Assignment 1

Machine Learning 1, SS24

Team Members		
Last name	First name	Matriculation Number
Farfan Roca	Javier	12130995
Omanovic	Melvis	1600002

1 Linear Regression – Detection of memristor faults

1.
$$E(\theta) = \frac{1}{m}\sum_{i=1}^{m}\left(h_\theta(\Delta R_i^{ideal}) - \Delta R_i\right)^2$$

$$\theta^* = \arg\min_\theta E(\theta)$$

$$\frac{\partial E(\theta)}{\partial \theta} = \frac{1}{m}\sum_{i=1}^{m} 2\cdot(\theta\,\Delta R_i^{ideal} - \Delta R_i)\cdot \Delta R_i^{ideal}$$

$$\frac{1}{m}\sum_{i=1}^{m} 2\cdot(\theta^*\cdot\Delta R_i^{ideal} - \Delta R_i)\cdot \Delta R_i^{ideal} = 0 \qquad /\cdot \frac{m}{2}$$

$$\theta^*\cdot\sum_{i=1}^{m}(\Delta R_i^{ideal})^2 - \sum_{i=1}^{m}\Delta R_i\cdot\Delta R_i^{ideal} = 0$$

$$\theta^*\cdot\sum_{i=1}^{m}(\Delta R_i^{ideal})^2 = \sum_{i=1}^{m}\Delta R_i\cdot\Delta R_i^{ideal} \qquad /: \sum_{i=1}^{m}(\Delta R_i^{ideal})^2$$

$$\theta^* = \frac{\sum_{i=1}^{m}\Delta R_i\cdot\Delta R_i^{ideal}}{\sum_{i=1}^{m}(\Delta R_i^{ideal})^2}$$

2.
$$E(\theta_0,\theta_1) = \frac{1}{m}\sum_{i=1}^{m}\left(h_{\theta_0,\theta_1}(\Delta R_i^{ideal}) - \Delta R_i\right)^2,$$

$$(\theta_0^*, \theta_1^*) = \arg\min_{\theta_0,\theta_1} E(\theta_0,\theta_1)$$

$$E(\theta_0,\theta_1) = \frac{1}{m}\sum_{i=1}^{m}\left(\theta_0 + \theta_1\cdot\Delta R^{ideal} - \Delta R_i\right)^2$$

$$\frac{\partial E(\theta_0,\theta_1)}{\partial \theta_0} = \frac{1}{m}\sum_{i=1}^{m} 2\cdot(\theta_0 + \theta_1\cdot\Delta R_i^{ideal} - \Delta R_i)\cdot 1$$

$$= \frac{2}{m}\sum_{i=1}^{m}\left(h_{\theta_0,\theta_1}(\Delta R_i^{ideal}) - \Delta R_i\right)$$

$$\frac{\partial E(\theta_0,\theta_1)}{\partial \theta_1} = \frac{1}{m}\sum_{i=1}^{m} 2\cdot\left(\theta_0 + \theta_1\cdot\Delta R_i^{ideal} - \Delta R_i\right)\cdot \Delta R_i^{ideal}$$

$$= \frac{2}{m}\sum_{i=1}^{m}\left(h_{\theta_0,\theta_1}(\Delta R_i^{ideal}) - \Delta R_i\right)\cdot\Delta R_i^{ideal}$$

$$\frac{2}{m}\sum_{i=1}^{m}\left(\Theta_0^* + \Theta_1^*\cdot\Delta R_i^{ideal} - \Delta R_i\right) = 0$$

$$m\cdot\Theta_0^* + \Theta_1^*\sum_{i=1}^{m}\cdot\Delta R_i^{ideal} - \sum_{i=1}^{m}\Delta R_i = 0$$

$$m\cdot\Theta_0^* = \sum_{i=1}^{m}\Delta R_i - \Theta_1^*\sum_{i=1}^{m}\Delta R_i^{ideal}$$

