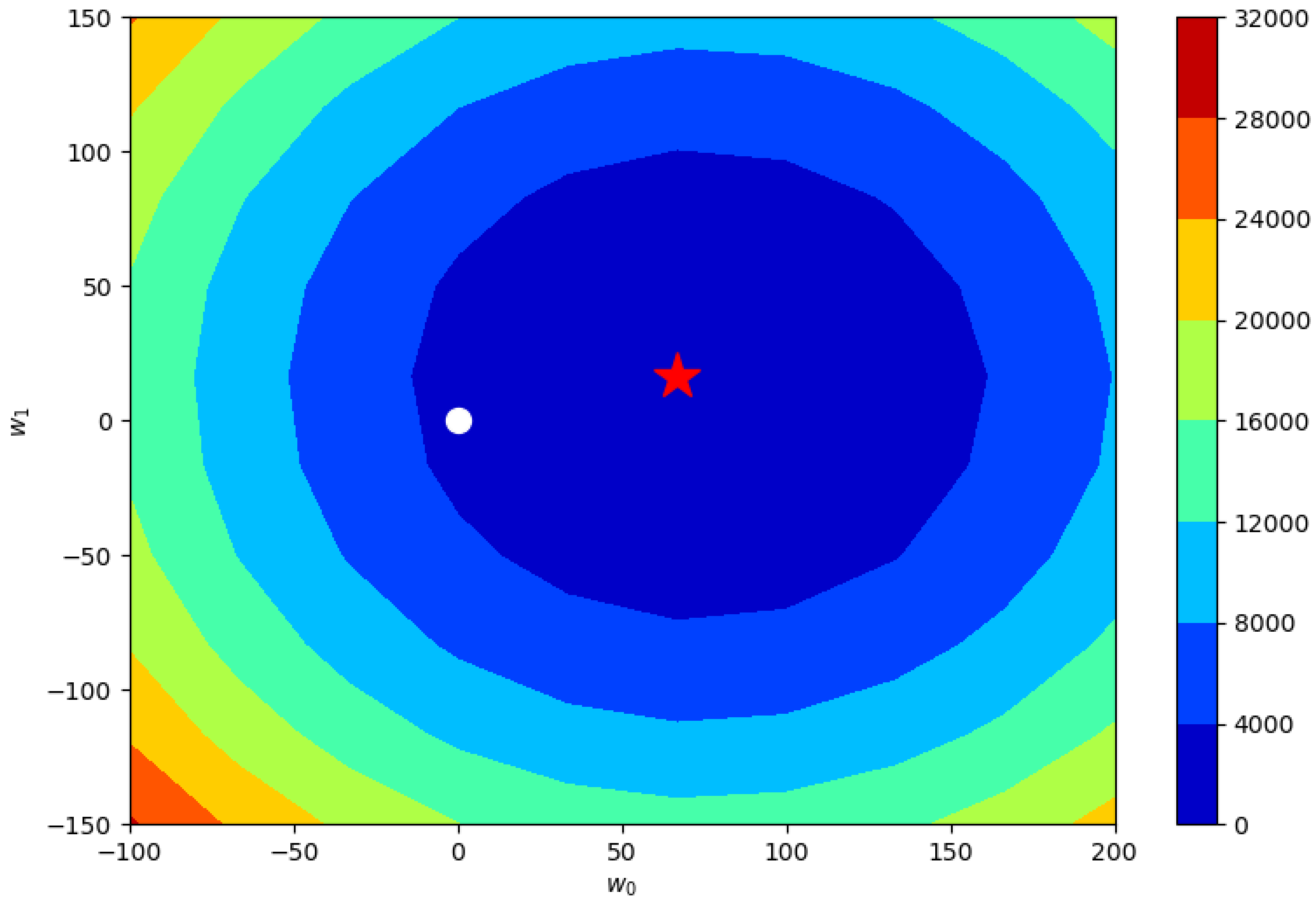


14000
12000
10000
8000
6000
4000
2000
0
w_1
0
10000
20000
30000
40000
50000
60000
70000
w_0
32000
28000
24000
20000
16000
12000
8000
4000
0

