Scalable Non-Linear Logistic Regression in R with an Application to the Census of Agriculture

Jake Abernethy
USDA NASS
Jake.Abernethy@usda.gov

Disclaimer

- The findings and conclusions of this presentation are those of the authors and should not be construed to represent any official USDA or U.S. Government determination or policy.

Contents

- Census of Agriculture
- Research Question
- Challenges
- R Implementations
- Results
- Conclusion

Census of Agriculture

- Conducted every 5 years (years ending in 2 and 7) using list-based frame (the census mailing list or CML)
- Count of all U.S. agricultural operations
  - Any place from which $1,000 or more of agricultural products were produced and sold or normally would have been sold during the year
- Only source of uniform, comprehensive agricultural data for every county or county equivalent in the U.S.
- Leading source of information on characteristics of people operating farms

Census of Agriculture

- How are the counts from the census produced?
- Information from two sources are used
  - COA
  - June Area Survey (JAS)
- June Area Survey
  - Area-frame based
  - Conducted annually via in-person interviews
  - Segments of land sampled
  - Sampled segments divided into tracts representing unique land operating arrangements
  - Measures the incompleteness of the CML

Census of Agriculture

- Statistical Assumptions
  - Individual's response to COA and JAS are independent, given information collected on the forms
  - Probability of response is a function of variables measured on COA / JAS forms
- One more assumption:
  - Farms / nonfarms are perfectly distinguishable
  - Not true in practice
  - NASS accounts for this
  - Assumed here for ease of exposition

Census of Agriculture

- Notation:
  - $\mathcal{C}$: Farms that returned census form
  - x_i: A farm's data, collected by census or JAS
  - c_i: Did the farm return the census form?
    - 1: Yes
    - 0: No
  - j_i: Did the farm return the JAS form?
    - 1: Yes
    - 0: No

Census of Agriculture

- Example: How do we use the COA and JAS to count the number of farms in the US?
- Model, given assumptions [2]

$$N_{farms} = \sum_{i \epsilon \mathcal{C}} p_i^{-1}(c_i = 1 | j_i = 1, x_i)$$

- Obviously, the model of the probabilities has an impact on the final estimate

Census of Agriculture

- Procedure in words
  1. Make matched dataset consisting of:
     a. All JAS tracts
     b. COA records overlapping JAS tracts (not all census records)
  2. Using the data from step 1, model probability of census response
  3. Use model to fit probabilities to all responding census farms (all census records)
  4. Get estimate by summing the inverse of these probabilities for every census record

Research Question

- All estimates produced by the COA depend on the response probabilities
- NASS uses linear logistic regression to model these probabilities
- How realistic is the linearity assumption?
- Test non-linear probability models to check
- Metric:
  - Divide matched data into training and validation set
  - Measure performance on validation set
  - Will examine affect on estimates (e.g. N_{farms}) in future study

Scalability Challenge

- Matched dataset for modeling is moderately sized
  - $N = 40{,}000$, $p = 55$
- This is still large enough to stress available resources
- Models must be run many times
  - Model selection / Cross Validation
- Algorithms must:
  - Be able to run quickly
    - Sublinear time complexity
  - Be able to fit in available memory
    - Linear or sublinear space complexity

USDA

AGRICULTURE COUNTS

Scalability Challenge

- Scalability requirements eliminate some algorithms
  - $O(N^2)$ space / time algorithms
  - Kernel methods
  - Nearest neighbors
  - Anything that inverts a large matrix
- Scalable algorithms typically use MBSGD
  - MBSGD: Mini Batch Stochastic Gradient Descent
  - Subsample of data is used to calculate gradient at each step
  - Unbiased estimate of true gradient
  - Complexity determined by batch size

R Implementations

- Some algorithms that employ MBSGD
  - Linear Logistic Regression
  - Extreme Gradient Boosting
  - Deep Neural Network
- Libraries
  - LiblineaR [3]
  - xgboost [4]
  - ANN2 [5]
- See next few slides for basic implementations
- See algorithm references for more detailed use

R Implementations

- Linear Logistic Regression: Train Model
  - model = LiblineaR(X_train, y_train, type = 6, cost = 8)
- Linear Logistic Regression: Make Prediction
  - guess = predict(model,X_validate)
- Linear Logistic Regression: Check Accuracy
  - acc = length(which(guess$predictions==y_val))/length(y_val)