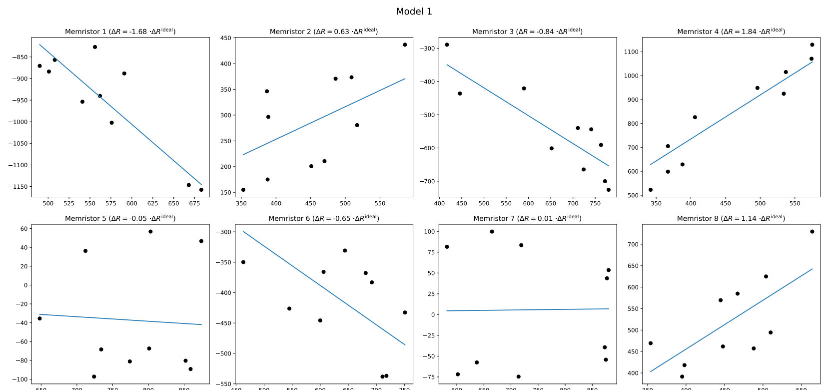
$$\Theta_0^* = \frac{\sum_{i=1}^{m}\Delta R_i - \Theta_1^*\cdot\Delta R_i^{ideal}}{m}$$

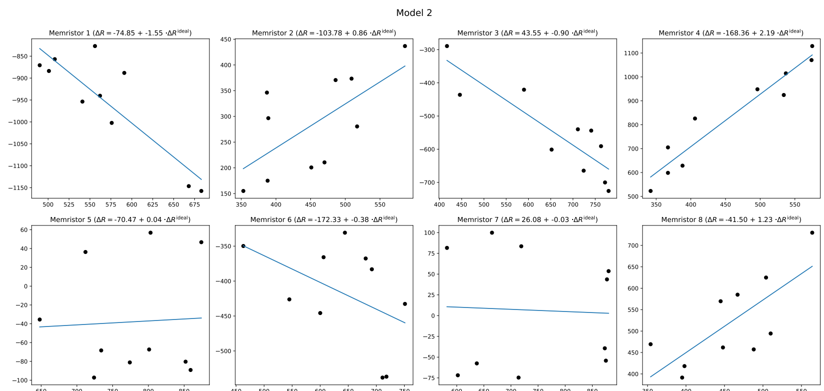
$$\frac{2}{m}\sum_{i=1}^{m}\left(\Theta_0^* + \Theta_1^*\cdot\Delta R_i^{ideal} - \Delta R_i\right)\cdot\Delta R_i^{ideal} = 0$$

$$\Theta_0^*\cdot\sum_{i=1}^{m}\Delta R_i^{ideal} + \Theta_1^*\cdot\sum_{i=1}^{m}\left(\Delta R_i^{ideal}\right)^2 - \sum_{i=1}^{m}\Delta R_i\cdot\Delta R_i^{ideal} = 0$$

$$\Theta_0^*\cdot\sum_{i=1}^{m}\Delta R_i^{ideal} = \sum_{i=1}^{m}\Delta R_i\cdot\Delta R_i^{ideal} - \Theta_1^*\cdot\sum_{i=1}^{m}\left(\Delta R_i^{ideal}\right)^2$$

$$\Theta_0^* = \frac{\sum_{i=1}^{m}\Delta R_i\,\Delta R_i^{ideal} - \Theta_1^*\cdot\sum_{i=1}^{m}\left(\Delta R_i^{ideal}\right)^2}{\sum_{i=1}^{m}\Delta R_i^{ideal}}$$





Model 1:
The parameter θ denotes the slope of the memristors. Negative values represent a negative slope (discordant), values close to 0 are stuck and positive values are more indicative of concordant faults.
Model 2:
In principle similar to model 1 but we now have the intercept. We want this to be close to 0 in order to classify something as ideal.