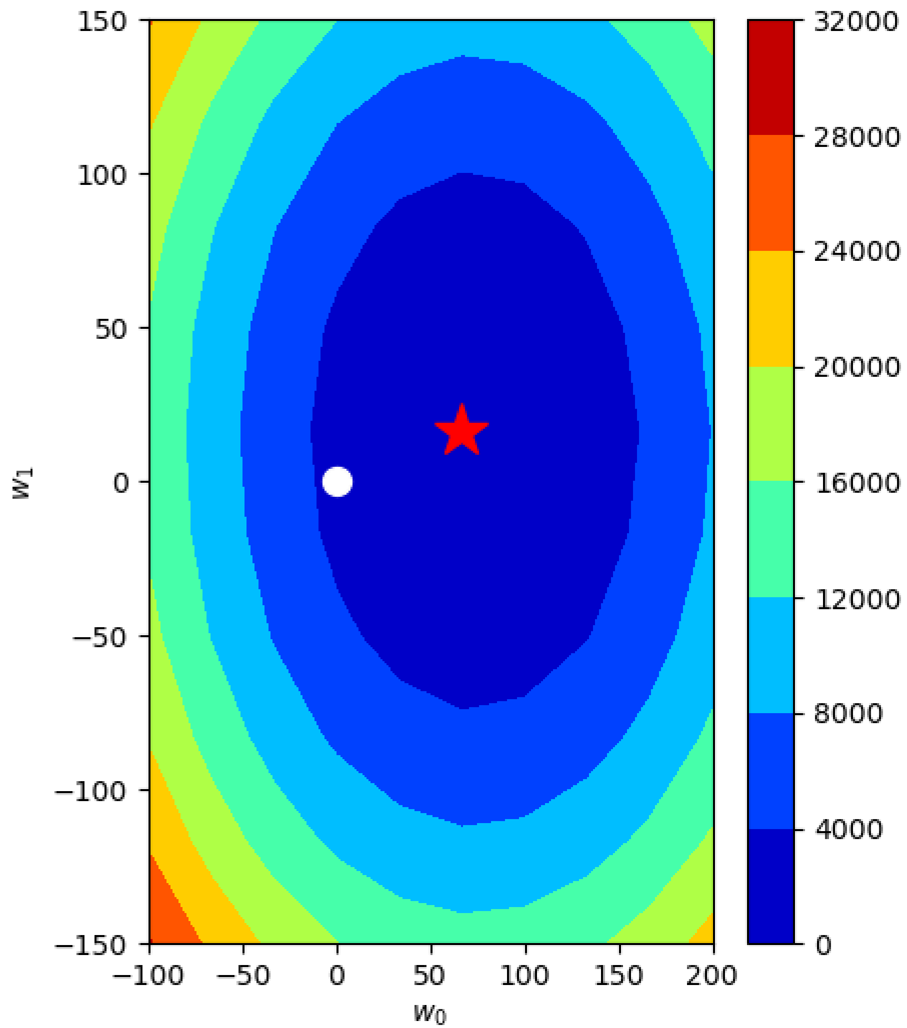


150
100
50
0
−50
−100
−150
w_1
−100
−50
0
50
100
150
200
w_0
32000
28000
24000
20000
16000
12000
8000
4000
0

