Clustering Linkage functions in Record Linkage

David Zimmerman
Kyongche Kang

Abstract

Our project focuses on exploring supervised and unsupervised learning algorithms for record linkage problems. We explored logistic regression as supervised and K-means clustering for unsupervised algorithms using Jaro-Winkler similarity metric. We developed a hybrid model by combining logistic regression and hierarchical clustering that has advantages of both supervised and unsupervised algorithms.

Introduction

Record linkage is to determine which records in a data set that refer to the same entity across different data sources. There are two ways to approach the record linkage problem: deterministic and probabilistic record linkage. One of the widely used method is probabilistic record linkage. It calculates weights for each identifier based on its estimated ability to correctly identify a match or a non-match, and estimates probabilities of pairs of records being a match. A match or non-match is determined based on the probabilities above certain threshold. In many cases, list of records are bound to contain some kind of typographical variations or errors. Deterministic approach compares exact matching of fields and this fails to match many pairs that are mathes but contain errors. Thus we approach the record linkage problem with probabilistic method. Two types of learning algorithms exist in classifying match and non-match - supervised and unsupervised learning algorithms. We are interested in comparing the two algorithms in record linkage problem to see which one performs better. Then we extended this to developing a hybrid model, which encompasses the advantages of two algorithms to effectively link duplicate records without compromising accuracy.

Data

The dataset we used in our problem is RLdata500, which is included in RecordLinkage package in R. It consists of 500 records of data, and the following fields: first name_1, first name_2, last name_1, last name_2, birth year, birth month and birth day. This dataset provides a true label of which pairs of records are match or non-match. There are 50 known matches. We chose this dataset, because it is a manageable size for record linkage problem. There is a similar dataset with larger list, RLdata10000. We took a subset of 1000 matching records and 1000 non-matching records. We used this subset in order to compare how varying number of matches and non-matches in the dataset affects modeling and error rate.

Methods

Supervised vs. Unsupervised

In classification problem, there exists two kinds of algorithms: supervised and unsupervised learning. The distinction is drawn from the way the algorithm classifies the data. In supervised learning, the classification is done by training the model on prelabeled data. With a trained model, it is then able to make inference for unseen instance. On the other hand, unsupervised learning seeks out similarity between pieces of data in order to determine whether they can be characterized as forming a group without training on the data first. It is able to discover hidden structure of the data.

Advantage of supervised learning is that because it is trained on the data, it is better than unsupervised learning in terms of accuracy. However, it requires training data and is prone to overfitting. Unsupervised learning is useful in practice, where labels do not necessarily come with the data. They are often hard and expensive to obtain. Learning directly from data is one of the strengths of unsupervised learning.

Logistic Regression

Our first approach to record linkage was to use logistic regression. This is a supervised learning algorithm where given a set of labels and inputs as training data, it is able to predict probabilities of pairs of records being a match or non-match. We are given a true label of matching records. In order to model the data, we constructed a pair-wise comparison of each field in the data. We used Jaro-Winkler string comparator metric that gives scores on agreement between two strings. This metric accounts for the lengths of the two strings and partially accounts for the types of errors - insertions, omissions, or transpositions that people make in typing information on the computer. We first constructed 500 by 500 matrix that contains Jaro-Winkler pairwise comparison scores for each of the field. The ith record in the list is compared with jth record in the list, thus creating a triangular matrix. The scores are then extracted into a vector, which gave 124,750 pairs of record comparisons for each of the 7 fields. We fitted logistic regression, with response variable as true label of being a match(1) or non-match(0) and Jaro-Winkler scores for first name_1, last name_1, birth year, birth month and birth day. Since first name_2 and last name_2 were missing in many of the records, they were not included in the model.

$$\log\left(\frac{p}{1-p}\right) = \beta_0 + \beta_1(\text{first name}) + \beta_2(\text{last name}) + \beta_3(\text{birth year}) + \beta_4(\text{birth month}) + \beta_5(\text{birth day})$$

K-means Clustering

Clustering is a unsupervised learning to group observations into clusters in such a way that observations in the same cluster are similar to each other than those in different clusters. We used K-means clustering algorithm as unsupervised learning. With initial number of clusters, k and points of centers of each cluster, the algorithm iterates to assign each observation to the cluster whose mean yields the least within-cluster sum of squares. Then, the positions of

centers are updated with the newly assigned observations within a cluster. The algorithm converges when there is no newly assigned observation.