I would choose model 1 because it is simpler and I am content with the results it delivered.
Memristors were classified with the following code:

```
if(theta > 2.0):
    return MemristorFault.IDEAL
if(theta >0.1):
    return MemristorFault.CONCORDANT
if (theta < -0.1):
    return MemristorFault.DISCORDANT
else:
    return MemristorFault.STUCK
```

The memristors were classified this way:
Memristor 1 is classified as discordant.
Memristor 2 is classified as concordant.
Memristor 3 is classified as discordant.
Memristor 4 is classified as concordant.
Memristor 5 is classified as stuck.
Memristor 6 is classified as discordant.
Memristor 7 is classified as stuck.
Memristor 8 is classified as concordant.

2 Logistic Regression

The following features were added to the design matrices on top of x1 and x2.

```
center_x = (X_data[:, 0].max() + X_data[:, 0].min()) / 2 #X_data[:, 0] = x1, X_data[:, 1] = x2
center_y = (X_data[:, 1].max() + X_data[:, 1].min()) / 2  #necessary for feature

feature = np.sqrt((X_data[:, 0] - center_x)**2 + (X_data[:, 1] - center_y)**2)
feature1 = X_data[:, 0] * X_data[:, 1]
feature2 = X_data[:, 0] + X_data[:, 1]
```

Features for data set 1

```
X = np.hstack((X_data, X_data ** 2))
```

Features for data set 2

```
sin_x1 = np.sin(x1)
sin_x2 = np.sin(x2)
cos_x1 = np.cos(x1)
cos_x2 = np.cos(x2)
sin_cos_x1 = np.sin(x1) * np.cos(x1)
x1_cubed = x1**3
extra = np.sin(x1)**2 - np.sin(x2)
extra2 = np.sin(x2)**2 - np.sin(x1)
extra3 = np.cos(x1)**2 - np.cos(x2)
extra4 = np.cos(x2)**2 - np.cos(x1)
extra5 = np.sin(x1) + np.cos(x1)
extra6 = np.sin((np.pi * x1 )/2)
```

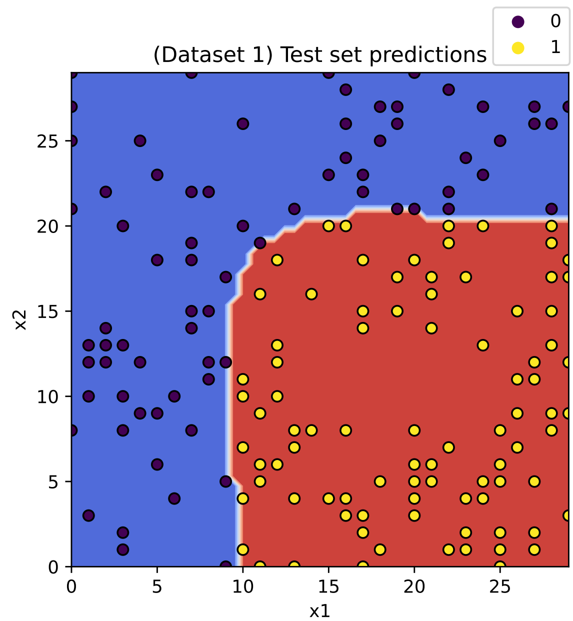
Features for data set 3

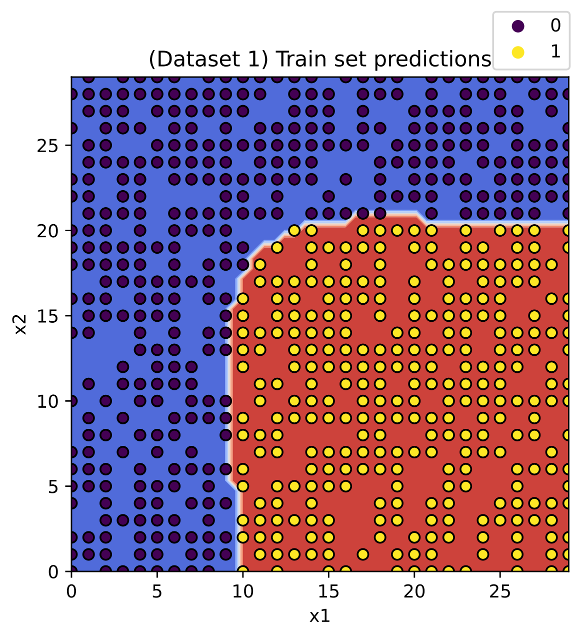
The l2 penalty was used. The following results were achieved:

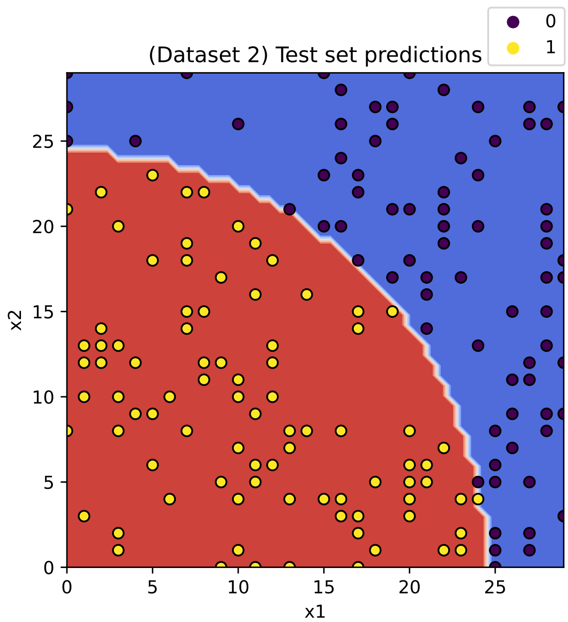
```
---- Logistic regression task 1 ----
Shapes of: X_train (640, 5), X_test (160, 5), y_train (640,), y_test (160,)
Train accuracy: 99.69%. Test accuracy: 98.75%.
Train loss: 0.022944184335746486. Test loss: 0.046497633893654675.
Parameters: [[ 2.70297543 -0.51206517 -0.76460317 -0.21709788  2.19091026]], [-36.32143224]
---- Logistic regression task 2 ----
Shapes of: X_train (640, 4), X_test (160, 4), y_train (640,), y_test (160,)
Train accuracy: 100.00%. Test accuracy: 100.00%.
Train loss: 0.0005398954451803412. Test loss: 0.0038387463242637856.
Parameters: [[ 0.21879489  0.0849633  -0.66633497 -0.66540454]], [394.5933908]
---- Logistic regression task 3 ----
Shapes of: X_train (568, 14), X_test (143, 14), y_train (568,), y_test (143,)
Train accuracy: 92.61%. Test accuracy: 95.10%.
Train loss: 0.17866642615085537. Test loss: 0.1419526673240821.
Parameters: [[-0.5580755  -2.43599138 -0.44461853 -1.94160822 -0.61507153 -1.43984865
   1.9409876   1.28167116 -0.32654179  2.63899903  3.683186   -1.60412163
  -1.05969005  0.81839252]], [-1.51649281]
```