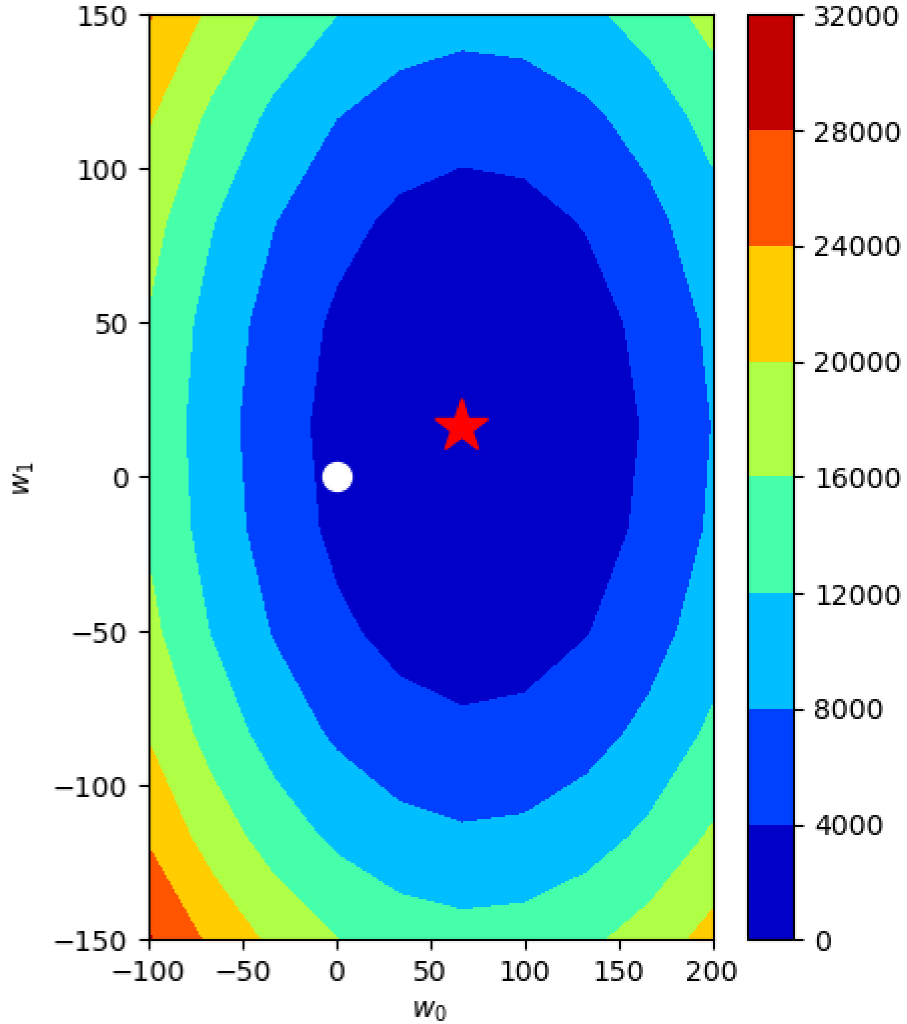

14000
12000
10000
8000
6000
4000
2000
0
w_1
0
10000
20000
30000
40000
50000
60000
70000
w_0
32000
28000
24000
20000
16000
12000
8000
4000
0

