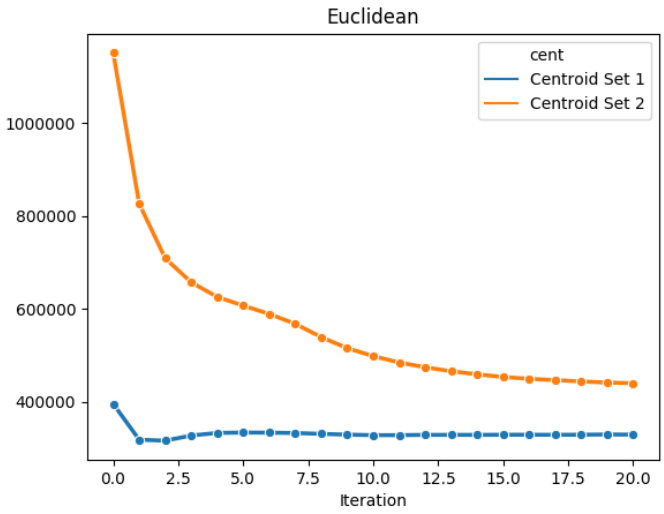
BIG DATA MINING, HW2, RESULTS

RORY FLYNN

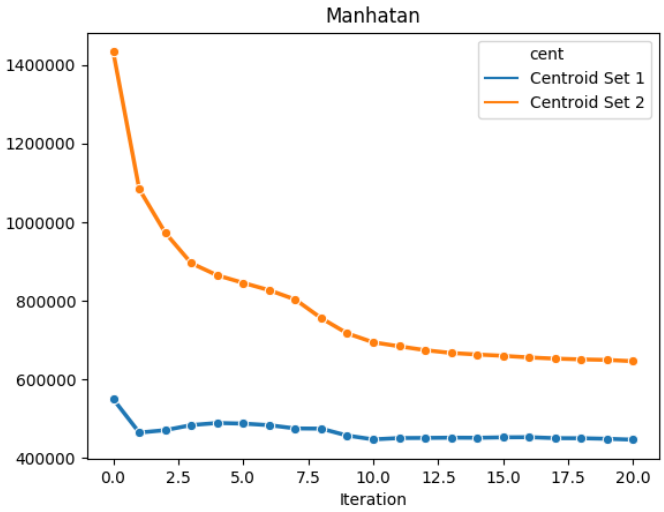
Part 1

Deliverable 1.1: The code as a Jupytier Notebook (.ipynb) or a .txt file (30 points). The code is in one txt file which will be submitted along with this document. It is used with a driver python file also included as a txt.

Deliverable 1.2: A plot of cost vs. iteration for two initialization strategies for a.1 (5 points). Below is a plot of the Euclidean Distance Cost vs iterations, for random cluster initialization(c1), and max-distance cluster initialization(c2).



Deliverable 1.3: A plot of cost vs. iteration for two initialization strategies for b.1 (5 points). Below is a plot of the Manhattan Distance Cost vs iterations for random cluster initialization(c1), and max-distance cluster initialization(c2).



Deliverable 1.4: Your answer for questions a.2 and b.2 (10 points.

- **a.2:** If you want to save time and computation power while optimising you clustering in terms of the cost metric; then our results indicate that you should chose random cluster initialization(c1), over max-distance cluster initialization(c2). One need only look at the graphs above to see a clear example of the c1 method. The max-distance centroids not only start with higher cost, they seem to reach convergence well above the random centroids.
- **b.2:** Whether the cost is calculated in Euclidean or Manhattan distance the answer remains the same. Random is clearly superior to max distance in this case. It is notable that both metrics produce similar results, with only trivial differences. This may be more interesting if cosine distance was used instead of Manhattan.

Part 2

Deliverable 2.1: The equations for ϵ_{iu}. Update equations in the Stochastic Gradient Descent algorithm (2.a)(5 points). The equation for ϵ_{iu} in python is:

```
eiu = np.asscalar(2*(riu - np.dot(qu, pi.T)))
```

The equation for ϵ_{iu} in LaTeXis:

$$\epsilon_{iu} = 2(r_{iu} - q_u \cdot p_i^t)$$

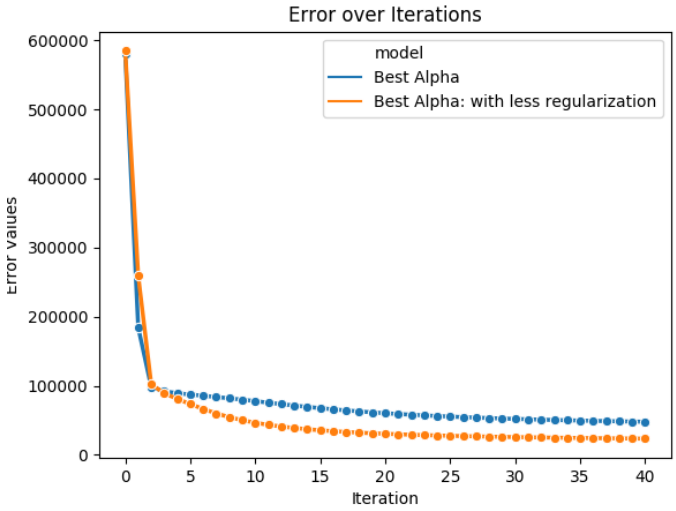
The equations for the updating in python is:

```
qi = qi + a*(exi*px - 2*l*qi)
px = px + a*(exi*qi - 2*l*px)
```

The equations for the update equations in LaTeXis:

$$q_u = q_u + \nu(\epsilon_{iu} \cdot p_i - 2\lambda \cdot q_u)$$
$$p_i = p_i + \nu(\epsilon_{iu} \cdot q_u - 2\lambda \cdot p_i)$$

Deliverable 2.2: What was the lowest error you got? What was the value of ν? (5 points). The lowest error I got with 40 iterations and a regularization factor of 0.1 was 47975.05, with $\nu = 0.01$. When I also ajusted the regularization factor(λ), I got 23509.74, whith $\nu = 0.01$, and $\lambda = 0.01$. The second value may constitute over-fitting however.



Deliverable 2.3: For the best ν, plot of E vs. number of iterations. Make sure your graph has a y-axis so that we can read the value of E. (5 points).

Deliverable 2.4: The code as a Jupyter Notebook (.ipynb) or a .txt file. (35 points). The code is in one txt file which will be submitted along with this document. It is used with a driver python file also included as a txt.