LITERATURE REVIEW
A Parallel Recommender System Using a Collaborative Filtering Algorithm for Movie Recommender System

Projna Saha
School of Computer Science
Carleton University
Ottawa, Canada K1S 5B6
projnasaha@cmail.carleton.ca

October 5, 2021

1 Introduction

In the past decades, recommender systems have gotten a lot of attention. A recommender system is sometimes known as a recommendation tool that takes the information about the user as input and finds a similarity based on the user's "rating" or "preference" for an item. For example, most famous websites such as YouTube, Netflix, IMDb, etc. are using some sort of recommendation systems that provide recommendations of the movies, shows that are similar to the ones that have been watched in the past [1]. In other words, Netflix's machine learning algorithm collects user data each time when a viewer spends time watching a movie or a show. In the mid-1990s, recommender systems emerged as an independent research topic where collaborative filtering got many interests to the researchers.

There are majorly five types of recommender systems that work primarily in the Media and Entertainment industry: Collaborative Recommender system, Content-based recommender system, Community-based recommender system, Knowledge-based recommender system, and Hybrid recommender system [2]. Hence, Collaboration filtering, content-based filtering, and hybrid filtering are the three most used conventional techniques. The idea behind a Content-based (cognitive filtering) recommendation system is to recommend an item based on a comparison between the content of the items and a user profile. In simple words, viewers may get recommendations for a movie based on the description of other movies [3]. On the other hand, collaborative filtering mimics user-to-user recommendations. It predicts user preferences as a linear, weighted combination of other user preferences. Collaborative filtering recommendation algorithm has many advantages, such as high speed, high efficiency, and good robustness, and the recommendation results are more accurate than other algorithms. Content-based filtering can recommend a new item but needs more data of user preference in order to incorporate the best match. Collaborative filters are expected to increase diversity because they help us discover new products. Though collaborative filtering (CF) is one of the most popular algorithms due to its simplicity, it consists of noises in data, i.e., data scarcity, poor accuracy, scalability, and cold start problems and this suffers overall algorithm performance [4].

To reduce the data noise and speed up the time, Sun & Luo [5] developed a novel parallel recommending system based on CF with correntropy. Instead of traditional measures used in recommendation algorithms, the correntropy was employed to compute the similarity of two items or users to achieve insensitive performance to outliers. Furthermore, to reduce the computational cost, Spark framework is a good choice for parallel processing. This paper aiming to evaluate and compare the results using the methods that was previously implemented on various social network datasets. My goal of this work is applying this technique to two different movie datasets, one is MovieLens dataset that they used in their research and another is IMDb dataset. This will help to compare and contrast the performance of parallel processing.

2 Literature Review

2.1 Similarity Computation

In general, CF algorithms can be classified into two categories: neighborhood-based approaches and model-based methods. Though the neighborhood-based collaborative algorithm is gaining popularity for its simplicity, stability, and efficiency methods, it spends lots of time obtaining the recommendation list even though it is faster compared with other algorithms. The most popular technique to create a recommender system is to use CF through neighborhood-based interpolation, and a vital stage is called "neighborhood selection," which is closely tied to the similarity weights measure. There are some methods for calculating similarity based on ratings, and if there is any faulty or incorrect data, their performance will quickly deteriorate [6].

2.1.1 Cosine Vector (CV) Similarity

Despite that user-based collaborative filtering is the most successful technology for building recommender systems to date and is extensively used in many commercial recommender systems, unfortunately, the computing complexity of these solutions grows linearly with the number of customers, which can be several million in typical commercial applications. [7]. Cosine similarity is a useful metric for determining how similar two documents are in terms of a particular topic. In the realm of data mining, the approach is also used to measure cluster cohesion. From the equation, u and s two instances and cosine similarity calculates the distance between their shape vector X_u and X_s. Using r_{ui} to define the rating from user u to item i. This can be obtained $X_{ui} = r_{ui}$ if user u has rated item i, and $X_{ui} = 0$ in other cases. The similarity between two users u and s could be computed by:

$$\cos(u, s) = \cos_{us} = \frac{\sum_{i \epsilon I_{us}} r_{ui} r_{si}}{\sqrt{\sum_{i \epsilon I_u} r_{ui}^2} \sqrt{\sum_{i \epsilon I_s} r_{si}^2}} \tag{1}$$

Where r_{ui} denotes the rating value from user u to item i, and the similar definition applies to r_{si}. In addition, I_{us} denotes the items that both user u and user s rated, the same conception holds true for I_u and I_s.

