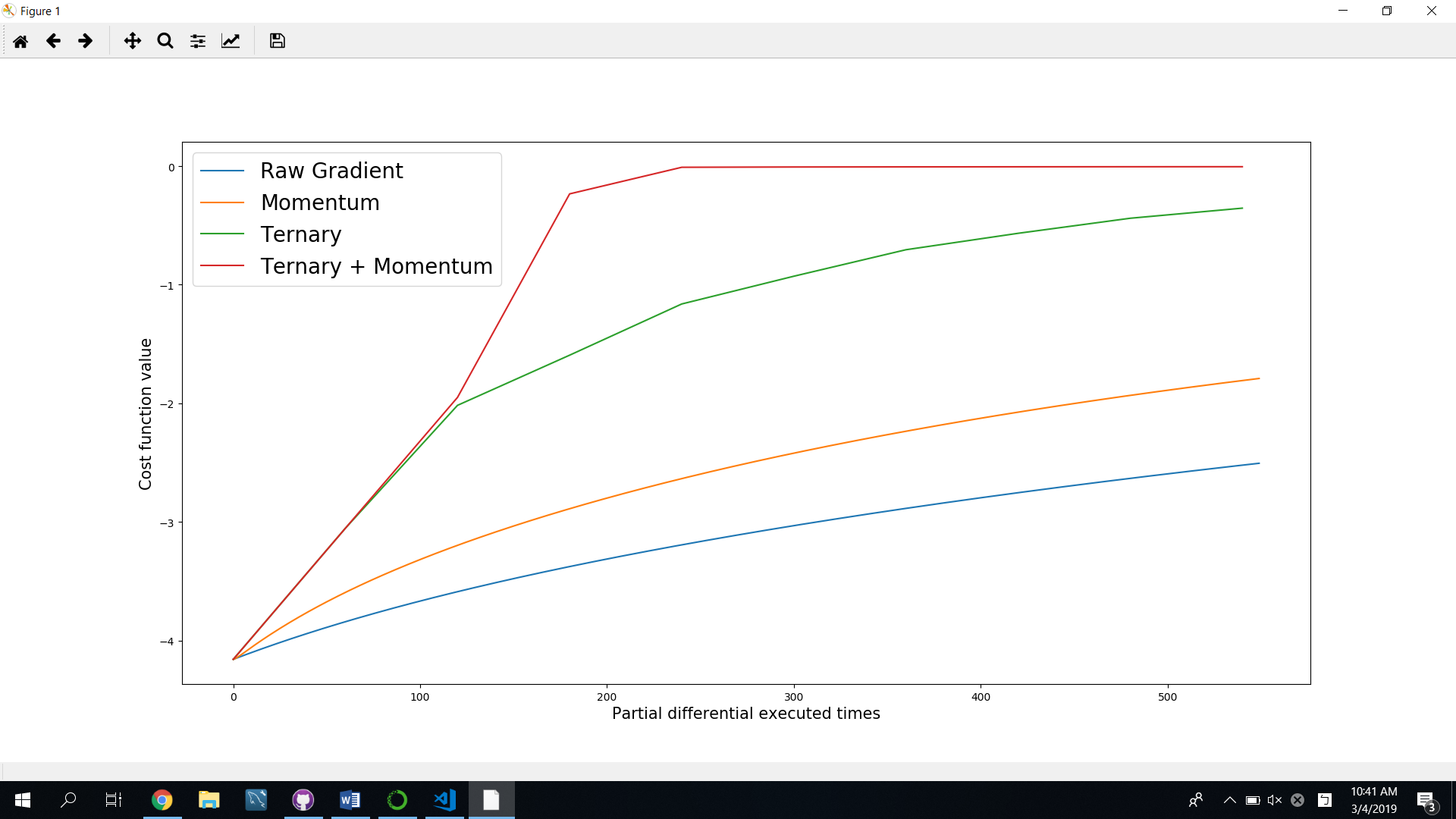
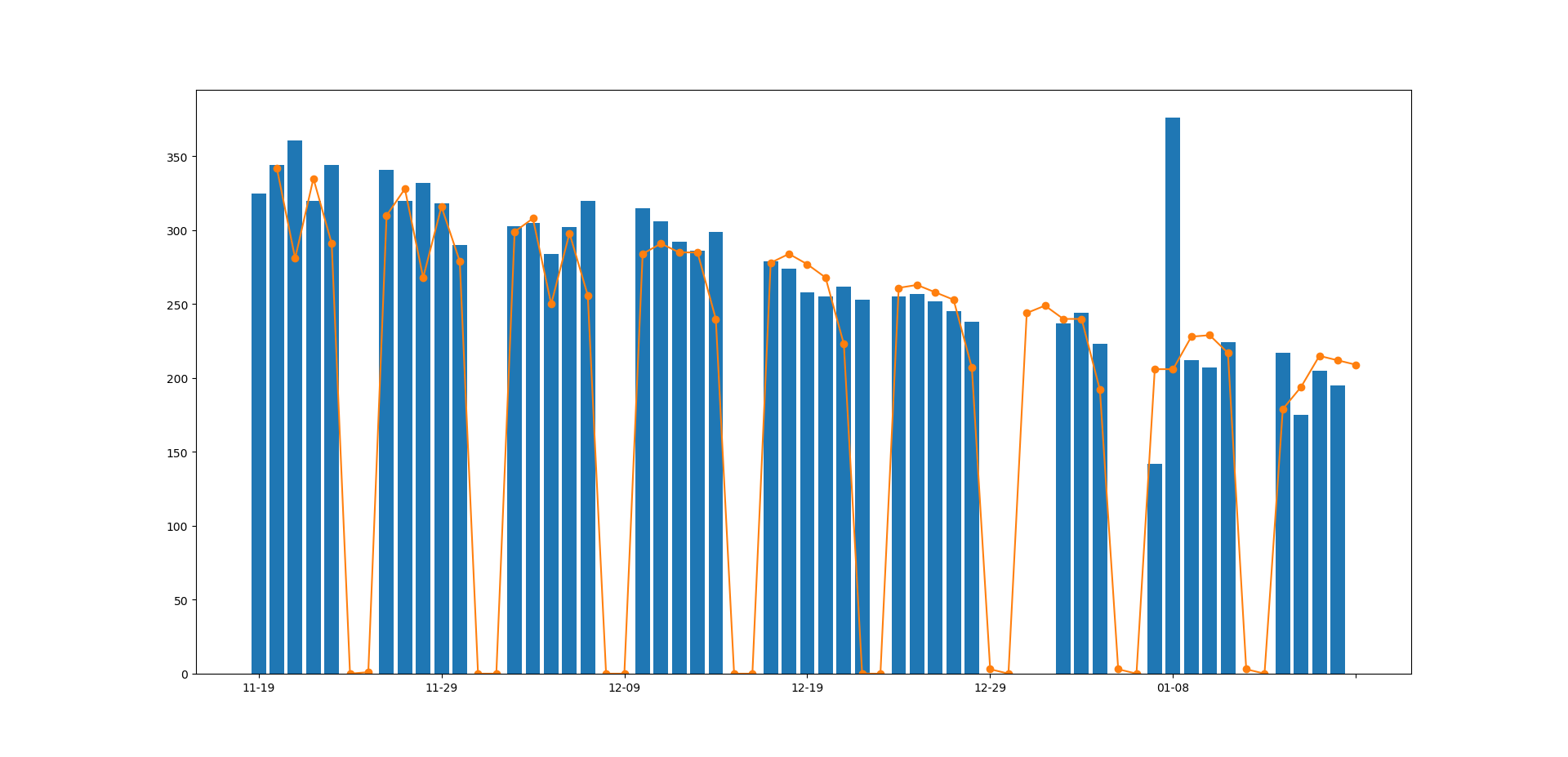
下圖為效能比較圖，損失函數越大越接近完成訓練，橫軸為計算偏微分的次數，縱軸為損失函數的值。



|  |  |  |
| --- | --- | --- |
|  | 優勢 | 劣勢 |
| 梯度上升法 | 實作簡單 | 有可能卡在局部最大值  迭代過慢  對於極度敏感 |
| 三分搜尋法 | 能夠處理過大的  而且損失函數必遞減 | 有可能卡在局部最大值 |
| 動量法 | 能夠脫離局部最大值 | 損失函數可能不增反減 |
| 動量 + 三分搜尋法 | 能夠處理過大的 | 損失函數可能不增反減 |

|  |
| --- |
| *Algorithm: LU Decompose* |
| float A[N][N] ,B[N] ,Y[N] ,X[N] ,L[N][N] ,U[N][N] ,rate ,sum;  int i ,j ,k;  for(i = 0;i != N;i++) L[i][i] = 1;  for(i = 0;i < N;i++) for(j = i + 1;j < N;j++) {  rate = A[j][i] / A[i][i];  L[j][i] = rate;  for(k = 0;k < N;k++) A[j][k] -= rate \* A[i][k];  } |



以六十天前的資料來訓練數量模型，藍色長條為實際點餐人數，橘色為模型預測值。

隨機取十三天作為測試，訓練資料為六十天前的資料，整體模型的預測準度如下，平均下來，整體模型的準確度大概有11.55%。