BRADLEY-TERRY AND THE RANKING LASSO: AN R VINAIGRETTE

ROGER KOENKER

Abstract. A penalized likelihood method for estimating the Bradley-Terry model for paired comparison data is considered and a Mosek implementation is developed.

1. Introduction

In accordance with Stigler's law of eponymy, the Bradley and Terry (1952) model for ranking competitors based on paired comparisons was first proposed by Zermelo (1929) for rating chess players.[1] In the simplest setting we have n players of ability, $\alpha_1, \alpha_2, \ldots, \alpha_n$ who meet in pairs; the probability that a player i defeats a player j is given by.

$$\pi_{ij} = \alpha_i/(\alpha_i + \alpha_j).$$

With a sufficient accumulated history of play, the α's can be estimated by maximum likelihood. Since n can be rather large there is an inducement to regularize and many proposals have been made. Among these the proposal of Masarotto and Varin (2012) seems particularly intriguing since it purports to group the estimated abilities into a few equivalence classes using a variant of ℓ_1 penalization. The ranking penalty has already been used earlier for clustering by Hocking et al. (2011), and in the context of quantile regression panel models by Gu and Volgushev (2019).

2. Some Convex Optimization

It is convenient to reparameterize abilties so $\theta_i = \log \alpha_i$ and π_{ij}, becomes,

$$\pi_{ij} = \frac{1}{1 + \exp(-(\theta_i - \theta_j))}$$

and to write the (logistic) log likelihood for m binary outcomes, $y_1, y_2, \ldots, y_m$, as,

$$\ell(\theta|y) = \sum_{k=1}^{m} y_k \log(h_\theta(x_k)) + (1 - y_k) \log(1 - h_\theta(x_k))$$

where $h_\theta(x) = 1/(1 + \exp(-\theta^\top x_k))$, x_k is an n vector with ith element 1, and jth element -1, and other elements 0. The "ranking lasso" of Masarotto and Varin (2012) proposes to estimate the vector, $\theta = (\theta_1, \ldots, \theta_n)$ by minimizing,

$$R(\theta) = -\ell(\theta|y) + \lambda ||D\theta||_1,$$

where $||D\theta||_1 = \sum_{i<j} |\theta_i - \theta_j|$. There is an extensive literature on ℓ_1 penalized logistic regression, notably Koh et al. (2007), but typically such methods are designed to shrink some

August 22, 2021. Thanks to Jiaying Gu, Ryan Tibshirani, Addison Hu and especially Michal Adamaszek for helpful suggestions. They bear no responsibility for errors, egregious or otherwise.

[1] Useful background on all this is provided by David (1988), Cattelan (2012) and Glickman (2013).

or all of the coefficients toward zero. In contrast the ranking lasso tries to reduce *differences* in the coefficients and thereby tries to identify groups of coefficients that take the same value. Masarotto and Varin (2012) propose an EM approach to minimizing $R(\theta)$; I would like to explore an alternative interior point approach that relies on Mosek ApS (2021*a*).

Since R is a convex function the problem can be reformulated to minimize a linear function subject to convex cone constraints. This can be written as,

$$\min\{\sum_{k=1}^{m} t_k + \lambda r \mid t_k \in \mathcal{T}_k, k = 1, \ldots, m, r \geq ||D\theta||_1\},$$

where $\mathcal{T}_k = \{t | t \geq y_k \log(h_\theta(x_k) + (1 - y_k) \log(1 - h_\theta(x_k)\}$. In Mosek terminology the $\mathcal{T}_k$ are built with exponential cones. An exponential cone in $\mathbb{R}^3$ is the closure of the set of points satisfying,

$$K = \{x \in \mathbb{R}^3 | x_1 \geq x_2 \exp(x_3/x_2), x_1 > 0, x_2 > 0\}$$

or equivalently,

$$K = \{x \in \mathbb{R}^3 | x_3 \leq x_2 \log(x_1/x_2), x_1 > 0, x_2 > 0\}.$$

For the logistic function $f(x) = \log(1 + e^x)$ its epigraph, the points (t, x) such that $t \geq \log(1 + e^x)$, or equivalently, $e^{x-t} + e^{-t} \leq 1$, so we have the cone,

