Homework 3 Solutions

Problem 1

A) Loss Function = $\sum_{i=1}^{11}(p(x_i; A_{12}, A_{21}) - p_i)^2$

OLS exists when:

- X^TX is invertible
- N < p
- N >= p but X*w=0
- Linear model

y = $\exp(a_1 + a_2 x + a_0)$

dy/da = $\exp(a^T x)$ * [x1 x2 1]

gradient is still a function of a and so is not linear meaning we can't use Least Squares for this model

B) Output:

```
Loss for current set is =  tensor(0.6702, grad_fn=<SumBackward0>)
New set of A values =  tensor([1.9584, 1.6892], requires_grad=True)

Loss for current set is =  tensor(0.6702, grad_fn=<SumBackward0>)
New set of A values =  tensor([1.9584, 1.6892], requires_grad=True)

Loss for current set is =  tensor(0.6702, grad_fn=<SumBackward0>)
New set of A values =  tensor([1.9584, 1.6892], requires_grad=True)

Loss for current set is =  tensor(0.6702, grad_fn=<SumBackward0>)
New set of A values =  tensor([1.9584, 1.6892], requires_grad=True)

Loss for current set is =  tensor(0.6702, grad_fn=<SumBackward0>)
New set of A values =  tensor([1.9584, 1.6892], requires_grad=True)

The final data sets for A =  [1.958413  1.6891907]
The loss at this location =  0.6702072
The gradiant at this location =  tensor([0., 0.])
```

Code is in the python file labeled Homework 3.py

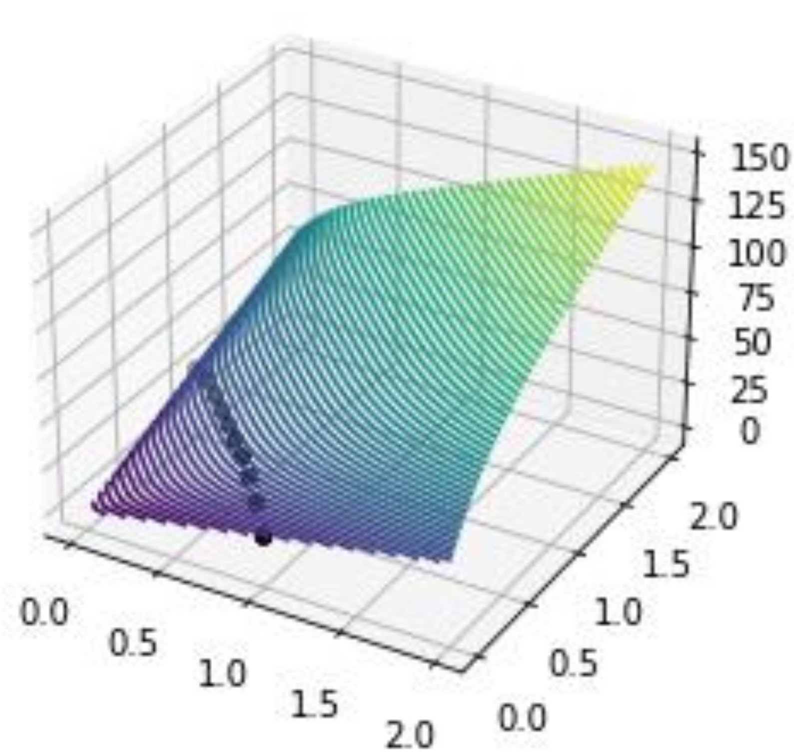
C) The loss couldn't be reduced to zero, so some error still exists between the actual model and the optimized model. However, this loss is minimal and only depends on how accurate you need the model to be as the loss between each point is in the hundredths or thousandths

A12 = 1.958413 A21 = 1.6891907

Optimized Model = $x_1 \exp\left(1.958\left(\frac{1.689x_2}{1.958x_1+1.689x_2}\right)^2\right)p_{water} +$
$x_2 \exp\left(1.689\left(\frac{1.958x_1}{1.958x_1+1.689x_2}\right)^2\right)p_{dioxane}$

p_pred	28.8241	34.64429	36.45295	36.8673	36.874	36.74983	36.39044	35.38483	32.9478	27.73003	17.47325
p_actual	28.1	34.4	36.7	36.9	36.8	36.7	36.5	35.4	32.9	27.7	17.5
loss_pred	0.52432	0.059678	0.061033	0.001069	0.005476	0.002483	0.012003	0.00023	0.002285	0.000902	0.000715
loss	0.670194										

Plot of the model with the measured data points:



Problem 2

Output:

```
Using Bayesian Optimization on the function returns:
X1 =  0.0918079780344101
X2 =  -0.7167301500347611
Y =  -1.0314847263331826
With a gradient =  [ 0.01124661 -0.06532344]

Using gradient descent to refine values returns:
X1 =  0.09077434518912046
X2 =  -0.7132673143955719
Y =  -1.0316225759431568
With a gradient =  [0. 0.]
```

Code is in the python file labeled Homework 3.py