2.1.2 Pearson Correlation (PC) Similarity

The Pearson correlation coefficient indicates how strong a linear link exists between two variables. Even though CV similarity does not account for differences in the mean and variance of the evaluations, a more popular method, known as PC similarity, was devised. It can solve the problems that CV similarity causes and produce better outcomes [8]. The following is the similarity measure [9]:

$$PC(u,s) = PC_{us} = \frac{\sum_{i \epsilon I_{us}} (r_{ui} - \overline{r_u})(r_{si} - \overline{r_s})}{\sqrt{\sum_{i \epsilon I_{us}} (r_{ui} - \overline{r_u})^2} \sqrt{\sum_{i \epsilon I_{us}} (r_{si} - \overline{r_s})^2}} \tag{2}$$

Where $\overline{r_u}$ denotes the average rating value of user u, and $\overline{r_s}$ denotes the average rating value of user s. It should be noted that the standard deviation of the ratings in evaluations is based on the common items rather than the entire set of items, and z-score normalization may be necessary. The sign of this similarity denotes whether the correlation between two users or two items is direct or inverse, and the magnitude denotes the connection's strength.

2.1.3 Spearman Correlation (SC)

When evaluating correlations involving ordinal variables, Spearman correlation is frequently utilized. It is another way of calculating similarity by using the rank of those ratings in place of actual rating value [10]. The equation becomes,

$$SC(i,j) = SC_{ij} = \frac{\sum_{u \epsilon U} (k_{u,i} - \overline{k_i})(k_{u,j} - \overline{k_j})}{\sqrt[2]{\sum_{u \epsilon U} (k_{u,i} - \overline{k_i})^2} \sqrt[2]{\sum_{u \epsilon U} (k_{u,j} - \overline{k_j})^2}} \tag{3}$$

Where $k_{u,i}$ denotes the rank of the rating of item i of user u and $k_{u,j}$ shows the rank of the rating of item j of user u. $\overline{k_i}$ and $\overline{k_j}$ calculate the average rank of item i and item j.

2.1.4 Adjusted Cosine (AC) Similarity

The adjusted cosine similarity measure is a modified version of vector-based similarity that accounts for the fact that various users have varying rating methods; in other words, some people may score objects highly in general while others may prefer to rank items lower. AC similarity is defined as [5],

$$AC(i,t) = AC_{it} = \frac{\sum_{u \epsilon U_{it}} (r_{ui} - \overline{r_u})(r_{ut} - \overline{r_u})}{\sqrt{\sum_{u \epsilon U_{it}} (r_{ui} - \overline{r_u})^2} \sqrt{\sum_{u \epsilon U_{it}} (r_{ut} - \overline{r_u})^2}} \tag{4}$$

Where U_{it} denotes the users who rated both items i and t. In addition, r_{ui} to define the rating from user u to item i.

2.1.5 JacRA Similarity

In collaborative filtering based recommendation system, cold start, scalability and data sparseness are important factors that affect its performance. Since some existing similarity computing methods are inaccurate in measuring similarity, and accuracy must be improved,

a new similarity calculation method is proposed, which is called JacRA similarity [11]. Defining similarity between two users in JacRA as,

$$sim(u,v) = \frac{|I_u \cap I_v|}{|I_u \cup I_v|} \cdot \frac{\sum_{i \epsilon I_u \cap I_v} \frac{min(r_{ui}, r_{vi})}{max(r_{ui}, r_{vi})}}{|I_u \cap I_v|} \tag{5}$$

Here, $r_{u,i}$ and $r_{v,i}$ are the ratings of users u and v to item i respectively, $I_u \cap I_v$ is the set of item which have been used by users u and v concurrently. $min(r_{ui}, r_{vi})$ is the minimum of $r_{u,i}$ and $r_{v,i}$. $max(r_{ui}, r_{vi})$ is the maximum of $r_{u,i}$ and $r_{v,i}$. Following the same way, similarity between items can be written as in JacRA as,

$$sim(u,v) = \frac{|U_i \cap U_j|}{|U_i \cup U_j|} \cdot \frac{\sum_{u \epsilon U_i \cap U_j} \frac{min(r_{ui}, r_{uj})}{max(r_{ui}, r_{uj})}}{|U_i \cap U_j|} \tag{6}$$

Here, U_i and U_j are the sets of users who have used item i and j, respectively. $U_i \cap U_j$ the set of users who have used item i and j concurrently. $U_i \cup U_j$ are the set of users who have used item i or j

2.2 Mean Squared Distance (MSD)

The mean squared distance (MSD) is a time-dependent measure of a particle's location divergence from a reference point. The similarity between item i and item j using MSD is computed as,

$$MSD(u,s) = MSD_{us} = \frac{I_{us}}{\sqrt{\sum_{i \epsilon I_{us}} (r_{ui} - r_{si})^2}} \tag{7}$$

In comparison to the Pearson Correlation (PC) method, the MSD method's application is limited because it does not capture negative correlations between two people or things [12]. When such negative correlations are taken into account, the forecast accuracy may actually improve.

