Fast Computation of Large-Scale Mixed Effects Models

Robin Yancey* Norm Matloff†

Abstract

This era of Big Data includes many applications of mixed effects modeling, such as large genomics studies, recommender systems, and salary prediction by micro-region and occupation. However, the time complexity for estimation in such models can grow as fast a $n^{1.5}$. Even worse, the computation may not fit into available memory, rendering estimation impossible. Parallel computation may be a remedy, using a model-specific algorithm, but here we present a *generally applicable* approach to the problem, and show computational speedup on various real datasets.

Key Words: Mixed effect models, parallel computation, Big Data, recommender systems

1. Introduction

Algorithms to compute statistically equivalent results on parallel computing platforms are required in a bast number of applications in Big Data. Platforms used for this type of computation include multicore CPUs, GPUs, clusters, and the cloud. The parallel computation term *embarrassingly parallel*, referring to algorithms that can be easily partitioned and then simply recombined, unfortunately often does not apply in the case of statistical methodology. Our solution to this is *chunks averaging*, which exploits the statistical properties of the algorithm by simply averaging results of chunks. This is a technique that we call *Software Alchemy*.

The Software Alchemy method (SA) [2] can be used to turn non-embarrassingly parallel algorithms into statistically equivalent embarrassingly parallel ones, with the equivalence being in terms of equivalent standard errors. SA may also be used as a technique to circumvent memory limitations such as RAM, GPU memory or object size (Section 4.3).

1.1 Background on Recommender Systems

Recommender Systems are used to attempt to predict users' interests given a set of known variables about them. Amazon's product recommender is an example of this type of system, because it uses information about past product purchases to match them to advertisements of potential items of interest to shoppers [1]. Most of the examples of the application of SA shown in the results section are recommender systems.

1.1.1 Data Format of Recommender Systems

The data is in standard format, consisting of (userID, movieID, rating) triples. Most users have not rated most movies, and the goal is to complete the ratings matrix. This process is called the *matrix completion* problem, and the general term for RS tools that combine user and item data is *collaborative filtering*.

*First author's affiliation, University of California, Davis Dept. of Computer Science

†Second author's affiliation, University of California, Davis Dept. of Computer Science

1.1.2 Recommender Systems Mixed Effects Model

Equation 1 shows the general form of the mixed effects model. The mean overall item rating has latent effects of each user α and item β, while ϵ is the error term. We also can include a vector of known covariates. For example, if we apply this model to Netflix movies then the users would be the people who have seen their movies, the items would be the specific movie, while covariates could be the user's age or gender.

$$Y_{ij} = \mu + \gamma' X_i + \alpha_i + \beta_j + \epsilon_{ij} \tag{1}$$

where:

μ is a fixed but unknown overall mean rating,

X_i is an optional known vector of covariates (e.g. age, gender) for user i, and β is a vector of fixed but unknown linear regression coefficients,

α_i is a latent effect for user i,

β_j is a latent effect for item (movie) j, and

ϵ_{ij} is an error term.

1.2 Software Alchemy Overview

Chunks averaging provides a very general method for parallelizing statistical algorithms. For example, the ordinary logistic model could be applied to the data for each chunk, producing estimated coefficients $\widehat{\beta}_{ij}$, where j is the chunk number. The values $\widehat{\beta}_{ij}$ are averaged over all chunks, yielding the final estimated coefficients $\widehat{\beta}_i$. SA is asymptotically efficient, i.e. yields the same standard errors as the (nonchunked) full estimator (Section 3).

2. Implementation

2.1 The partools Package

Our R **partools** package provides useful tools for fast parallel computation, so this was used for the experiments discussed in section 3. Some of the important calls used to apply SA to mixed effects models include **setclsinfo()**, and **distribsplit()**. **setclsinfo()** provides each worker node with a unique ID that can be used to access a specific node or communicate between nodes (with **partools** message passing **ptME**). **distribsplit()** splits a data frame into even row chunks on each cluster node with the same name as the original data.