R Implementations

- Extreme Gradient Boosting: Train Model
  - parms = list(max_depth = 2, eta = 0.1, nthread = 6,
    objective = "binary:logistic", eval_metric = "auc" ,
    subsample = 0.5)
  - model = xgboost(data = X_train, label = y_train, params = parms)
- Extreme Gradient Boosting: Make Prediction
  - guess = predict(model,X_validate)
  - guess = ifelse(guess>0.5,1,0)
- Extreme Gradient Boosting: Check Accuracy
  - acc = length(which(guess==y_validate))/length(y_validate)

R Implementations

- Artificial Neural Network: Train Model
  - model = neuralnetwork(X_train, y_train, hidden.layers=10, loss.type = "log", activ.functions = "relu", optim.type = "adam", learn.rates = 1e-05, L2 = 1, adam.beta1 = 0.9, adam.beta2 = 0.999, n.epochs = 4000, batch.size = 2048, val.prop = 0.1)
- Artificial Neural Network: Make Prediction
  - guess = predict(model,X_validate)
- Artificial Neural Network: Check Accuracy
  - acc=length(which(guess[[1]]==y_validate))/length(y_validate)

R Implementation: Full Workflow

- Full workflow example (using xgboost as example)
  1. Get data
     - Training and validation
  2. Choose parameters using validation set
  3. Check Accuracy

R Implementation: Get Data

- X_train <- read.csv(“File")
- X_val <- read.csv(" File ")
- y_train <- read.csv(“File")
- y_val <- read.csv(“File")

- X_train = X_train[,-1]
- X_val = X_val[,-1]
- y_train = y_train[,-1]
- y_val = y_val[,-1]

- X_train = data.matrix(X_train)
- X_val = data.matrix(X_val)
- y_train = unlist(y_train)
- y_val = unlist(y_val)

- maxs = apply(X_train,2,max)
- mins = apply(X_train,2,min)
- X_train = scale(X_train,center=mins,scale = maxs-mins)
- X_val = scale(X_val,center=mins,scale = maxs-mins)

R Implementation: Choose Parameters

- md = 2:10
- eta = seq(0.1,1,0.1)
- cvpairs = expand.grid(md,eta)
- acc = rep(0,nrow(cvpairs))

- for(i in 1:nrow(cvpairs)){
- parms <- list(max_depth = cvpairs[i,1], eta = cvpairs[i,2], nthread = 6,
- objective = "binary:logistic", eval_metric = "auc" , subsample = 0.5)
- mod1 = xgboost(data = X_train, label = y_train, params = parms, nrounds = 5000, verbose = 0)
- guess = predict(mod1,X_val)
- guess = ifelse(guess>0.5,1,0)
- acc[i]=length(which(guess==y_val))/length(y_val)
- }

R Implementation: Check Accuracy

- which(acc==max(acc)) #1
- max(acc) #0.8827336
- cvpairs[1,] # Depth = 2 eta = 0.1

Results

Model	Validation Accuracy
Linear	85.56
Linear with Interactions	85.63
Extreme Gradient Boosting	88.27
Artificial Neural Network	84.58

Conclusion

- Conclusions
  - Extreme gradient boosting performs the best
  - Linear model is still close behind
  - Demonstrates linearity is not a bad assumption
  - This is good! Lends additional credence to results on previous censuses

Future Work

- Try more non linear models
  - For example, approximate kernel logistic regression using random Fourier features [6] and MBSGD
- Try more census years
  - 2012 was used here
  - Also have data for 2017
- Compare actual estimated totals
  - The validation accuracy is not very sensitive to the choice of model
  - Does the same hold for the estimate of N_{farms}?

References

- [1]: Young, Linda J., Andrea C. Lamas, and Denise A. Abreu. "The 2012 Census of Agriculture: a capture–recapture analysis." *Journal of Agricultural, Biological and Environmental Statistics*22.4 (2017): 523-539.
- [2]: Alho, Juha M. "Logistic regression in capture-recapture models." *Biometrics* (1990): 623-635.
- [3]: Fan, Rong-En, et al. "LIBLINEAR: A library for large linear classification." *Journal of machine learning research* 9.Aug (2008): 1871-1874.

References

- [4]: Chen, Tianqi, and Carlos Guestrin. "Xgboost: A scalable tree boosting system." *Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining*. ACM, 2016.
- [5] : https://cran.r-project.org/web/packages/ANN2/ANN2.pdf
- [6] : Rahimi, Ali, and Benjamin Recht. "Random features for large-scale kernel machines." *Advances in neural information processing systems*. 2008.


Thank
you!