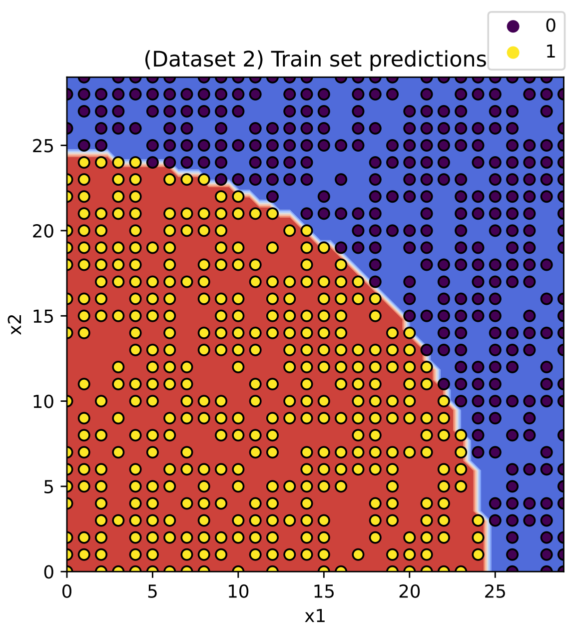
Features for data set 3

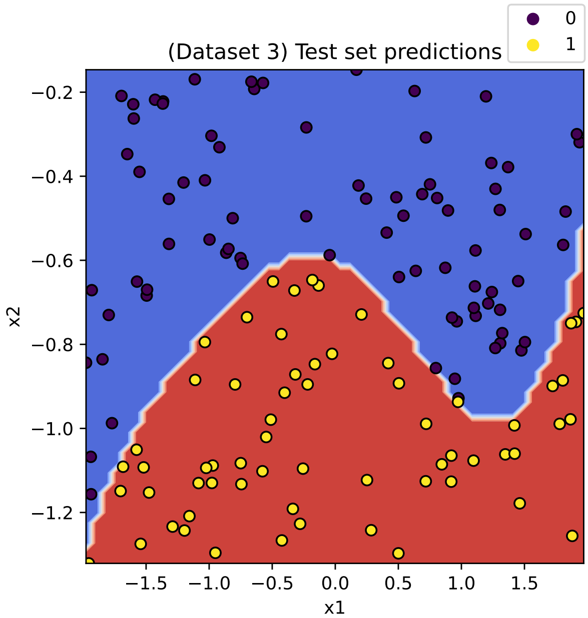
Resulting plots:

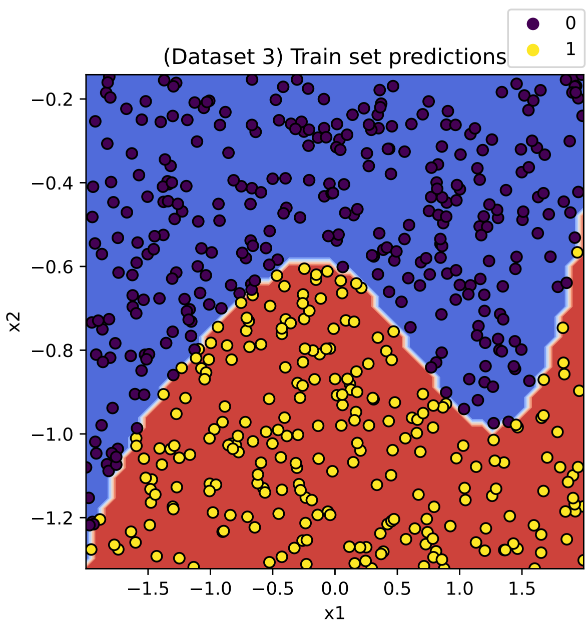





(Dataset 2) Test set predictions
0
1
x2
x1
0
5
10
15
20
25


(Dataset 2) Train set predictions
0
1
x2
x1
0
5
10
15
20
25





Classifier weights and bias for first data set:

`[[ 2.70297543 -0.51206517 -0.76460317 -0.21709788  2.19091026]], [-36.32143224]`

Classifier weights and bias for second data set:

```
[[ 0.21879489  0.0849633  -0.66633497 -0.66540454]], [394.5933908]
```

Classifier weights and bias for third data set:

```
[[-0.5580755  -2.43599138 -0.44461853 -1.94160822 -0.61507153 -1.43984865
   1.9409876   1.28167116 -0.32654179  2.63899903 3.683186   -1.60412163
   -1.05969005 0.81839252]], [-1.51649281]
```

Assume we have trained a logistic regression classifier (binary classes) and are given a test dataset D. Is the following statement correct? "If the classifier predicts the correct class for all elements in D (100 percent accuracy), then it follows that the cross-entropy loss (w.r.t. D) is 0." Explain your reasoning.

This is not necessarily the case because cross-entropy is computed in relation to probability and note the actual classification prediction.

