[AAA] Advanced Analytics and Applications

Summer Semester 2021

Problem Set 7 – Neural Networks & Gradient Descent

1. Calculation Task:

a. Analytically calculate the minima of the following function. You don't need to verify whether the critical point is a minima using the hessian matrix.

$$x^2 + y^2 + x$$

b. Now, for the same function, calculate the minima using the gradient descent algorithm. Select a learning rate of 0.3, (x0, y0) = (-1, 0) as the starting point, and terminate after 2 iterations.

c. What would happen if the learning rate is set to 1?

2. Programming Neural Networks:

a. **Regularization**:
   i. What does it mean to reduce the network size (i.e., complexity) of neural network? Explain using a code snippet.
   ii. Implement a neural network of your choice and compare its performance using no regularization techniques, L1 regularization, and L2 regularization.
   iii. Additionally, implement a dropout layer and compare its results.

b. **Gradient Descent**: The next task is to implement the gradient descent algorithm by hand using Python. Assume we have one independent feature x and one cont. output feature y. Our task is to estimate a regression line which models the linear relationship between

$$y = m*x + b$$

   i. Implement a function that estimates the coefficients m and b for a given dataset X (1 dimensional) and its true outputs Y (1 dimensional) using gradient descent. The number of iterations and the learning rate must be passed to the function.

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.datasets import california_housing
from sklearn.metrics import mean_squared_error

housing_data = california_housing.fetch_california_housing()

features = pd.DataFrame(housing_data.data, columns=housing_data.feature_names)
target = pd.DataFrame(housing_data.target, columns=['Target'])

df = features.join(target)

X = df.MedInc
Y = df.Target
```

ii. Additionally, extend your gradient descent algorithm to the stochastic gradient descent algorithm by using only a random subsample of size k (must be too a parameter) at each iteration. Note, that this is not the momentum approach!