$$C = \{u + v \leq 1 | (u, 1, x - t) \in K, (v, 1, -t) \in K\}$$

from which we can construct all the t_k constraints. The penalty term involving only lnear inequality constraints poses no further problems. In Mosek the ℓ_1 norm can be treated as a special case of the quadratic cone constraint since $t \geq \sum |x_i|$ can be formulated as $\{(z_i, x_i) \in Q^2, \sum z_i = t\}$ with $Q^2 = \{x \in \mathbb{R}^2 | x_1 \geq \sqrt{x_2^2}\}$. So this adds n quadratic cone constraints with the role of x_1 played by r and the x_2's by the elements of the vector $D\theta$. In practice it seems simpler and may be faster to formulate the ℓ_1 constraint as linear inequality constraints as illustrated in the next section.

An advantage of Mosek's insistence that problems be formulated with linear objective functions and cone constraints is that their dual formulation is straightforward. Given a primal problem,

$$\min\{c^\top x | Ax = b, x \in K\} \tag{P}$$

then we have dual problem,

$$\min\{b^\top y | c - A^\top y \in K^*\} \tag{D}$$

where K^* is the dual cone of K.

3. Implementation

After embarking on this mini-project I discovered that the documentation for the Mosek APIs for C, python and java all contained code for an ℓ_2, aka ridge, penalized version of logistic regression. So the remaining task seemed to be to translate one of these implementations into R and replace the penalty with the ranking lasso penalty. Unfortunately, this proved to be more difficult than it first appeared, and closer reading of Koh et al. (2007) suggested that perhaps Mosek wasn't an ideal vehicle for future development anyway. At that point I toyed with the idea of making an implementation based on the C code from Koh et al. (2007). If I were more conversant with C this probably would have been a reasonable strategy, but in the end I decided to take the coward's way out and construct something based on code that I

already had considerable experience with – to wit quantile regression. Further details about this adventure are provided in the Appendix.

Fortunately, just as my initial experimentation with this median regression implementation was drawing to a close, I received a personal communication Adamaszek (2021) with illustrative Rmosek code for the ℓ_2 penalized version of the logistic regression problem. From there it was easy to implement an alternative version with an ℓ_1 penalty on $D\theta$. The implementation in R using the Rmosek package,Mosek ApS (2021*b*), is quite concise, further bells and whistles, notably weighting effects can be easily added.

```
# Regularized logistic regression
# L2 version by Michal Adamaszek Aug 16, 2021
# See:  https://groups.google.com/g/mosek/c/T2MYKTc8uD4
# L1 version slightly modified by Roger Koenker
# Problem: min{ -l(theta) + lambda || D theta ||_1 }
RLR <- function(X, y, D, lambda)
{
    prob <- list(sense="min")
    n <- nrow(X)
    p <- ncol(X)
    m <- nrow(D)

    # Variables: r, theta(d), u(m), t(n), z1(n), z2(n)
    prob$c <- c(lambda, rep(0,p+m), rep(1, n), rep(0,n), rep(0,n))
    prob$bx <-rbind(rep(-Inf,1+p+m+3*n), rep(Inf,1+p+m+3*n))

    # l1 constraints
    A1 <- rbind(cbind(0, D, diag(m), Matrix(0, m, 3*n)),
                cbind(0, -D, diag(m), Matrix(0, m, 3*n)),
                c(1,rep(0,p),rep(-1,m),rep(0, 3*n)))
    # z1 + z2 <= 1
    A2 <- sparseMatrix( rep(1:n, 2),
                c((1:n)+1+p+m+n, (1:n)+1+p+m+2*n),
                x = rep(1, 2*n))
    prob$A <- rbind(A1,A2)
    prob$bc <- rbind(c(rep(0,1+2*m), rep(-Inf, n)),
                     c(rep(Inf,1+2*m),rep(1, n)))

    # (z1(i), 1, -t(i)) \in \EXP,
    # (z2(i), 1, (1-2y(i))*X(i,) - t(i)) \in \EXP
    FE <- Matrix(nrow=0, ncol = 1+p+m+3*n)
    for(i in 1:n) {
        FE <- rbind(FE, sparseMatrix( c(1, 3, 4, rep(6, p), 6),
                c(1+p+m+n+i, 1+p+m+i, 1+p+m+2*n+i, 2:(p+1), 1+p+m+i),
                x = c(1, -1, 1, (1-2*y[i])*X[i,], -1),
                dims = c(6, 1+p+m+3*n) ) )
    }
    gE <- rep(c(0, 1, 0, 0, 1, 0), n)
```