2.2 partools Software Alchemy Code Example

The code shown below is a full example of how SA can be applied to a mixed effects model. Below we reduce computational time to predict whether a patient will miss an appointment (see Section 3), using the R language and the R **parallel** and **partools** packages. We start by simply creating a **parallel** cluster of 8 with the **makeCluster()** call, setting **setclsinfo()**, and splitting up the data among chunks. Since in these experiments we measure how well SA preserves accuracy, we partition into training and test sets, using a variable **testidxs** to hold the indices of the latter. MLE computation is handled via the **lme4** package.

The training set is split using **distribsplit()**. Each cluster node then holds enough data samples to produce a Maximum Likelihood estimate for the overall data. Finally, when models are called back from the cluster nodes, we can average their results to obtain a model similar to the model that could be produced serially on one node (if time and memory allowed). As shown in Section 3, the accuracy of the predictions and the variance of random

Chunks	Scatter	Train	Pred.	Mean Abs. Pred. Error
full	-	715.135s	285.150s	0.2968
2	1.326s	202.189s	111.490s	0.3047
4	3.000s	197.411s	45.137s	0.2916
8	7.973s	284.732s	38.029s	0.3046

Table 1: Generalized MLE Model, Missed Appointments Data

effects are also close to the serial version. Therefore, this can then be used for further predictions on whether a patient misses an appointment.

```
library(partools)
cls <- makeCluster(8)
setclsinfo(cls)
testidxs <- sample(1:nrow(data),50000)
datatest <- data[testidxs,]
clusterExport(cls,'datatest')
datatrain <- data[-testidxs,]
distribsplit(cls,'datatrain',scramble=T))
clusterEvalQ(cls,library(lme4))
lmeroutC <- clusterEvalQ(cls, lmeroutC <- glmer(No.show ~
   (1|Neighborhood/PatientId) + Gender + Age + Hypertension +
   Diabetes + Alcoholism + Handicap + Scholarship,
   family = binomial, data=datatrain))
predictC <- clusterEvalQ(cls, predict(lmeroutC, data = datatest,
   type="response", allow.new.levels = TRUE))
predictC<-Reduce('+',predictC)/8
```

3. Results and Analysis: Timings and Generated Model Parameters

3.1 Missed Appointments Data

As introduced above, the missed appointments data can be applied with SA. In this case we have a random effect of a patient living in a certain location and fixed effects that include gender, age, alcoholism, hypertension, diabetes, scholarship, and handicap. The following R call can be used to predict whether an appointment is missed by a patient within a given location. The base software used in the MLE context here is **lme4**.

```
glmer( No.show ~ (1|Neighborhood/PatientId) + Gender
+ Age + Hypertension + Diabetes + Alcoholism + Handicap
+ Scholarship, family = binomial, data = dataTrain)
```

As shown in table 1, there is a super-linear speedup using a cluster of 2, an approximately linear speed-up with a cluster of 4, and sub-linear speedup with a cluster of 8. This is a significant improvement compared to serially training the model and computing predictions. There is little additional computation and no loss in prediction accuracy.

Please, note that the scatter column is usually not required (due to the "Leave it There" philosophy.) This means that the data has already been distributed among cluster nodes to save communication time, throughout the given application.

The variance of the random effects is also small, especially in the case of a small cluster size. Table 2 shows the **variance** of the random effects intercepts using SA, as well as the absolute error versus of each effect versus serial for each size cluster.

Chunks	PatientID	Abs. Error
full	0.84016	-
2	0.88642	0.04626
4	0.79317	0.04699
8	0.77563	0.06453

Table 2: Variance of Random Effects, Missed Appointments Data MLE

Tables 3 and 4 shows the fixed effects generated by **lme4** with data undistributed (row 1) vs a cluster of sizes 2, 4, and 8 in the remaining rows. We can see that although the difference is slight, and there is no loss overall prediction accuracy, to maintain estimation accuracy in a few of the fixed effects the cluster size should remain small.

