CSE6250 Homework3 Answer

Hongshen Zhao

September 30, 2018

1 Programming: Rule based phenotyping

No report deliverable in this section.

2 Programming: Unsupervised Phenotyping via Clustering

2.1 Feature Construction

No report deliverable in this section.

2.2 Evaluation Metric

No report deliverable in this section.

2.3 K-Means Clustering

b. The results are in Table 1 and Table 2.

Percentage Cluster	Case	Control	Unknown
Cluster 1	11.68%	61.92%	32.313%
Cluster 2	88.32%	37.55%	64.342%
Cluster 3	0%	0.53%	3.345%
	100%	**100%**	**100%**

Table 1: K-Means Clustering with 3 centers using all features

Percentage Cluster	Case	Control	Unknown
Cluster 1	1.54%	100%	1.23%
Cluster 2	98.46%	0%	29.54%
Cluster 3	0%	0%	69.23%
	100%	**100%**	**100%**

Table 2: K-Means Clustering with 3 centers using filtered features

2.4 Clustering with Gaussian Mixture Model (GMM)

b. The results are in Table 3 and Table 4.

Percentage Cluster	Case	Control	Unknown
Cluster 1	7.68%	2.11%	11.85%
Cluster 2	73.26%	43.25%	63.66%
Cluster 3	19.06%	54.64%	24.49%
	100%	**100%**	**100%**

Table 3: GMM Clustering with 3 centers using all features

Percentage Cluster	Case	Control	Unknown
Cluster 1	80.23%	0%	26.56%
Cluster 2	0.20%	97.78%	57.23%
Cluster 3	19.57%	2.22%	16.21%
	100%	**100%**	**100%**

Table 4: GMM Clustering with 3 centers using filtered features

2.5 Clustering with Streaming K-Means

a. Steaming K-Means algorithm:

Suppose at time t there are total N_t new data points. The streaming K-Means algorithm can be divided into the following four steps:

(1) Assign new data points to the closest cluster.

(2) Calculate the discount according to the time unit and update the cluster weights: begindocument

```
procedure UpdateDiscount
    if $timeUnit = StreamingKMeans.POINTS$ then
        $\alpha_t$ = weight discount
        $\alpha_t = math.pow(decayFactor, N_t)$
    else if $timeUnit = StreamingKMeans.BATCHES$ then
        $\alpha_t = \exp(\log(0.5)/halfLife)$
```

(3) Update clusters:

```
procedure UpdateClusters
    for $i = 1$ to $k$ do
        $c_t^i$ = centroid vector of the ith cluster
        $m_t^i$ = number of new points in the ith cluster
        $vector\_sum_t^i$ = sum of all vectors in the ith cluster
        $x_t^i = vector\_sum_t^i / m_t^i$, the centroid estimated on the current batch
        $n_t^i$ = weight of the ith cluster
        $n_{t+1}^i = n_t^i \alpha_t + m_t^i$
        if $n_{t+1}^i < 1^{-6}$ then
            // avoid numerical instability
            $\lambda = m_t^i / 1^{-6}$
        else
            $\lambda = m_t^i / n_{t+1}^i$
        $c_{t+1}^i = (1 - \lambda) c_t^i + \lambda x_t^i$
```

In normal case, the ith centroid will be updated to

$$c_{t+1}^i = \frac{c_t^i n_t^i \alpha_t + x_t^i m_t^i}{n_t^i \alpha_t + m_t^i} \tag{1}$$

The new weight for the ith cluster will be updated to

$$n_{t+1}^i = n_t^i \alpha_t + m_t^i \tag{2}$$

(4) Adjust the weights of the smallest and the largest clusters.

reference: `https://github.com/apache/spark/blob/master/mllib/src/main/scala/org/apache/spark/mllib/clustering/StreamingKMeans.scala`

In the update rule (3), the new centroid is updated by a weighted average of the centroid of the existing data points and the centroid of the new data points. The weight for the existing data points is an exponentially-weighted moving average of the vector number and the weight for the new data points is the count of the new vectors.

Pros:

(1) Can perform fast predictions on the fly by only being trained by the data received and attractively update the model accuracy.

(2) Has the freedom to adjust the weights of the data received in the past to improve performance.

Cons:

(1) Data are weighted differently by the decay value α_t which can hurt the performance if not well selected.

(2) Additional computation cost is incurred to perform weight decays and moving average calculations.

The forgetfulness value α_t balances the relative importance of new data versus past history by producing an exponentially-weighted moving average. With $\alpha_t = 1$ all data will be used from the beginning; with $\alpha_t = 0$ only the most recent data will be used.

c. The results are in Table 5 and Table 6.

Percentage Cluster	Case	Control	Unknown
Cluster 1	15.88%	14.24%	26.70%
Cluster 2	78.89%	43.14%	62.47%
Cluster 3	5.23%	42.62%	10.83%
	100%	**100%**	**100%**

Table 5: Streaming K-Means Clustering with 3 centers using all features

Percentage Cluster	Case	Control	Unknown
Cluster 1	4.92%	2.22%	0.72%
Cluster 2	5.53%	97.78%	70.36%
Cluster 3	89.55%	0%	28.92%
	100%	**100%**	**100%**

Table 6: Streaming K-Means Clustering with 3 centers using filtered features

2.6 Discussion on K-means and GMM

a. Generally, the datasets with filtered features had better clustering results in both 2.3b and 2.4b compared to the datasets with all features.

In 2.3b, cluster 1 and cluster 2 had the most of the data points while cluster 3 had little with all features. After filtering the features, the purity was greatly improved with cluster 1 containing 100% Control class patients, cluster 2 containing 98.46% Case class patients and cluster 3 containing 69.23% Unknown class patients.

In 2.4b, cluster 2 and cluster 3 had the most of the data points while cluster 1 had little with all features. After filtering the features, the purity was slightly improved with cluster 1 containing 80.23% Case class patients, cluster 2 containing 97.78% Control class patients and cluster 3 containing a mix of patients from all three classes.

This showed that many feature were redundant and could be noisy in the clustering process. The dimension reduction using PCA was very efficient in extracting the most essential features.

Comparing K-Means and GMM, they had about the same performance (purity value) with all features. But K-Means outperformed GMM after feature filtering.

b. The results are in Table 5 and Table 7. Generally, purity values increased as k increased. When k was too small, data points from different classes could not be well separated. So, more clusters were needed to perform better clustering by grouping the most similar data points together. But the largest k value was not necessary the most appropriate. This was because purity could be biased by large k values. Imagine the extreme case where each data point is contained in one and only one cluster and the purity value becomes 1. By using elbow method, we could find the optimal number of clusters which was between 6 to 10 because apparent improvement was observed from 5 to 10 clusters in purity values but from 10 to 15 there was little improvement.

	K-Means	K-Means	GMM	GMM
k	All features	Filtered features	All Features	Filtered features
2	0.47831	0.65954	0.47831	0.38565
5	0.55965	0.88306	0.54393	0.69334
10	0.66594	0.89031	0.57430	0.88479
15	0.69794	0.89479	0.59382	0.86651

Table 7: Purity values for different number of clusters