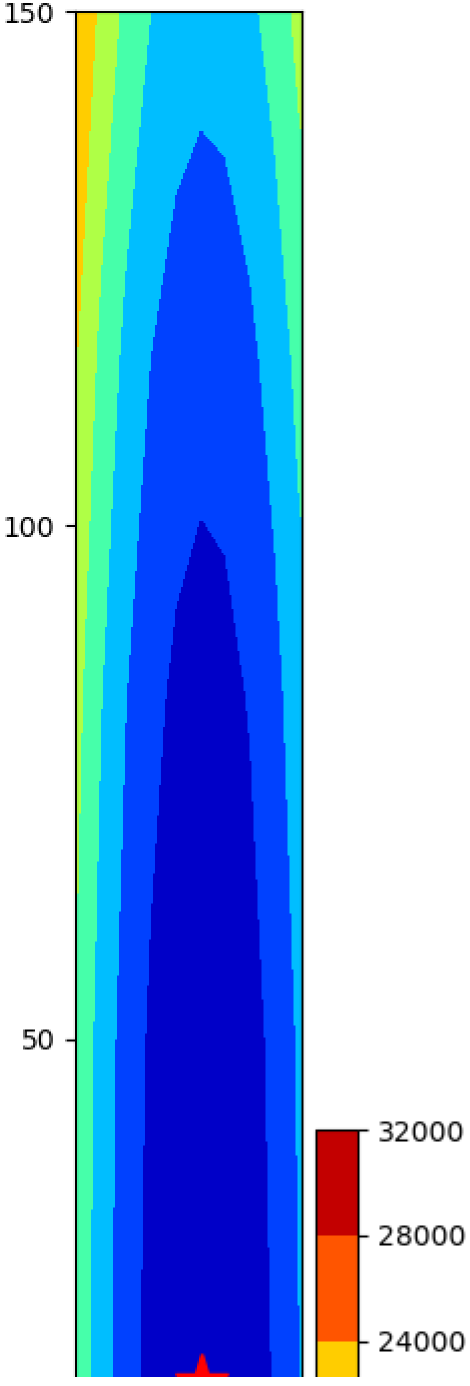


150
100
50
0
−50
−100
−150
w_1
−100
−50
0
50
100
150
200
w_0
32000
28000
24000
20000
16000
12000
8000
4000
0

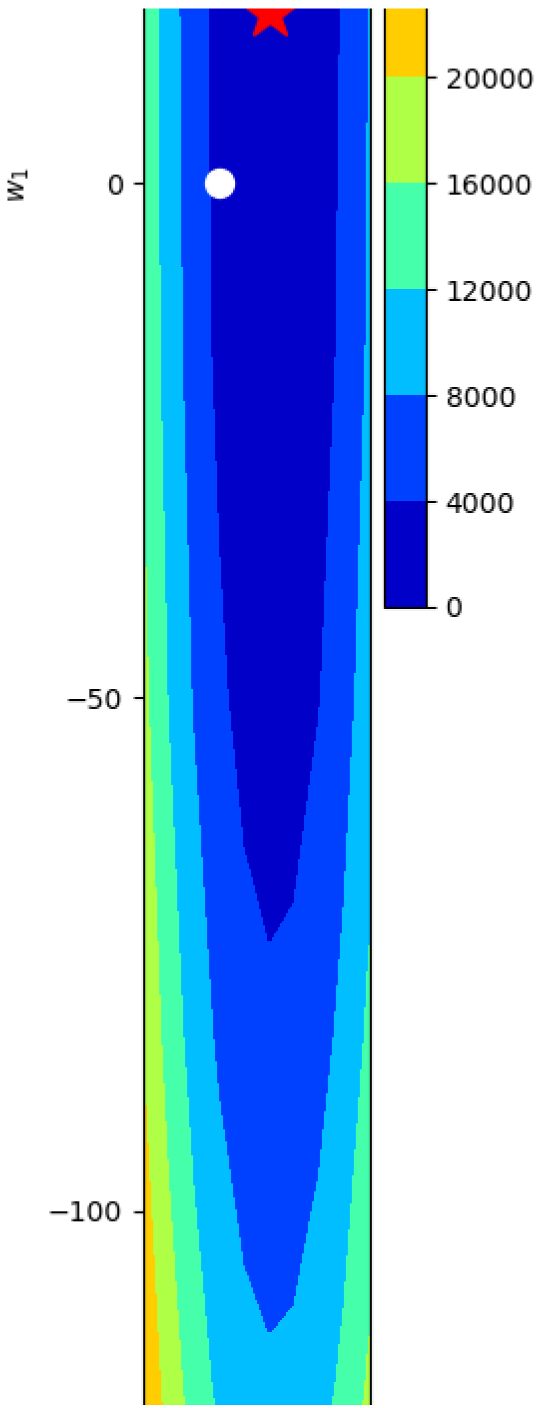


150
100
50
0
−50
−100
−150
w_1
−100
−50
0
50
100
150
200
w_0
32000
28000
24000
20000
16000
12000
8000
4000
0