Chunks	Intercept	Gender	Age	Hypertension
-	-1.3905	-0.0360	-0.0065	-0.1084
2	-1.4010	-0.0395	-0.0065	-0.1002
4	-1.3975	-0.0537	-0.0070	-0.0937
8	-1.3657	-0.0428	-0.0068	-0.0910

Table 3: Fixed Effects, Missed Appointments Data MLE

3.2 Instructor Evaluation data

Similar to the patient and hospital data, we see a significant speedup in the Instructor Evaluation data, as shown in tables 5 and 6. Again, with a cluster size of two there is no loss in prediction accuracy and the difference in the variance of random effects is within a few percent.

```
glmer( glout <- glmer(y ~ (1|s) + (1|d) + studage + lectage
+ service , data = datatrain , family = binomial)
```

This Maximum Likelihood model is used to predict whether a student gives a given instructor an positive rating or not, using as covariates number of years in school (studage), amount of lecture time (lectage) and service-course status, as shown in the R call.

Chunks	Diabetes	Alcoholism	Handicap	Scholarship
-	0.0806	0.2877	-0.0636	0.1864
2	0.0678	0.2561	-0.0516	0.1962
4	0.0778	0.2005	-0.0383	0.2096
8	0.0773	0.1839	-0.0179	0.2052

Table 4: Fixed Effects, Missed Appointments Data MLE

Chunks	Scatter	Train	Pred.	Mean Abs. Pred. Error
full	-	220.435s	1.406s	0.3088
2	0.412s	114.405s	1.046s	0.3053

Table 5: Generalized MLE Model, Instructor Evaluation Data MLE

Chunks	Student	Abs. Error	Instructor	Abs. Error
full	0.2390	-	0.5544	-
2	0.2449	0.0059	0.5811	0.0267

Table 6: Variance of Random Effects, Instructor Evaluation Data MLE

3.3 1M MovieLens data

The Movie Lens data [4] is used for predicting the likelihood that a user will give a high or low rating to a given movie (based on their ratings of other movies and other users' ratings.) Using MLE on 250K of the 1M MovieLens set, a data significant speedup occurred between 20 and 50 threads (on a single multicore CPU). Interestingly, the more threads, the more speedup occurred with only very small loss in accuracy, as shown in table 7. This is especially useful when data is already distributed since there is not as much speedup when taking into account the scattering of the data.

Chunks	Scatter	Train	Mean Abs. Pred. Error
full	-	115.786s	0.774
16	3.036	41.922s	0.796
20	3.216	30.429s	0.799
24	4.773	26.129s	0.805
28	5.254	20.446s	0.806
32	6.529	15.508s	0.814
40	7.822	13.738s	0.826
50	9.876	12.349s	0.828

Table 7: MLE Model, Movie Lens Data

The following table shows the **variance** of the random effects intercepts using SA, as well as the absolute error of each effect versus serial for each size cluster (or the difference in the latent effect term in the cluster model vs. serial model term). Accuracy is lesser here.

Chunks	User	Abs. Error	MovieID	Abs. Error
full	0.1251	-	0.3918	-
16	0.1414	0.0163	0.3227	0.0690
20	0.1435	0.0184	0.3163	0.0755
24	0.1442	0.0191	0.3068	0.0850
40	0.1534	0.0283	0.2980	0.0937

Table 8: Variance of Random Effects, Movie Lens Data MLE

3.4 Book Crossings data

The results show a sub-linear speedup effect on the Book Crossings data [5]. In this recommender system example we predict the rating a user gives to a certain book, based on their ratings of other books and other users' book ratings. This can then be used for the industry to choose which books to advertise to a given user. Excellent results here, in both time and accuracy.