2.3 Rating Prediction

Rating prediction is a well-known recommendation task that attempts to forecast a user's rating for goods that the user has not yet evaluated. Users' explicit input, i.e. their past ratings on some goods, is used to calculate predictions [13]. The rating prediction system first finds the similarity among users using the previous data available in the data set. Using that similarity, the system predicts the ratings for different items for different users [14].

2.3.1 Weighted Average (WA)

The weighted average of neighborhood ratings is used to predict the ratings of the unobserved items of the target user. A serious problem that may occur is that users provide ratings having a different sense of rating items. For example, some users tend to rate all the items highly and some others low. In order to fix this problem, the ratings must be mean-centered in a row-wise fashion, before determining the average rating of the neighborhood. The weighted average of these ratings is the predicted value and it is given by [25]:

$$\widehat{r}_{ui} = \frac{\sum_{j \epsilon N_u(i)} sim(i,j) r_{ju}}{\sum_j \epsilon N_u(i) |sim(i,j)|} \tag{8}$$

Where r_{iu} and r_{ju} are the ratings of items i and j given by user u and where $\widehat{r}_{ui}$ depicts the predicted value of item i of user u.

2.3.2 Mean-Centering (MC)

In moderated multiple regression models or polynomial regression models, mean centering has been proposed as a solution to problems of collinearity. Due to significant correlations between variables, mean centering does not minimize critical collinearity [15]. By comparing a rating to the mean rating, mean-centering is utilized to establish if it is positive or negative [16]. The equation becomes as the following in the prediction of rating using similarity value with mean centering approach. User-based prediction of a rating r_{ui} can be defined as:

$$\widehat{r}_{ui} = \overline{r_u} + \frac{\sum_{j \epsilon N_u(u)} sim(i,j)(r_{ju} - \overline{r_j})}{\sum_j \epsilon N_u(u)|sim(i,j)|} \tag{9}$$

Where $\overline{r_u}$ represents the average of average of the ratings given by user u to the items in i. This method can also be used for the item-based CF algorithm, and then the item-based predicted rating of r_{ui}.

$$\widehat{r}_{ui} = \overline{r_i} + \frac{\sum_{j \epsilon N_u(i)} sim(i,j)(r_{ju} - \overline{r_j})}{\sum_j \epsilon N_u(i)|sim(i,j)|} \tag{10}$$

Where $\overline{r_i}$ represents the average of the item ratings, i given by the user u.

2.3.3 Z-Score (ZS)

To handle the spread in the individual rating scales, z-score normalization is introduced in addition to the mean-centering method. In addition to the mean-centering method, z-score normalization is used to handle the spread in the individual rating scales [17]. In user-based CF algorithm, it can be obtained by normalizing the rating $\widehat{r}_{ui}$ by dividing the user-mean-centered rating by the standard deviation σ_u of the ratings given by user u:

$$\widehat{r}_{ui} = \overline{r_u} + \sigma_u \frac{\sum_{j \epsilon N_u(u)} sim(i,j)(r_{ju} - \overline{r_j})/\sigma_u}{\sum_j \epsilon N_u(u)|sim(i,j)|} \tag{11}$$

Similarly, in item-based CF algorithm, the Z-score normalization of $\widehat{r}_{ui}$ by dividing the item-meancentered rating by the standard deviation σ_i of ratings given to item i [18],

$$\widehat{r}_{ui} = \overline{r_i} + \sigma_i \frac{\sum_{j \epsilon N_u(i)} sim(i,j)(r_{ju} - \overline{r_j})/\sigma_i}{\sum_j \epsilon N_u(i)|sim(i,j)|} \tag{12}$$