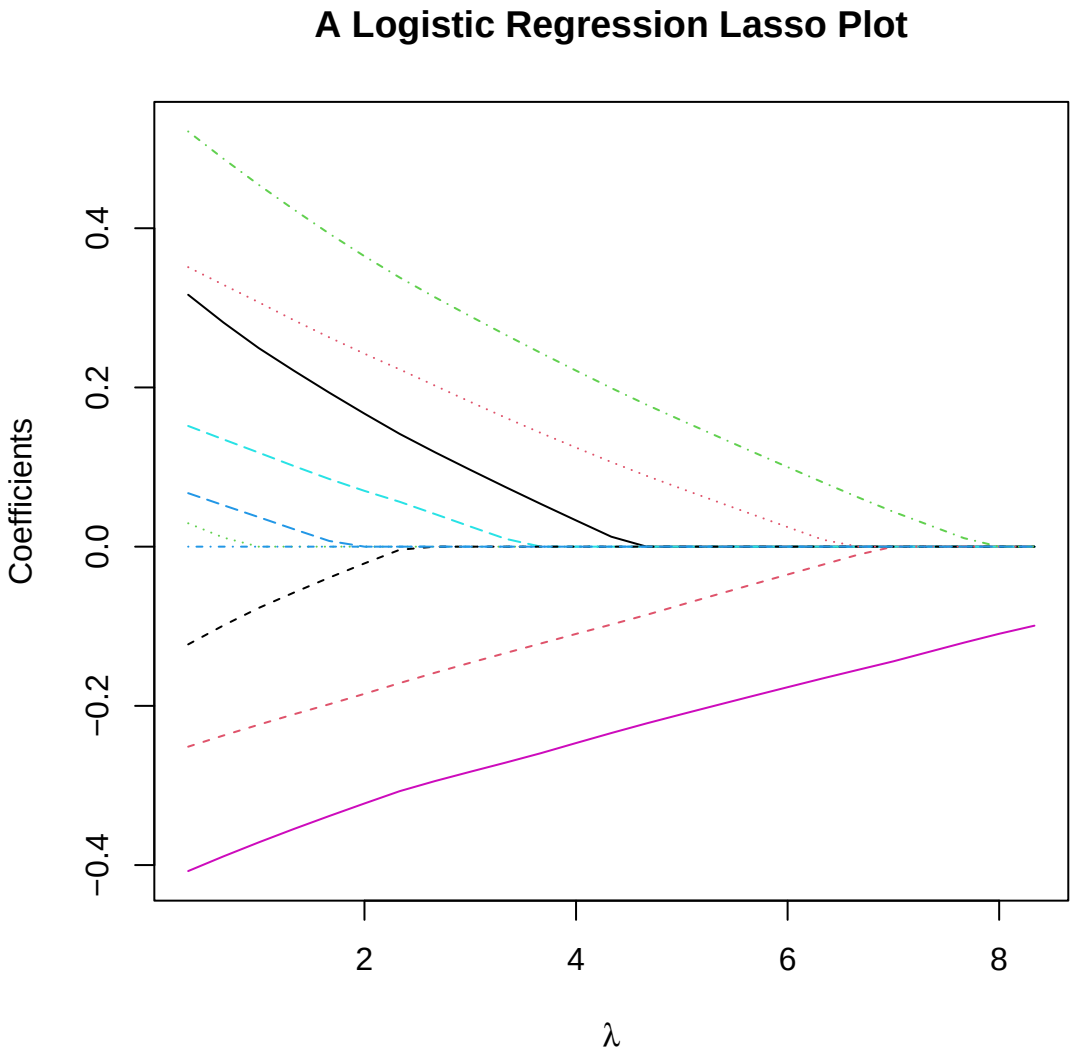
```
    prob$F <- FE
    prob$g <- gE
    prob$cones <- matrix(list("PEXP", 3, NULL), nrow=3, ncol=2*n)
    rownames(prob$cones) <- c("type","dim","conepar")

    # Solve, no error handling!
    # r <- mosek(prob, list(soldetail=1))
    r <- mosek(prob, list(verbose = 0))

    # Return theta
    r$sol$itr$xx[2:(p+1)]
}
```