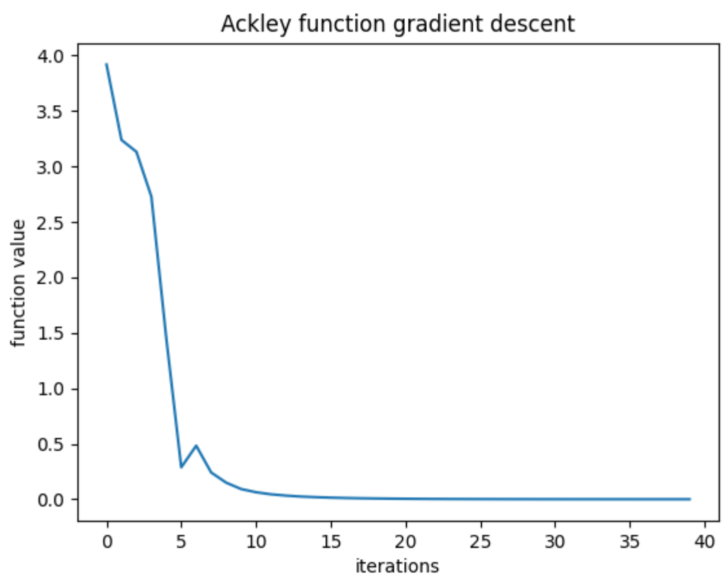
3 Gradient Descent

We set out to minimize the Ackley function. It can be written as:

$$f(x,y) = 20 \cdot e^{-0.2\sqrt{0.5(x^2+y^2)}} - e^{0.5(\cos(2\pi x)+\cos(2\pi y))} + e + 20$$

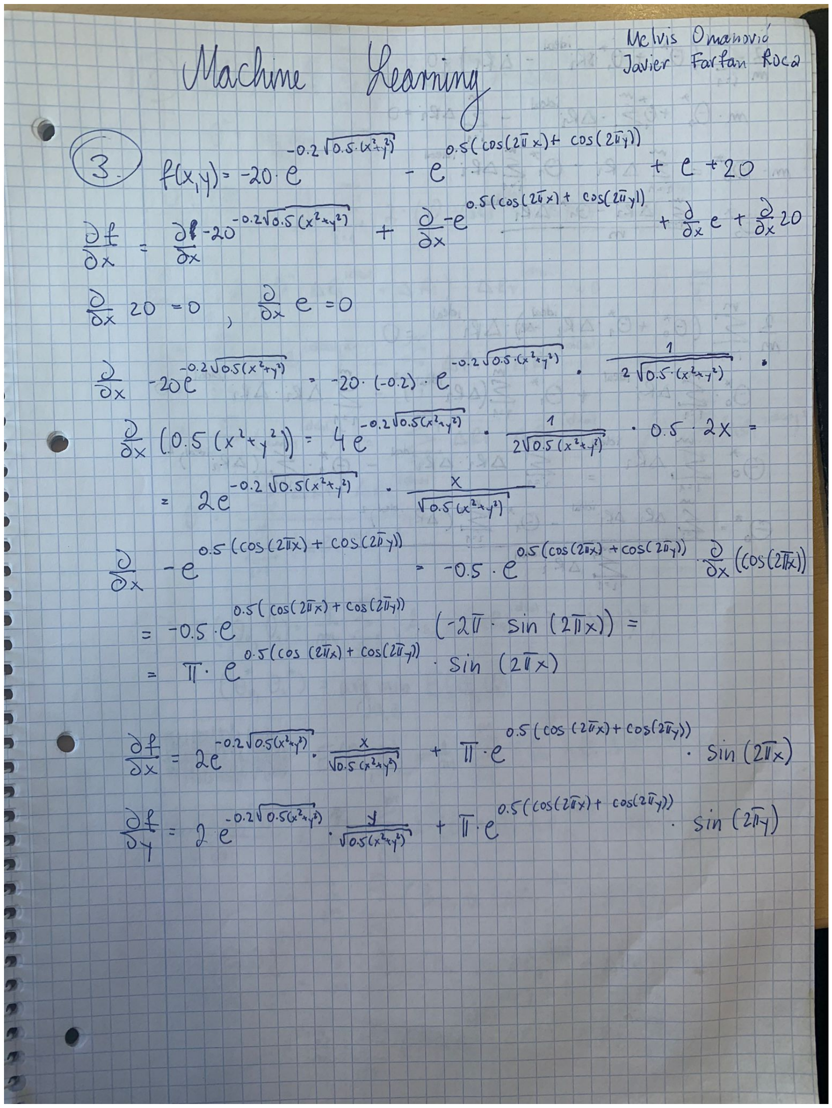
We can reach its global minimum at point (0,0) through gradient descent.