2.4 Correntropy Similarity

In non-Gaussian noise situations, correntropy is a useful tool for studying higher order statistical moments. With real, complex, and quaternion data, correntropy has been applied [19]. Correntropy is a measure of how similar two random variables are in a neighbourhood dominated by a variable called kernel bandwidth. The kernel bandwidth also affects the "observation window" in which similarity is assessed, which could be an effective strategy to reduce the negative effects of inaccurate data [20], [21], [22]. A common form of correntropy between two variables P and Q is:

$$V_\sigma(P,Q) = E[K_\sigma(P-Q)] \tag{13}$$

Where $k_\sigma(.)$ is a kernel function that satisfies Mercer theory, and $E[K_\sigma(P-Q)]$ denotes the expectation operator of the kernel. It takes advantage of a kernel trick that nonlinearly maps the input space to a higher dimensional feature space. Generally, correntropy is robust against outliers. Due to the specific properties of correntropy, this method is been improved in the CF algorithm. Sun & Luo (2020) [5] proposed a new method for correntropy to compute the similarity of two items or users to achieve insensitive performance to outliers. As mentioned before, mean-centering and z-score could be used to convert individual ratings to a universal scale because each user has their own scale. Then, a new user-based prediction could be obtained by:

$$\widehat{r}_{ui} = \overline{r_u} + \sigma_u \frac{\sum_{s \epsilon N_i(u)} (\frac{1}{\sigma_s} CT_{us}(r_{si} - \overline{r_s}))}{\sum_{s \epsilon N_i(u)} |CT_{us}|} \tag{14}$$

Here, $N_u(i)$ is used to denote the k nearest neighbors of user u which have also rated for item i. Similarly, the item-based prediction could be achieved by:

$$\widehat{r}_{ui} = \overline{r_i} + \sigma_i \frac{\sum_{t \epsilon N_u(i)} (\frac{1}{\sigma_t} CT_{it}(r_{ut} - \overline{r_t}))}{\sum_{s \epsilon N_u(i)} |CT_{it}|} \tag{15}$$

Where $N_i(u)$ denotes the k nearest neighbors of item i which have also rated by user u. Here $CT(_{u,s}) = CT_{us}$ and $CT(_{i,t}) = CT_{it}$ denotes similarity between users u and s and the similarity between items i and t respectively.

2.5 Spark Framework

Apache Spark is an open-source, Java virtual machine (JVM)-based cluster framework that used for big data workloads like Hadoop or Mesos. It utilizes in-memory caching and optimized query execution for fast queries against data of any size. It can also run in the cloud. However parallel computation offers a viable way to alleviate this problem [23]. Spark provides programmers with a resilient distributed dataset (RDD). The RDD is a read-only multiset of data items that is distributed throughout the cluster. As shown in Fig-1, in the first phase, RDD objects are constructed to generate directed acyclic graph (DAG) and operate it. In the second phase, a high-level DAG scheduler is responsible for splitting the DAG into several smaller stages of tasks and setting each task as ready condition. In the next phase, tasks are scheduled and distributed by cluster managers to parallel workers. And in the last stage, workers execute their tasks concurrently in individual threads or cores. Moreover, the RDD is the basis of Spark framework, and the whole operations in RDD support the entire Spark programming. The high-level Spark regards the language scale as the upper implementation of RDD, and it is believed the first system to use an interactive, general, efficient programming language to process a considerable scale of datasets on distributed machines [24].