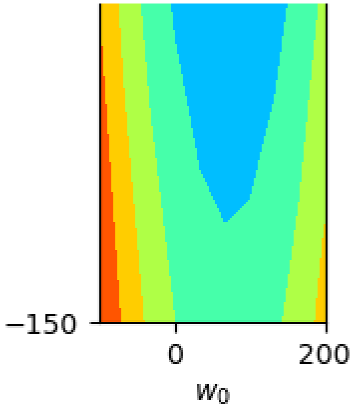

150
100
50
0
−50
−100
−150
w_1
−100
−50
0
50
100
150
200
w_0
32000
28000
24000
20000
16000
12000
8000
4000
0



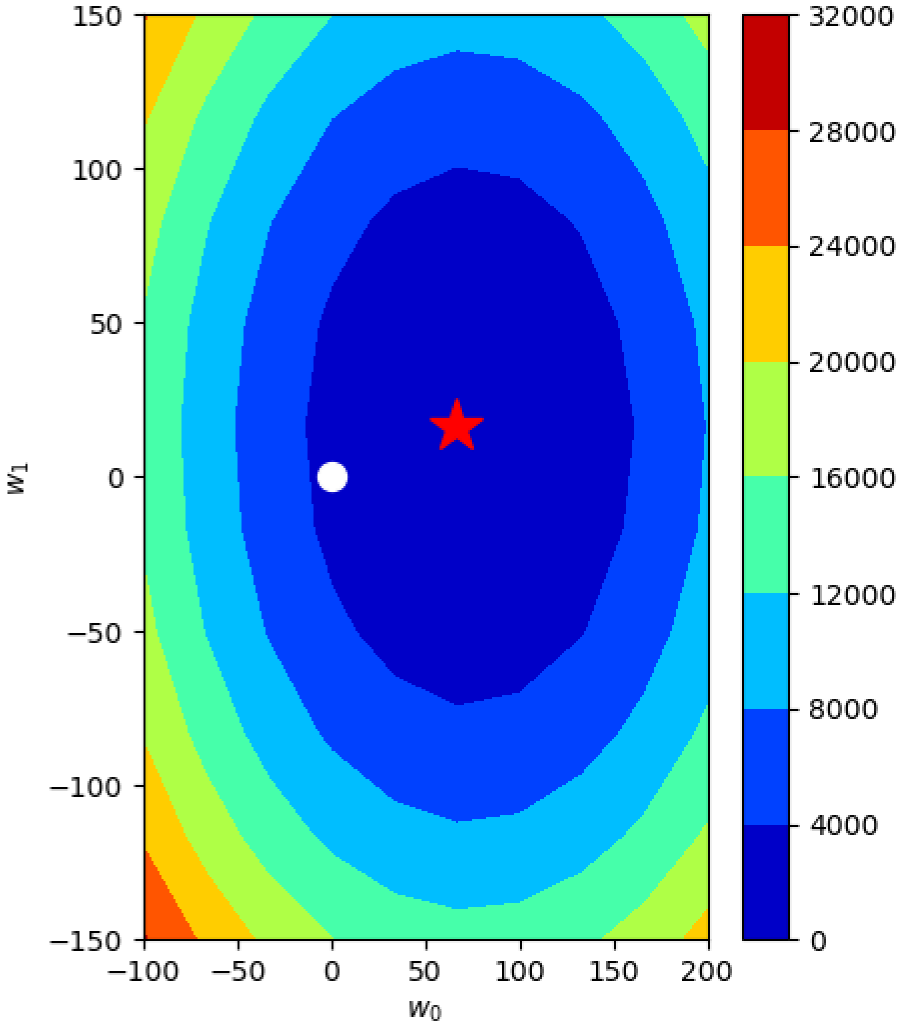
150
100
50
32000
28000
24000



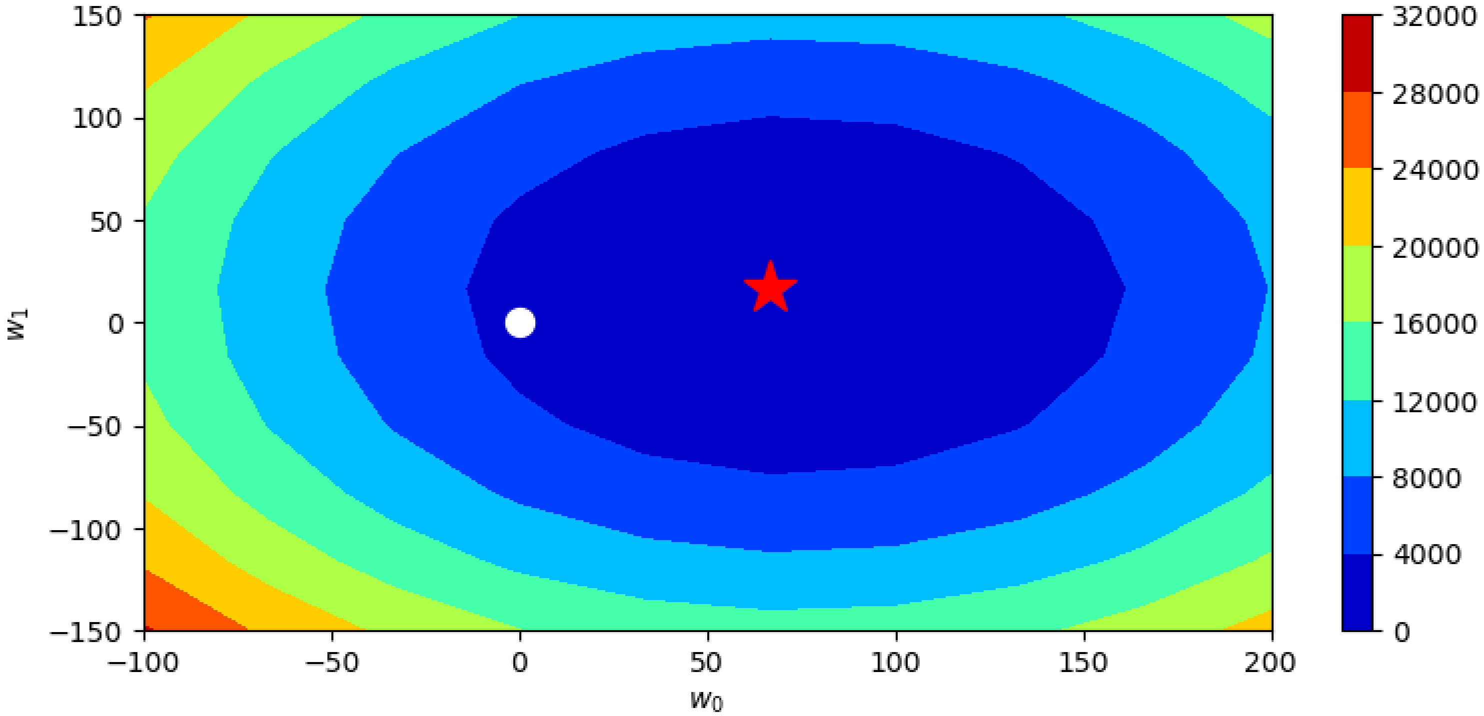
w_1
0
−50
−100
20000
16000
12000
8000
4000
0

