Data Mining: Learning from Large Data Sets - Spring Semester 2014

lukas.elmer@student.ethz.ch
sandro.felicioni@student.ethz.ch
frederick.egli@student.ethz.ch

June 7, 2014

1 Approximate near-duplicate search using Locality Sensitive Hashing

For the fist assignment, we implemented LSH with multiple bands and multiple rows per band.

1.1 Mapper

1. Generate multiple permutations for each shingle
2. Generate a hash for every permutation
3. For each hash, write $< band - nr >$:$< hash >$, so that the probability is high that two duplicates are hashed to the same bucket

1.2 Reducer

In the reducer, we implemented the Jaccard distance function to check if two similar rated items are to 85% similar, or in other words, to have a precision of 100%. Before doing this, we had a score of 0.97964.

1.3 Remarks

We first tried few bands and rows per band, and already got good results. Our last three submissions contained the following number of bands and rows:

1. 35 bands, 20 rows per band: Precision = 1.00000, recall = 0.99496.
2. 40 bands, 20 rows per band: Precision = 1.00000, recall = 0.99664.
3. 55 bands, 20 rows per band: Precision = 1.00000, recall = 1.00000.

Which of course sets the final score of: *(score=1.0)*.

2 Large-Scale Image Classification

final score of: *(score=0.819053)*.

3 Extracting Representative Elements From Large Datasets

The first approach was *K-Means of K-Means*, or in other words, each *mapper* was running a *k-Means* algorithm, and the *reducer* was doing a *K-Means* based on the output of the *mappers*. The result was obviously not that much satisfying so we changed our strategy.

We then decided to implement an *online k-Means*. Therefore each *mapper* just passed the data to the *reducer*, which did the *sequential k-Means*. We also experimented with the *mini batch* feature of *scikit-learn*. Even though we got good results, the score was still too high for the baseline hard.

The final approach was using *coresets*. After reading [2] we had to figure out how many points were necessary to sample in order to get a good coreset (β parameter). After finding β and with a clever choice of initializing the cluster centers during the reducer phase we could successfully beat the baseline hard. with a final score of: *737.16*.

4 Explore-Exploit Tradeoffs in Recommender Systems

The first recommender algorithm was based on random decisions, which does not learn over time. After uploading it to the evaluation system we had a *CTR* of *0.035394*. The first implemented bandit algorithm was *UCB1*. Because it is a *context free* bandit algorithm its results were not satisfying. We then concentrated on the *LinUCB* algorithm. After reading [1] we implementing the first draft of *LinUCB*. But we had concerns about the computational time limit. We refactored the algorithm such that no loops are required for the *arg_max* of the *UCB scores*. *LinUCB* needs one parameter, called α, to regularize the exploration part. We gained the best result with $\alpha = 0.2$.

An additional improvement was achieved by using the timestamps. As soon as a news article was seen for the first time, we initialized a counter. For each article which was still available after 24 hours we reseted the weights.

We had tested plenty of other algorithms, including UCB-V, HybridLinUCB and K-Means, but none of them were satisfying.

Our final score was: 0.059848

References

[1] Lihong Li, Wei Chu, John Langford, Robert E. Schapire. A Contextual-Bandit Approach to Personalized News Article Recommendation. http://www.research.rutgers.edu/ lihong/pub/Li10Contextual.pdf

[2] Dan Feldman, Matthew Faulkner, Andreas Krause. Scalable Training of Mixture Models via Coresets. In Proc. Neural Information Processing Systems, 2011