The chosen hyperparameters are as follows:
Number of iterations: 40
Learning rate: 0,1
Learning rate decay: 0,8



The Ackley function is regularly used as a test for optimization algorithms. This function is not convex and it poses a challenge mainly because it has numerous local minima which could "trap" an algorithm and keep it from finding the global minima.

What if we were to use a constant step size?
If we were to set the learning rate decay to 1 and thus have a constant step size it would be very difficult for a function to converge towards the optimum because it would keep "overshooting".



Machine Learning

Melvis Omanović
Javier Farfan Roca

(3) $f(x,y) = -20\cdot e^{-0.2\sqrt{0.5\cdot(x^2+y^2)}} - e^{0.5(\cos(2\pi x)+\cos(2\pi y))} + e + 20$

$$\frac{\partial f}{\partial x} = \frac{\partial f}{\partial x} -20 e^{-0.2\sqrt{0.5(x^2+y^2)}} + \frac{\partial}{\partial x} -e^{0.5(\cos(2\pi x)+\cos(2\pi y))} + \frac{\partial}{\partial x} e + \frac{\partial}{\partial x} 20$$

$$\frac{\partial}{\partial x} 20 = 0, \quad \frac{\partial}{\partial x} e = 0$$

$$\frac{\partial}{\partial x} -20e^{-0.2\sqrt{0.5(x^2+y^2)}} = -20\cdot(-0.2)\cdot e^{-0.2\sqrt{0.5\cdot(x^2+y^2)}} \cdot \frac{1}{2\sqrt{0.5\cdot(x^2+y^2)}} \cdot \frac{\partial}{\partial x}(0.5(x^2+y^2)) = 4e^{-0.2\sqrt{0.5(x^2+y^2)}} \cdot \frac{1}{2\sqrt{0.5(x^2+y^2)}} \cdot 0.5 \cdot 2x =$$

$$= 2e^{-0.2\sqrt{0.5(x^2+y^2)}} \cdot \frac{x}{\sqrt{0.5(x^2+y^2)}}$$

$$\frac{\partial}{\partial x} -e^{0.5(\cos(2\pi x)+\cos(2\pi y))} = -0.5\cdot e^{0.5(\cos(2\pi x)+\cos(2\pi y))} \cdot \frac{\partial}{\partial x}(\cos(2\pi x))$$

$$= -0.5\cdot e^{0.5(\cos(2\pi x)+\cos(2\pi y))}(-2\pi\cdot\sin(2\pi x)) =$$

$$= \pi\cdot e^{0.5(\cos(2\pi x)+\cos(2\pi y))}\cdot\sin(2\pi x)$$

$$\frac{\partial f}{\partial x} = 2e^{-0.2\sqrt{0.5(x^2+y^2)}}\cdot\frac{x}{\sqrt{0.5(x^2+y^2)}} + \pi\cdot e^{0.5(\cos(2\pi x)+\cos(2\pi y))}\cdot\sin(2\pi x)$$

$$\frac{\partial f}{\partial y} = 2e^{-0.2\sqrt{0.5(x^2+y^2)}}\cdot\frac{y}{\sqrt{0.5(x^2+y^2)}} + \pi\cdot e^{0.5(\cos(2\pi x)+\cos(2\pi y))}\cdot\sin(2\pi y)$$

Ackley function gradient