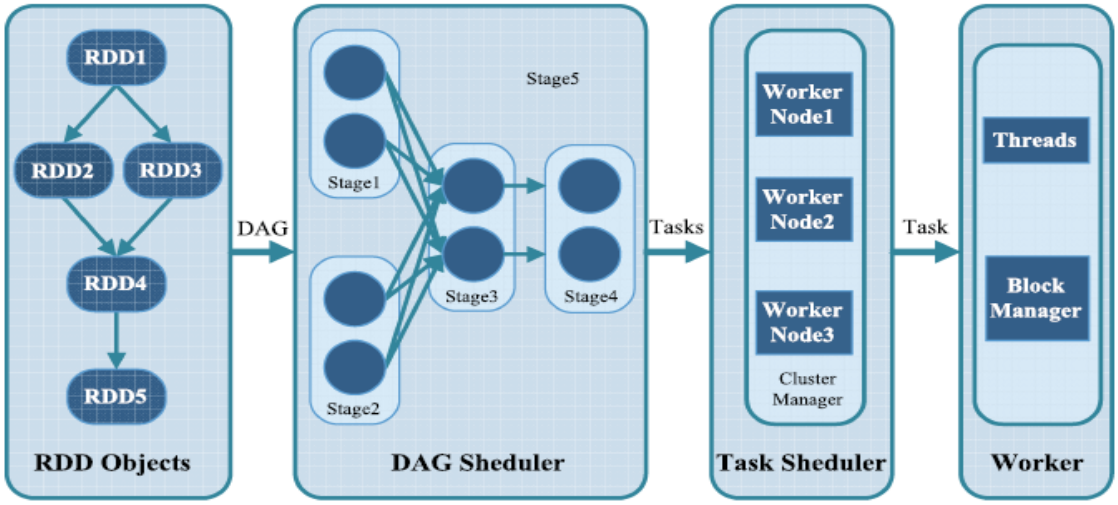

Figure 1: The task scheduling procedure in Spark. [5]

Considering the advantages of Spark, in this article, Sun & Wang [5] developed a parallel personalized recommender system as shown in Figure-2, through the use of a Spark framework.

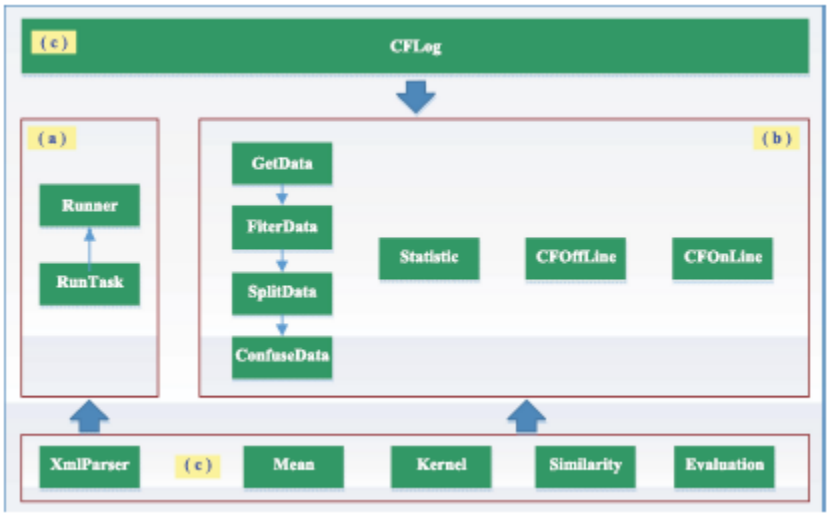

Figure 2: System modules in Spark framework. (a) Basic modules. (b) Task modules. (c) Extra modules. [5]

Here, the basic module mainly handles XML file, calling the running methods in task modules and passing the parameters in the XML file to task modules in the form of map. There are other extra modules used to support the task modules and the basic modules such as evaluation, kernel, mean, similarity, CFLog and XmlParser. In the new proposed Spark Framework, CFOffLine & CFOnLine are the key parts where CFOffLine module calls Similarity module to calculate the corresponding similarity of the data with outliers, and CFOnLine module predicts the item scores based on some algorithms, e.g., k-nearest neighbors (k-NN) algorithm. The advantages of the aforementioned approach could achieve

results with less computational efforts, which outperforms other traditional CF methods in computational time both on stand-alone and distributed computing environments. Specifically, when the size of the dataset is too big, this algorithm could achieve a more obvious advantage with the help of parallelism computing mechanism. Furthermore, compared with other methods, this newly proposed correntropy-based approach may achieve better performance in addressing the datasets with outlier, since correntropy is insensitive to invalid data. When the size of invalid data is relatively big, newly SPARK framework may still perform better.

References

[1] F. Ricci, L. Rokach, and B. Shapira, "Recommender systems: Introduction and challenges," *Recommender Systems Handbook*, p. 1–34, 2015.

[2] A. Werner-Stark and Z. Nagy, "A heuristic method to recommendation systems," *2020 6th International Conference on Information Management (ICIM)*, p. 200–204, 2020.

[3] A. Pal, P. Parhi, and M. Aggarwal, "An improved content based collaborative filtering algorithm for movie recommendations," *2017 Tenth International Conference on Contemporary Computing (IC3)*, p. 1–3, 2017.

[4] R. Ji, Y. Tian, and M. Ma, "Collaborative filtering recommendation algorithm based on user characteristics," *2020 5th International Conference on Control, Robotics and Cybernetics (CRC)*, p. 56–60, 2020.