A simple test problem using the classical $||\theta||_1$ penalty produces a typical lasso shrinkage plot.

```
require(Matrix)
## Loading required package:  Matrix
require(Rmosek)
## Loading required package:  Rmosek
set.seed(1729)
n = 100
p = 10
X <- matrix(rnorm(n*p),n,p)
y <- sample(0:1, n, replace = TRUE)
lambdas <- 1:25/3
B <- matrix(0, length(lambdas),10)
for(i in 1:length(lambdas))
    B[i,] <- RLR(X,y,diag(p),lambdas[i])
matplot(lambdas, B, type = "l", xlab = expression(lambda),ylab = "Coefficients")
title("A Logistic Regression Lasso Plot")
```



In our intended applications the matrix, D, would be slightly more complicated as illustrated in the median regression implementation in the Appendix.

Appendix A. Regularized Median Logistic (Binomial) Regression

What you might ask does ℓ_1 regularized logistic regression have to do with quantile regression? The obvious answer is: When you have a hammer everthing looks like a nail. But two factors conspired to make the RLR problem look more like a nail than one might think. The first factor was that I had no intention of dealing with applications that involved the usual binary response model in its pure Boolean form, rather my applications were all focused on models for which I could aggregate responses into binomial form with relatively large cell counts. Thus, the idea of exploiting the Gaussian approximation to the binomial offered an escape route from Mosek's exponential cones. After all, even in my introductory applied econometrics course hadn't I taught that with frequency data one could get by estimating

models with weighted least squares? See e.g. Koenker (2016). But why least squares, when it would be much easier to attach the ℓ_1 penalty to median regression? All of a sudden, we have a simple data augmentation formulation with a very sparse design structure that is ideally suited to the algorithms already available in the R package **quantreg**.

The canonical implementation I want to consider looks like this,

$$\min_\theta \{\sum_{i=1}^n w_i \rho_\tau(y_i - x_i^\top \theta) + \lambda \|D\theta\|_1\},$$

where w_i's denote weights that would be typically the square root of the weights one would use in the associated least squares problem. The matrix D might also incorporate weights of the form used in Masarotto and Varin (2012) designed to shrink coordinates more when their unconstrained estimates was already close to zero. The intention is to restrict applications to $\tau = 1/2$, but you never know when the temptation to stray from this intention might arise.

Appendix B. An Illustrative Example

To illustrate the approach consider the simple baseball example of Turner and Firth (2012) based on results from the 1987 season involving the 7 teams in the eastern division of the American League. Each pair of teams play 13, 6 at home and 7 away, or vice versa. The unconstrained Bradley Terry model can be estimated easily in R with the following code:

```
library(BradleyTerry2)
MBTm <- BTm(cbind(home.wins, away.wins), home.team, away.team,
            data = baseball, id = "team")
MBTm$coef
##    teamBoston teamCleveland   teamDetroit teamMilwaukee  teamNew York
##     1.1076977     0.6838528     1.4364084     1.5813559     1.2476178
##   teamToronto
##     1.2944851
```

The fitting is accomplished with the aid of base R's `glm.fit` function and uses the default `family = binomial` option with the logistic link function. By default the first team is assigned rating zero. The **BTm** package is somewhat opaque when revealing how the model is actually constructed and fed into the `glm.fit` function so in an effort to demystify things a bit, one can also do this:

```
X <- model.matrix(~home.team-1, data = baseball) -
       model.matrix(~away.team-1, data = baseball)
Y <- with(baseball,cbind(home.wins, away.wins))
Mglm <- glm(Y ~ X[,-1] - 1, family = binomial)
Mglm$coef
##    X[, -1]home.teamBoston X[, -1]home.teamCleveland   X[, -1]home.teamDetroit
##                 1.1076977                 0.6838528                 1.4364084
## X[, -1]home.teamMilwaukee  X[, -1]home.teamNew York   X[, -1]home.teamToronto
##                 1.5813559                 1.2476178                 1.2944851
```