Chunks	Scatter	Train	Predict	Mean Abs. Pred. Error
full	-	1114.155	0.455	2.67
2	5.101	685.757	0.455	2.72
4	11.134	423.018	1.173	2.77
8	10.918	246.668	1.470	2.82

Table 9: MLE Model, Book Crossings Data

3.5 Book Crossings data (cluster)

When we ran the same test with the Book Crossings data on a real cluster of separate machines we achieve super linear speedup. In this setting, speedup is again excellent, though not as good as in the multicore case, due to network communication overhead.

chunks	tot. time
2	247.818
4	156.461
8	190.602

Table 10: MLE Model, Book Crossings, Cluster

4. Theoretical Foundations

4.1 In What Settings Is Software Alchemy Fast?

To get an idea of where SA should do well, say we have an algorithm with time complexity $O(n^c)$ for some constant c, and we have r processes. Then theoretically the speedup is calculated as in Equation 2.

$$O[(n/r)^c] = O(n^c/r^c) \tag{2}$$

This means that a speedup of about r^c is obtained using SA. If $c > 1$, we can achieve super-linear speedup, a rarely seen outcome in the parallel computation world.

4.2 Efficiency of Software Alchemy

The SA method is fully efficient if the sample parameter $\hat{\theta}$ can be shown to be approximately normally distributed. In this case it can be shown that the SA estimator based on $\hat{\theta}$ will have the same asymptotic variance as the original.

It is important to note that (in spite of this assumption of i.i.d. data) SA does accommodate fixed effects, as theory shows they may be considered random [2]. [3] demonstrates how this theory could be adapted to the independent-but-not-identically-distributed cases using Method of Moments estimators in mixed effects models (and exactly how one can obtain essentially the same estimators if one treats the fixed effects as random.)

4.3 Memory Issues

Computation may not even fit into available memory, making direct, single-stage estimation impossible. These constraints could include physical memory such insufficient RAM or GPU memory, or software constaints such as the R languages maximum byte size for any single object. The given algorithm's auxiliary data generated during computation may exceed the size of the input data. For example, support vector machine algorithm (SVM) has a $O(n^3)$ time and has $O(n^2)$ space. When used on Big Data this exceeds the memory in many moderately-sized machines common in machine learning (ML) applications

5. Conclusion

Software Alchemy is a method that is easy for statistics users in all research fields to implement. Additionally, it can be applied across a wide range of data sets and is generalizable across many machine learning models. It is especially useful for highly computationally intensive statistics models such as maximum likelihood estimation, and as shown in the results section. we have verified the loss in prediction accuracy is small and difference in fixed and random effects versus the serially produced model is also slight. Although one may need to be somewhat cautious with clusters greater than 2 or 4, the loss in accuracy in latent effects and prediction accuracy is nearly insignificant.

REFERENCES

1. Greg Linden, Brent Smith, and Jeremy York. 2003. Amazon.com Recommendations: Item-to-Item Collaborative Filtering. IEEE Internet Computing 7, 1 (January 2003), 76-80.
2. Matloff, N. (2016). Software Alchemy: Turning Complex Statistical Computations into Embarrassingly-Parallel Ones. Journal of Statistical Software, 71(4), 1 - 15. doi:http://dx.doi.org/10.18637/jss.v071.i04
3. Matloff, N. (2013). A New Framework for Random Effects Models, JSM 2016.
4. F. Maxwell Harper and Joseph A. Konstan. 2015. The MovieLens Datasets: History and Context. ACM Transactions on Interactive Intelligent Systems (TiiS) 5, 4, Article 19 (December 2015), 19 pages. DOI=http://dx.doi.org/10.1145/2827872
5. Improving Recommendation Lists Through Topic Diversification, Cai-Nicolas Ziegler, Sean M. McNee, Joseph A. Konstan, Georg Lausen; Proceedings of the 14th International World Wide Web Conference (WWW '05), May 10-14, 2005, Chiba, Japan.