[5] J. Sun, Z. Wang, X. Luo, P. Shi, W. Wang, L. Wang, J.-H. Wang, and W. Zhao, "A parallel recommender system using a collaborative filtering algorithm with correntropy for social networks," *IEEE Transactions on Network Science and Engineering*, vol. 7, no. 1, p. 91–103, 2020.

[6] D. Lalwani, D. V. L. N. Somayajulu, and P. R. Krishna, "A community driven social recommendation system," *2015 IEEE International Conference on Big Data (Big Data)*, 2015.

[7] K. B. Fard, M. Nilashi, and N. Salim, "Recommender system based on semantic similarity," *International Journal of Electrical and Computer Engineering (IJECE)*, vol. 3, no. 6, 2013.

[8] M. Deshpande and G. Karypis, "Item-based top- n recommendation algorithms," *ACM Transactions on Information Systems*, vol. 22, no. 1, p. 143–177, 2004.

[9] P. Ahlgren, B. Jarneving, and R. Rousseau, "Requirements for a cocitation similarity measure, with special reference to pearsons correlation coefficient," *Journal of the American Society for Information Science and Technology*, vol. 54, no. 6, p. 550–560, Apr 2003.

[10] J. Bobadilla, A. Hernando, F. Ortega, and A. Gutiérrez, "Collaborative filtering based on significances," *Information Sciences*, vol. 185, no. 1, p. 1–17, 2012.

[11] X. Wu, Y. Huang, and S. Wang, "A new similarity computation method in collaborative filtering based recommendation system," *2017 IEEE 86th Vehicular Technology Conference (VTC-Fall)*, 2017.

[12] J. Herlocker, J. A. Konstan, and J. Riedl, "An empirical analysis of design choices in neighborhood-based collaborative filtering algorithms." [Online]. Available: https://link.springer.com/article/10.1023/A:1020443909834citeas

[13] Pero and T. Horváth, "Opinion-driven matrix factorization for rating prediction," *User Modeling, Adaptation, and Personalization Lecture Notes in Computer Science*, p. 1–13, 2013.

[14] P. Kumar, V. Kumar, and R. S. Thakur, "A new approach for rating prediction system using collaborative filtering," *Iran Journal of Computer Science*, vol. 2, no. 2, p. 81–87, 2018.

[15] M. Hofer, "Mean centering," *The International Encyclopedia of Communication Research Methods*, p. 1–3, 2017.

[16] J. S. Breese, D. Heckerman, and C. Kadie, "Empirical analysis of predictive algorithms for collaborative filtering," Jan 1998. [Online]. Available: https://arxiv.org/abs/1301.7363

[17] J. L. Herlocker, J. A. Konstan, A. Borchers, and J. Riedl, "An algorithmic framework for performing collaborative filtering," *Proceedings of the 22nd annual international ACM SIGIR conference on Research and development in information retrieval - SIGIR 99*, 1999.

[18] K. Molugaram and G. S. Rao, "Random variables," *Statistical Techniques for Transportation Engineering*, p. 113–279, 2017.

[19] W. Wang, H. Zhao, and X. Zeng, "Geometric algebra correntropy: Definition and application to robust adaptive filtering," *IEEE Transactions on Circuits and Systems II: Express Briefs*, vol. 67, no. 6, p. 1164–1168, 2020.

[20] W. Liu, P. P. Pokharel, and J. C. Principe, "Correntropy: Properties and applications in non-gaussian signal processing," *IEEE Transactions on Signal Processing*, vol. 55, no. 11, p. 5286–5298, 2007.

[21] B. Chen, X. Liu, H. Zhao, and J. C. Principe, "Maximum correntropy kalman filter," *Automatica*, vol. 76, p. 70–77, 2017.

[22] X. Luo, J. Deng, W. Wang, J.-H. Wang, and W. Zhao, "A quantized kernel learning algorithm using a minimum kernel risk-sensitive loss criterion and bilateral gradient technique," *Entropy*, vol. 19, no. 7, p. 365, 2017.

[23] J. L. Reyes-Ortiz, L. Oneto, and D. Anguita, "Big data analytics in the cloud: Spark on hadoop vs mpi/openmp on beowulf," *Procedia Computer Science*, vol. 53, p. 121–130, 2015.

[24] M. Zaharia, R. S. Xin, P. Wendell, T. Das, M. Armbrust, A. Dave, X. Meng, J. Rosen, S. Venkataraman, M. J. Franklin, and et al., "Apache spark," *Communications of the ACM*, vol. 59, no. 11, p. 56–65, 2016.