For record linkage problem, we used K-means algorithm on Jaro-Winkler pairwise scores. Each comparison is a point in n-dimensional space before clustering. We assigned 3 clusters: 'link', 'non-link', and 'possibly link'. According to Elfeky who proposed this method, the comparison vector $c_{i,j}$ is defined as comparison of records r_i r_j that has a value 0 if two records are a perfect match and 1 if they are completely different. Hence, the center of cluster nearest to the origin represents the 'link' cluster and that of the cluster furthest from the origin represents 'non-link'. The remaining cluster represents 'possibly link'.

Hierarchical Clustering

The drawback of K-means clustering is that the clustering assignment depends on the initial choice of the centers. Hence, the results are not consistent and the accuracy varies with different centers. Hierarchical clustering is another unsupervised learning, which takes pairwise dissimilarities between i and j to produce a sequence of clustering assignments, from n assignment, each observation being a separate cluster, to k=1 assignment. At each level, two clustering assignments that have the smallest dissimilarity scores will be combined. The final clustering assignment depends on the cut-off value for the maximum number of distance we would allow for each cluster.

There are different linkage functions that returns dissimilarity scores between the two cluster assignments. Single linkage returns the dissimilarity between two clusters as the smallest distance between two points in opposite groups. Complete linkage is the opposite of single linkage, in which dissimilarity is the largest distance between two points in opposite groups. Average linkage takes the average dissimilarity over all points in opposite groups.

We combined supervised and unsupervised learning by clustering on probabilities of a match generated from logistic regression. The probabilities of a match we took as a measure of similarity, thus in order to cluster we took one minus the probability to create a distance matrix. With this matrix we tried three different types of hierarchical clustering: single linkage, complete linkage, and average linkage. Each technique starts will all points as their own clusters. At each step the algorithm looks for the most similar points or groups to join. Single linkage looks for the minimum distance between any point in one cluster and any point in another cluster. Complete linkage looks for the point in each cluster that is greatest away and then joins the clusters with the smallest maximum distance. Average linkage looks for the smallest average distance between clusters. All three algorithms were applied to the distances between records and looked at different cut points to determine how close points had to be in order to be joined.

Results

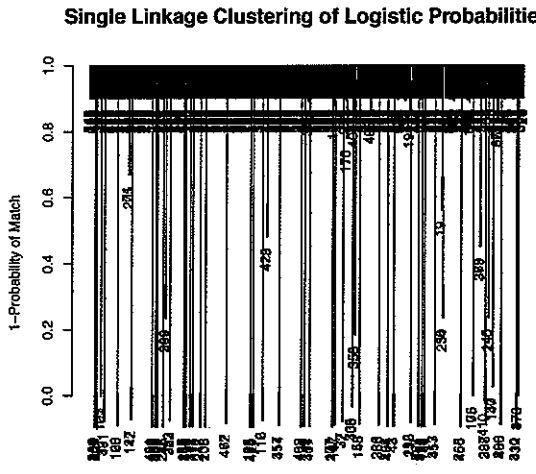
Our main results come from the RLdata500 because the model converged and we had numerous significant variables. All variables except the first name were significant at the 0.05

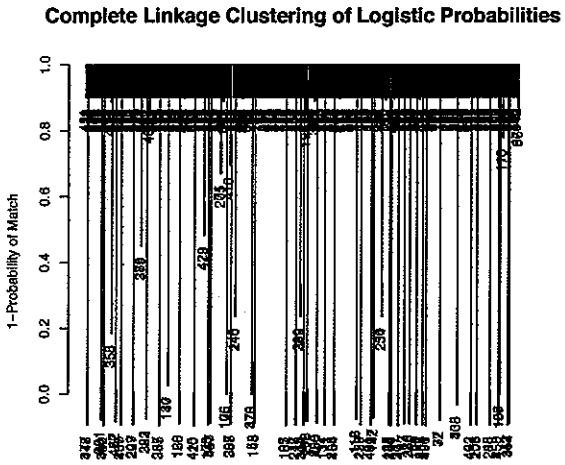
level. This fits intuition because the first names tend not to be as unique and they do not provided a lot of diagnostic help in differentiating people. Many people share first names but little or no other information so it makes sense that the Jaro-Winkler score would not aid in separating matches from non-matches.

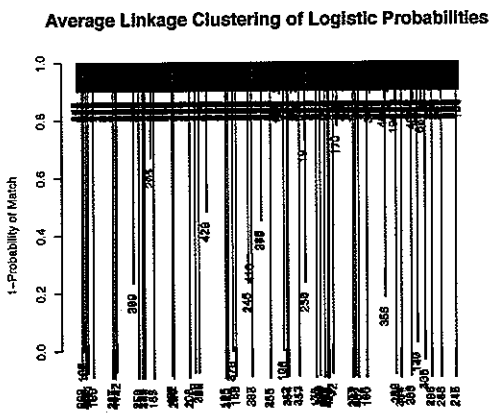
Logistic Regression on RLdata500

	Estimate	Std. Error	z value	Pr(>|z|)
(Intercept)	-76.5686	26.2199	-2.92	0.0035
fname_c1.comp	32.3748	20.0866	1.61	0.1070
lname_c1.comp	33.0743	13.8282	2.39	0.0168
by.comp	7.6162	2.5007	3.05	0.0023
bm.comp	4.1918	1.6358	2.56	0.0104
bd.comp	5.7277	1.6509	3.47	0.0005

With this model we predicted the probability of a match for each pair of records. By taking one minus the probability we created a distance and then began clustering. The dendogram plots show how the records that are actually pairs fall out nicely. Essentially all pairs of records that have a probability of matching that is greater than 0.05 are matching records. This is a fairly robust finding across all three types of linkages.

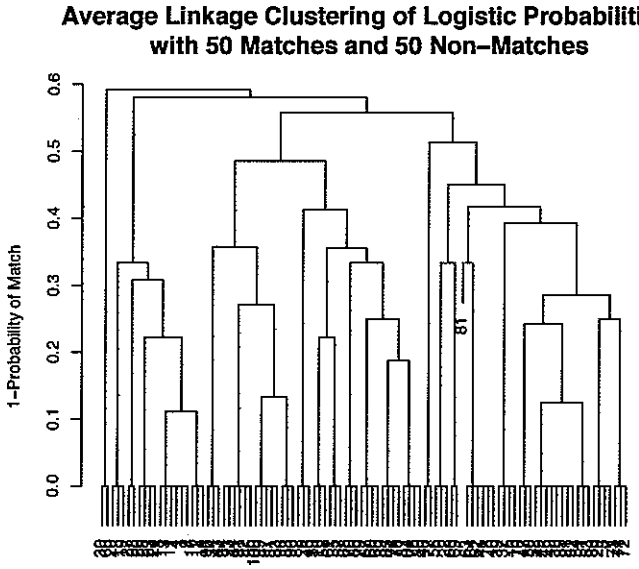






All three linkage functions seem to be picking up on fairly similar patterns. The black two black bars across the top of the dendogram is that there are about 124,750 comparisons made and in the plot they are overlapping because there is not enough space to see all of them individually. Looking at tables of the predicted versus truth it is clear that all three of the algorithms are making few mistakes. One remarkable feature of this clustering is that joining records that have a probability of matching of 0.2 actually helps the performance. It is as if the distrubtion for probability of matching has been shifted down so that records that should have a 50/50 chance of matching are actually predicted to have approximately zero percent chance of matching.

In our model with 50 matches and 50 non-matches the clustering algorithm does not work because our logistic regression model has no significant variables. It is guessing pseudorandomly for each pairwise comparison whether it is a match or not. Thus the dendograms look very strange and it is clear that records are not being put into pairs that are actually matching. Below in the dendogram for average linkage at the bottom of the tree may or the groups are sets of three records. Based on our data there are not groups of three or more records that are links so this algorithm is clearly not doing well.



When we expanded our sample to 1000 matches and 1000 non-matches the algorithm for clustering would not work because we had too much data. The logistic regression model converged and had the same significant variables with the addition of first name. One method around the too much data would be to block on birth year but have a dynamic range for each individual person. For example we could take every person who is born in 1950 and compare them only to people born from 1945 to 1955. This would dramatically reduce the computation needed to complete this task. With this approach we could no longer cluster because we do not have a total distance matrix between points, but it could run much more quickly for our other procedures.

Logistic Regression on 1000 Matches and 1000 Non-Matches

	Estimate	Std. Error	z value	Pr(>|z|)
(Intercept)	-93.8228	14.3225	-6.55	0.0000
fname_c1.comp	43.7298	11.7691	3.72	0.0002
lname_c1.comp	28.2785	5.5200	5.12	0.0000
by.comp	18.2294	2.2501	8.10	0.0000
bm.comp	4.1631	0.5135	8.11	0.0000
bd.comp	5.2696	0.6192	8.51	0.0000

Using a very simplistic measure of accuracy, the percentage of matches or non-matches estimated correctly, we can order our methods to rate which one is best. In general the accuracies are very high because of the large percentage of non-matches in the data. If a classifier were to only guess non-match it would have a very good accuracy level without needing to do any predictive computation. Our worst method was K-means with an accuracy rating of 0.9906. All other methods were at least 0.9999 accurate where complete and average linkage were the best followed up by single linkage and logistic regression.

K-means

RLdata500

	Link	Possible link	Non-link
Match	10866	0	1141466
Non-match	989	0	11

Accuracy: 0.990569

Subset

	Link	Possible link	Non-link
Match	115	0	73206
Non-match	188	0	71

Accuracy: 0.997472

The accuracy of the cluster methods goes up as the cutoff on probability of a match is reduced. At a cutoff of 0.8 which intuitively seems like a good value the clustering performs reasonably well, but does not produce the best results. In fact the model performs better up until probability of a match is 0.05. In general the models are quite good where they get between 43 and 50 of the true matches with a couple of false positives and false negatives. The differences between the three methods are quite small. In general the single linkage clustering performs just a little bit worse by having more false positives or false negatives.

Complete Linkage (cut-off=0.8)

Truth / Predicted	Link	Non-link
Match	50	0
Non-Match	4	124696

Accuracy: 0.9999679

Average Linkage (cut-off=0.8)

Truth / Predicted	Link	Non-link
Match	50	0
Non-Match	4	124696

Accuracy: 0.9999679

Logistic Regression (cut-off=0.8)

Truth / Predicted	Link	Non-link
Match	44	6
Non-Match	2	124698

Accuracy: 0.9999359

An ROC plot shows how similar the three linkage methods are. Here the endpoints are fixed at (0,0) and (1,1) and the curve between these points indicates how good the model is. Looking at the accuracy ratings and the tables at different cutoffs is the only way to distinguish which of the methods is best. Our ROC curve goes from 0 to 1 for the cutoff value by 0.05. All of these points help to push test the model at many different levels. The general shape of the ROC curves is that of an ideal predictor where the area under the curve is maximized.

Conclusions

We approached record linkage problem with supervised and unsupervised learning algorithms. We used logistic regression for supervised and K-means clustering algorithm for unsupervised. Although logistic regression performed better than K-means in terms of accuracy, we expected this result as logistic regression is trained on true labels. To overcome limitations of supervised and unsupervised learning algorithms, we proposed combining supervised and unsupervised learning algorithms to create a hybrid algorithm, which takes advantages of both algorithms. We developed a hybrid model of hierarchical clustering by constructing dissimilarities matrix of records i and j with predicted probabilities from logistic regression and clustered with hierarchical clustering. This model produced more accurate results in recognizing duplicate records than logistic regression or K-means clustering.

Acknowledgement

We would like to thank Professor Fienberg and Professor Nugent for this great course and Sam for helping with brainstorming project ideas.

References

Christen, P. (2012). Data Matching: Concepts and Techniques for Record Linkage, Entity Resolution, and Duplicate Detection. New York: Springer.

Dunn, Halbert L. "Record Linkage". American Journal of Public Health 36 (12): pp. 1412–1416, December 1946.

Elfeky, M.G., Elmagarmid A.K., and Verykios, V.S. Tailor: A record linkage tool box. In Proc. of the 18th Intl. Conference on Data Engineering, pages 17–28, San Jose, California, February 2002.

Sariyar, M. and Borg, A. The RecordLinkage Package: Detecting Errors in Data, The R Journal Vol. 2/2, December 2010

Appendix

R-code

```
#K-means clustering
rpairs=compare.dedup(RLdata500, identity=identity.RLdata500, blockfld=list(1,3,5,6,7))
result=classifyUnsup(rpairs,method="bclust")
summary(result)
rpairs=compare.dedup(half.big.50.50, identity=id.half.big.50.50, blockfld=list(1,3,5,6,7))
result=classifyUnsup(rpairs,method="bclust")
summary(result)

# Logistic Regression
 lreg <- glm(Match ~ . , family = "binomial", data = RL500JW[,-c(2,4,8,9)])
 summary(lreg)
 xtable(summary(lreg))
 predictions <- predict(lreg, RL500JW[,-c(2,4,8,9)])
 probs <- exp(predictions)/ (1 + exp(predictions))   # Use this to cluster
 eset.values <- ifelse(probs > 0.8, 1,0)

###### Small sample equal proportions
names(RL500.50m.50u)
lreg.50.50 <- glm(identity.RL500.50m.50u ~ ., family = "binomial", data = RL500.50m.50u[,-
 c(2,4,8,9)])
dim(RL500.50m.50u[,c(2,4,8,9)])
summary(lreg.50.50)
preds <- predict(lreg.50.50, RL500.50m.50u[,-c(2,4,8,9)])
probs.50.50 <- exp(preds) / (1 + exp(preds))    # Use this to cluster

singleclust <- hclust(distances, method = "single")
completeclust <- hclust(distances, method = "complete")
averageclust <- hclust(distances, method = "average")

singleclust.50.50 <- hclust(distances.50.50, method = "single")
completeclust.50.50 <- hclust(distances.50.50, method = "complete")
averageclust.50.50 <- hclust(distances.50.50, method = "average")

pdf("single_500.pdf")
plot(singleclust, main = "Single Linkage Clustering of Logistic Probabilities", ylab= "1-Probability
```

```
 of Match", cex.main = 1.5)
dev.off()
pdf("compl_500.pdf")
plot(completeclust, main = "Complete Linkage Clustering of Logistic Probabilities", ylab= "1-
 Probability of Match", cex.main = 1.5)
dev.off()
pdf("ave_500.pdf")
plot(averageclust, main = "Average Linkage Clustering of Logistic Probabilities", ylab= "1-
 Probability of Match", cex.main = 1.5)
dev.off()
pdf("single_100.pdf")
plot(singleclust.50.50, main = "Single Linkage Clustering of Logistic Probabilities\n with 50
 Matches and 50 Non-Matches" , ylab= "1-Probability of Match", cex.main = 1.5 )
dev.off()
pdf("comp_100.pdf")
plot(completeclust.50.50, main = "Complete Linkage Clustering of Logistic Probabilities\n with
 50 Matches and 50 Non-Matches", ylab= "1-Probability of Match", cex.main = 1.5)
dev.off()
pdf("ave_100.pdf")
plot(averageclust.50.50, main = "Average Linkage Clustering of Logistic Probabilities\n with 50
 Matches and 50 Non-Matches", ylab= "1-Probability of Match", cex.main = 1.5)
dev.off()

cutoffs <- seq(from= 0.05, to = 0.95, by = 0.05)
results <- matrix(NA, nrow = 6, ncol = length(cutoffs))
for (i in 1:length(cutoffs)) {
 cutoff <- cutoffs[i]
 print(cutoff)
 single.labels <- cutree(singleclust, h = cutoff)
 complete.labels <- cutree(completeclust, h = cutoff)
 average.labels <- cutree(averageclust, h = cutoff)

 single.matches <- calc.pcs(as.data.frame(single.labels), type = "e", add.comp = F)
 # print(table(Match,single.matches[,1]))
 single.table <- table(Match, single.matches[,1])
 print("single")
 print(single.table)
 print(sum(Match == single.matches[,1]) / length(Match))
 # Calculating sensitivity and Specificity
 results[1,i] <- single.table[2,2] / (single.table[2,1] + single.table[2,2])
 results[2,i] <- single.table[1,1] / (single.table[1,2] + single.table[1,1])

 complete.matches <- calc.pcs(as.data.frame(complete.labels), type = "e", add.comp = F)
 # print(table(Match,complete.matches[,1]))
 complete.table <- table(Match, complete.matches[,1])
 print("complete")
 print(complete.table)
 print(sum(Match == complete.matches[,1]) / length(Match))
 results[3,i] <- complete.table[2,2] / (complete.table[2,1] + complete.table[2,2])
 results[4,i] <- complete.table[1,1] / (complete.table[1,2] + complete.table[1,1])
```

```
 average.matches <- calc.pcs(as.data.frame(average.labels), type = "e", add.comp = F)
 # print(table(Match,average.matches[,1]))
 average.table <- table(Match, average.matches[,1])
 print("average")
 print(average.table)
 print(sum(Match == average.matches[,1]) / length(Match))
 results[5,i] <- average.table[2,2] / (average.table[2,1] + average.table[2,2])
 results[6,i] <- average.table[1,1] / (average.table[1,2] + average.table[1,1])
}


pdf("ROC_curve_ends.pdf")
plot(1-results[2,], results[1,], ylim = c(0.8,1), main = "ROC Curves by Linkage Functions", type =
 "b", xlab = "1-Specificty", ylab = "Sensitivity", col = "blue", pch = 3, lwd = 2, cex.main = 1.5)
lines(1-results[4,],results[3,], type = "b", col = "darkgreen", pch = 10, lwd = 2)
lines(1-results[6,],results[5,], type = "b", col = "darkred", pch = 6, lwd = 2)
legend("bottomright", legend = c("Single", "Complete", "Average"), col = c("blue", "darkgreen",
 "darkred"), pch = c(3,10,6))
dev.off()

distances <- matrix(NA, nrow = 500, ncol = 500)
lowerTriangle(distances) <- 1-probs
diag(distances) <- 1
distances <- as.dist(distances)

### Creating Subset of big
sub <- sample(seq(1,2000),1000)
sub.row.matches <- row.matches[sub,]
sub.id.matches <- id.matches[sub]

subset <- sample(seq(1,8000), 1000)
non.match.1000 <-row.non.matches[subset,]
id.non.match.1000 <- id.non.matches[subset]

half.big.50.50 <- rbind(sub.row.matches, non.match.1000)
id.half.big.50.50 <- c(sub.id.matches, id.non.match.1000)

JW.half.big.50.50 <- calc.pcs(half.big.50.50)
dim(JW.half.big.50.50)
JW.id.half.big.50.50 <- exact.match.col(id.half.big.50.50,1)
length(JW.id.half.big.50.50)
dist.50.50 <- matrix(NA, nrow = 2000, ncol = 2000)
lowerTriangle(dist.50.50) <- probs.big.50.50


### Using Logistic to predict matches for large sample
lreg.large.sample <- glm(JW.id.half.big.50.50 ~ ., family = "binomial", data = JW.half.big.50.50[,-
 c(2,4,8,9)])
xtable(summary(lreg.large.sample))
```