Data Mining: Learning from Large Data Sets - Fall Semester 2015

myhshirley@student.ethz.ch
luchen@student.ethz.ch
zhhan@student.ethz.ch

December 2, 2015

Extracting Representative Elements

The goal is to extract representative elements from a large data set $\mathcal{D}$. Suppose the representative subset is $\mathcal{C}$ and its size is k, $|\mathcal{C}| = k$, we want to minimize the following object function:

$$\min_{\mathcal{C},|\mathcal{C}|=k} Q(\mathcal{C}) = \frac{1}{|D|} \sum_{x \in \mathcal{D}} \min_{c \in \mathcal{C}} \|x - c\|^2 \tag{1}$$

Typically, Lloyd's algorithm is used. However, due to its complexity, Lloyd's algorithm cannot be directly used for this project. In order to solve this problem, we tried two approaches: Weighted k-means via importance sampling and Multi-level k-means.

Method1: Extracting Representative Elements via Coresets

The basic idea is to first summarize original data set $\mathcal{D}$ to $\mathcal{D}'$ and then perform weighted k-means on $\mathcal{D}'$. Detailed procedures work as follows:

- **mapper:** Each mapper performs importance sampling on the original data set, according to the algorithm in [1]. Here, we set the coreset size to 200 in each mapper.
- **reducer:** The only reducer merges all coresets $\mathcal{D}'_1, \ldots, \mathcal{D}'_N$ (suppose there are N mappers used) to $\mathcal{D}'$ and performs weighted Lloyd's algorithm on $\mathcal{D}'$.

Method2: Extracting Representative Elements via Multi-level K-means

The basic idea is to perform k-means several times. That is we first divide $\mathcal{D}$ into several subsets and perform k-means on each subset. Then we take as input all cluster centers which could represent the structure of each subset and perform k-means on these cluster centers. And these two steps can exactly be done in mapper and reducer separately.

Algorithm 1: Coreset construction

Input: Data set D, ε, δ, k
Output: Coreset $C = \{(\gamma(\mathbf{x}_1), \mathbf{x}_1), \ldots, (\gamma(\mathbf{x}_{|C|}), \mathbf{x}_{|C|})\}$
$D' \leftarrow D$; $B \leftarrow \emptyset$;
while $|D'| > 10dk\ln(1/\delta)$ **do**
  Sample set S of $\beta = 10dk\ln(1/\delta)$ points uniformly at random from D';
  Remove $\lceil |D'|/2 \rceil$ points $\mathbf{x} \in D'$ closest to S (i.e., minimizing $\mathrm{dist}(\mathbf{x}, S)$) from D';
  Set $B \leftarrow B \cup S$;
Set $B \leftarrow B \cup D'$;
for *each* $b \in B$ **do** $D_b \leftarrow$ the points in D whose closest point in B is b. Ties broken arbitrarily;
for *each* $b \in B$ *and* $\mathbf{x} \in D_b$ **do**
  $m(\mathbf{x}) \leftarrow \left\lceil \frac{5}{|D_b|} + \frac{\mathrm{dist}(\mathbf{x},B)^2}{\sum_{\mathbf{x}' \in D} \mathrm{dist}(\mathbf{x}',B)^2} \right\rceil$;
Pick a non-uniform random sample C of $10\lceil dk|B|^2\ln(1/\delta)/\varepsilon^2 \rceil$ points from D, where for every $\mathbf{x}' \in C$ and $\mathbf{x} \in D$, we have $\mathbf{x}' = \mathbf{x}$ with probability $m(\mathbf{x})/\sum_{\mathbf{x}' \in D} m(\mathbf{x}')$;
for *each* $\mathbf{x}' \in C$ **do** $\gamma(\mathbf{x}') \leftarrow \frac{\sum_{\mathbf{x} \in D} m(\mathbf{x})}{|C| \cdot m(\mathbf{x}')}$;

Figure 1: Important sampling algorithm in [1].

- **mapper:** Each mapper performs k-means (we call the *MiniBatchKMeans()* function in *python slearn package*[1]) and send these centers to reducer. Here, we set the cluster number to 200 for each mapper.
- **reducer:** The only reducer receives the centers from mappers and uses Lloyd's algorithm to get the final 100 representative elements.

Finally, the second approach achieves our best score.

Contribution

Each of us worked out a solution and we put together our codes to get the final solution. We also tried different parameters, e.g. the sample size / cluster number for each mapper, until we find a good enough solution.

References

[1] Dan Feldman, Matthew Faulkner, and Andreas Krause. Scalable training of mixture models via coresets. In *Advances in Neural Information Processing Systems*, pages 2142–2150, 2011.

[1] sklearn MiniBatchKMeans