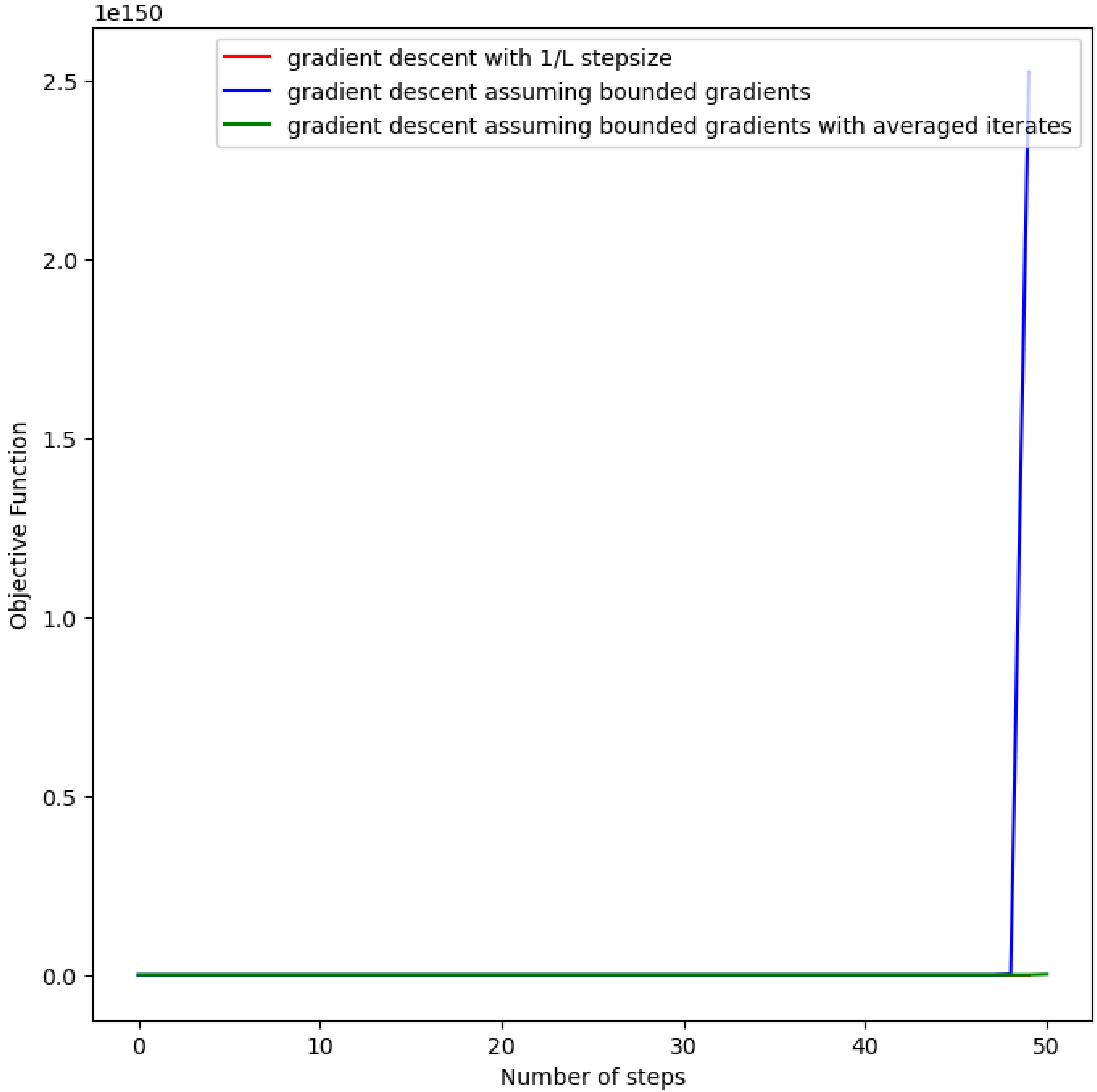


−150
0
200
w_0



150
100
50
0
−50
−100
−150
w_1
−100
−50
0
50
100
150
200
w_0
32000
28000
24000
20000
16000
12000
8000
4000
0


150
100
50
0
−50
−100
−150
w_1
−100
−50
0
50
100
150
200
w_0
32000
28000
24000
20000
16000
12000
8000
4000
0



1e150
2.5
2.0
1.5
1.0
0.5
0.0
Objective Function
0
10
20
30
40
50
Number of steps
gradient descent with 1/L stepsize
gradient descent assuming bounded gradients
gradient descent assuming bounded gradients with averaged iterates

In [ ]: