Data Mining 21/22 Homework 3

Marco Calamo

November 2021

All work is tracked on Github at `https://github.com/IlKaiser/DM22_HW`

1 Exercise 1

The First Exercise notebook is available at `https://gist.github.com/IlKaiser/9cec5be8ce989695697e1fa86e541562` (the same version handed with the homework).
The simplified version (with full dataset taken) is available at `https://gist.github.com/IlKaiser/9b238ee36ab187978bb7b5be0008436e`.
The whole Steam Dataset counts more than 21 millions rows times 22 columns and occupies about 8GB on the disk.
This made me opt for the cloud python notebook platform that offers Kaggle for free in order to process the data without having necessarily to download the whole file on my own computer.
Since the Dataset is too big to fit even in the RAM memory Kaggle offered, I did the entire computation on a small fraction of the Dataset, only a few MB, by taking a row every 1000 trying to have a significant sample of data.
After the method was settled, I used a free-tier Google Cloud computer with 8 cores and 64 GB of RAM in order to process bigger section of data (up to 1GB with full method and full dataset with simplified method).
I used mostly ***sklearn,nltk,numpy,pandas*** and ***matplotlib*** Python libraries to elaborate data.

1.1 Raw Data Process

In order to compute k-means on the row data, it is necessary to first drop the columns that are not numbers (in this case we drop String and Boolean columns). Afterwards it is necessary – in order to have the algorithm work – to drop the rows with eventually missing information (like NaNs or Infite values).
In order to determine the best value for k, I used the elbow method with a similar implementation to the one provided with the homework text, with both Distortion and Inertia as metrics. The optimal value came up to be **k=3**.
The computation overall is quite fast after the import of the datset and the elbow method takes about 3 minutes on 200k entries.

In order to print labelled data I performed PCA for 2 dimensions to the resulting dataset.
All graphs and stats are available on the final notebook.

1.2 Feature Engineering

The most meaningful part of the dataset is of course the **review text content** and I think it is necessary to have it included in the k-means computation. Afterwards I made made some other tweaks in order to have the dataset processed in the most accurate way possible. The total steps were:

- **Text pre-processing** (Tokenization, Stemming, Filtering) the reviews in all different languages
- **TF-IDF** on the most used words among reviews to keep the size of the dataset not too big for computation but at the same time adding useful information.
- **Features aggregation and feature drop**: aggregating all reviews related and authored related info from about 10 different columns in two resulting columns while dropping the others. The accuracy boost was noticeable
- **One-Hot encoding** for Boolean,languages and app name column
- **Min-Max Normalization** for better k-means performances
- **PCA on 256 dimensions** for a smaller but significant dataset to process
- **Elbow method** in order to compute the new optimal number of clusters.

The final cluster size is **k=4** with way much more distinct cluster shapes.
The computation time is about six times greater with about 4 minutes, for 200k entries (for the elbow method only) and about 14 minutes for the whole pre-processing steps described before.
In order to print labelled data I performed PCA for 2 dimensions to the resulting dataset.
All graphs and stats are available on the final notebook.

1.3 Conclusion

Despite having the computation time increased by almost a factor of 10, the feature engineering part was very useful to making the dataset processed in a meaningful way. The interesting part was to observe that the tendencies of the smaller sample are confirmed in the full dataset, implying the sampling and pre-processing were consistent.

2 Exercise 2

2.1 One Dimensional Polynomial K-means

It is possible to find optimal solution for k-means problem where $d = 1$ with a dynamic programming algorithm.
We define a sub-problem as finding the minimum cost of clustering $x_1, ..., x_i$ into m clusters, with $i < n$.
First is necessary to sort the data in a non-decreasing way, then record the corresponding minimum cost in entry $D[i, m]$ of an n + 1 by k + 1 matrix D. Thus $D[n, k]$ is the minimum cost value to the original problem.
Let j be the index of the smallest number in cluster m in an optimal solution to $D[i, m]$. It is evident that $D[j-1, m-1]$ must be the optimal cost for the first $j-1$ points in $m-1$ clusters, for otherwise one would have a better solution to $D[i, m]$. We can fill the matrix using this equation:

$$D[i, m] = \min_{m \leq j \leq i} \{D[j-1, m-1] + d(x_j, \ldots, x_i)\} \tag{1}$$

Where $1 \leq i \leq n, 1 \leq m \leq k$, and $d(x_j, ..., x_i)$ is the sum of squared distances from $x_j, ..., x_i$ to their mean. The matrix is initialized as $D[i, m] = 0$, when $m = 0$ or $i = 0$.
In this way the optimality is guaranteed by the correctness of the equation.
By definition each entry requires $\mathcal{O}(n^2)$ time to compute using the given recurrence if $d(x_j, ..., x_i)$ is computed in linear time, resulting in $\mathcal{O}(n^3 k)$ total runtime.

2.2 One Dimensional K-Means Cost

2.2.1 3 - Means reduced by $\frac{3}{4}$

If we divide the data in three equal clusters C_0, C_+, C_-, one having as a center 0, the others having ℓ = the average of all positive x_i and $-\ell$ = the average of all negative x_i.
We choose to demonstrate for $+\ell$ but the same computation is valid for $-\ell$

$$\sum_i |x_i - \ell|^2 = \sum_i |x_i - \frac{\sum_{i \in C_+} x_i}{m}|^2 \tag{2}$$

With some simple algebra steps:

$$\sum_i (x_i^2 + (\frac{\sum_{i\in C+} x_i}{m})^2 - 2x_i\frac{\sum_{i\in C_+} x_i}{m}) \leq \frac{3}{4}\sum_i x_i^2 \tag{3}$$

$$\sum_i (-\frac{1}{m^2}(\sum_{i\in C_+} x_i)^2 + \frac{2}{m}x_i \sum_{i\in C_+} x_i) \geq \frac{1}{4}\sum_i x_i^2 \tag{4}$$

$$\frac{2}{m}(\sum_i x_i)^2 - \frac{1}{m}(\sum_i x_i)^2 \geq \frac{1}{4}\sum_i x_i^2 \tag{5}$$

$$(\sum_i x_i)^2 \geq \frac{m}{4}\sum_i x_i^2 \tag{6}$$

and 6 is always true due to cluster shapes. □

2.2.2 Reduce cost to ϵ

Repeating the steps described on 2.2.1 after adding exactly $\log_{\frac{3}{4}}(\frac{c}{\epsilon c})$ centers (where c is optimal cost), which are $\mathcal{O}(\frac{1}{\epsilon})$.