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

Bojana Dimceva (dbojana@student.ethz.ch)
Diego Ballesteros Villamizar (diegob@student.ethz.ch)

June 10, 2014

1 Approximate near-duplicate search using Locality Sensitive Hashing

Maximum of 2 pages per section.

2 Large-Scale Image Classification

2.1 Algorithm

For this classification problem, the approach used was parallel stochastic gradient descent as described in the class and in [4].

In each of the mappers, a linear SVM was trained using the given subset of data and then the output coefficient vector was passed to the reducer where it was averaged to produce the final linear model for evaluation. This was possible because it was known a priori the number of mappers and there was a single reducer.

The specific details of the implementation of the SVM were left to the SGDClassifier class from the scikit-learn library [2] which implements a batch stochastic gradient descent algorithm.

Another approach that was implemented for this task was the use of stochastic sampling in order to approximate kernel SVMs, this was done by transforming the input vectors in each mapper using the RBFSampler class from scikit-learn. This is the method of using random Fourier features to approximate kernels as described in [3].

2.2 Parameter tuning

For the linear SVM method, the parameters to tune were:

- The loss function, this could be a hinge or logistic loss which implement a soft-margin SVM or logisitic regression respectively.
- The penalty function, this could be L1 or L2 which determines the sparsity of the solution.
- The regularization parameter, which helps reduce overfitting due to increased model complexity.

For the SVM with random features, two extra parameters were added:

- The gamma parameter for the RBF sampler which controls the kernel function.
- The number of random features to generate, more features imply a better approximation but it also increases the complexity of the operation.

These parameters were tuned using 10-fold cross validation on the given test data with full grid search on the parameters.

2.3 Results

The best score was attained using a linear SVM because any attempt to use the random features with a significant number of features resulted in timeouts in the mappers. However, the attained scored was halfway between the easy and hard baseline. It is possible that some values of the parameters were not explored during the cross validation or rather the approach was too simplistic to solve the problem and instead an approach like PEGASOS should have been tried.

3 Extracting Representative Elements From Large Datasets

Maximum of 2 pages per section.

4 Explore-Exploit Tradeoffs in Recommender Systems

4.1 Algorithm

For this problem, the algorithm used was linear upper confidence bound (linUCB) as described in the class' lectures and in [1]. For this task, only the user features provided at each evaluation step were used, so the algorithm can be classified more accurately as the linUCB with disjoint linear models (Algorithm 1 of [1]).

In summary, the implemented algorithm works as follows:

1. When a recommendation is requested, the Upper Confidence Bound is calculated for each article in the given subset. The UCB is computed using the closed solution for the ridge regression problem, i.e. equation 1, presented in [1] where A_a is the inverse of the covariance matrix for article a, b_a is the response vector from previous feedback for article a and x_t is the vector of user features for the current recommendation. Finally, the recommended article is the one with maximum calculated UCB.

$$UCB = (A_a^{-1} b_a)^T x_t + \alpha \sqrt{x_t^T A_a^{-1} x_t} \tag{1}$$

2. After a recommendation is made and the feedback is received then there are two cases to consider:
   - If the feedback is -1, then the article recommended was not recommended by the random policy that generated the logs and therefore no information can be retrieved from that log line. The algorithm simply continues to the next recommendation step.
   - If the feedback is 0 or $+1$, then the article recommended matched the one in the log and an update can be made on the covariance matrix and the mean vector for the recommended article, this update is made using the equations given in [1].

4.2 Parameter tuning

In this algorithm, there is a single parameter that can be used to control the trade-off between exploitation and exploration. From equation 1 it can be observed that this value regulates the effect of the estimated covariance in the UCB, large values of α may lead to too much exploration that wastes exploitation opportunities but values to small may fall short of exploring potentially good options. In order to increase the click through rate (CTR) several submissions were made with different values of α using as a reference the results presented in [1] for the learning bucket, in the end it was found that the optimal value is 0.2 which is the same value that produces a peak in the results from [1].

The result with the chosen algorithm and parameter value were enough to surpass the hard baseline.

References

[1] Li, L., Chu, W., Langford, J., and Schapire, R. E. A contextual-bandit approach to personalized news article recommendation. In *Proceedings of the 19th international conference on World wide web* (2010), ACM, pp. 661–670.

[2] Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., and Duchesnay, E. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research 12* (2011), 2825–2830.

[3] Rahimi, A., and Recht, B. Random features for large-scale kernel machines. In *NIPS* (2007), vol. 3, p. 5.

[4] Zinkevich, M., Weimer, M., Smola, A. J., and Li, L. Parallelized stochastic gradient descent. In *NIPS* (2010), vol. 4, p. 4.