Note that we have dropped the first column of the X matrix and removed the intercept in the formula of the `glm` call. Except for different labeling of the coefficients the fits seem to agree. An unweighted, unregularized median regression version looks like this:

```
logit <- function(p, eps = 0.01) {
    p <- pmax(eps, pmin(1-eps,p))
    log(p/(1-p))
}
y <- logit(Y[,1]/apply(Y,1,sum))
Mrq <- rq(y ~ X[,-1] - 1)
Mrq$coef
##    X[, -1]home.teamBoston X[, -1]home.teamCleveland   X[, -1]home.teamDetroit
##                 1.3217558                 0.9162907                 1.6094379
## X[, -1]home.teamMilwaukee  X[, -1]home.teamNew York   X[, -1]home.teamToronto
##                 1.8971200                 1.6094379                 1.3217558
```

The usual estimate of the variance of logit of $\hat{p}_i$ is $1/(n_i p_i(1-p_i))$ so we should weight the median regression objective function by $\sqrt{n_i p_i(1-p_i)}$.

```
w <- Mrq$fitted
w <- sqrt(apply(Y,1,sum) * exp(w)/(1 + exp(w))^2)
Mwrq <- rq(y ~ X[,-1] - 1, weights = w)
Mwrq$coef
##    X[, -1]home.teamBoston X[, -1]home.teamCleveland   X[, -1]home.teamDetroit
##                 1.0340738                 0.6286087                 1.6094379
## X[, -1]home.teamMilwaukee  X[, -1]home.teamNew York   X[, -1]home.teamToronto
##                 1.6094379                 1.3217558                 1.0340738
```

The ranking, or grouping, penalty is easily implemented as a data augmentation device, we simply introduce pseudo observations corresponding to the penalty terms: Since the first team is assigned rating zero, the penalty matrix includes an identity term that shrinks all the remaining coefficients towards this value, as well as rows that shrink the remaining coefficients toward one another. It is conventional to weight the coordinates of the penalty contribution by the reciprocals of their unconstrained estimates, but these weights can be introduced in the call to `rq` as indicated above. The λ that controls the global shrinkage will be incorporated into the D matrix as shown below.

```
teams <- factor(levels(baseball[,2])[-1])
pairs <- t(combn(teams,2))
D <- model.matrix(~ pairs[,1] + 0) - model.matrix(~ pairs[,2] + 0)
D <- rbind(diag(length(teams)),D)
lambda = 0.1
penw <- abs(D %*% Mwrq$coef) + 1
yp <- c(y, rep(0, length(penw)))
Xp <- rbind(X[,-1], lambda * D)
wp <- c(w, 1/penw)
Mprq <- rq(yp ~ Xp - 1, weights = wp)
Mprq
## Call:
## rq(formula = yp ~ Xp - 1, weights = wp)
##
```

```
## Coefficients:
##    Xphome.teamBoston Xphome.teamCleveland   Xphome.teamDetroit
##            1.0340738            0.6931472            1.6094379
## Xphome.teamMilwaukee  Xphome.teamNew York   Xphome.teamToronto
##            1.6094379            1.3217558            1.0340738
##
## Degrees of freedom: 63 total; 57 residual
```

In this example, two of the proposed penalty weights were zero, so I've introduced a arbitrary additive factor of one to all the weights. With a $\lambda = 0.1$ we get a solution that ranks Detroit and Milwaukee as equivalent and Boston and Toronto as equivalent. Of course, as usual we do not have a principled way to choose λ. Further experimentation with this is clearly needed.

Appendix C. Regularized Logistic Binomial Median Regression

Assembling the foregoing code we can construct a unified function for future reference.

```
RLBMR <- function(D, lambda = 1){
    # D is a data.frame with columns W,L,T1,T2
    X <- model.matrix(~ T1 - 1, data = D) - model.matrix(~ T2 - 1, data = D)
    Y <- with(D,cbind(W, L))
    n <- nrow(X)
    M <- glm(Y ~ X[,-1] - 1, family = binomial)
    logit <- function(p, eps = 0.01) {
        p <- pmax(eps, pmin(1-eps,p)) # Somewhat Kludgy
        log(p/(1-p))
    }
    y <- logit(Y[,1]/apply(Y,1,sum))
    M <- rq(y ~ X[,-1] - 1)
    w <- M$fitted
    w <- sqrt(apply(Y,1,sum) * exp(w)/(1 + exp(w))^2)
    M <- rq(y ~ X[,-1] - 1, weights = w)
    teams <- factor(levels(D$T2)[-1])
    pairs <- t(combn(teams,2))
    P <- model.matrix(~ pairs[,1] - 1) - model.matrix(~ pairs[,2] - 1)
    P <- rbind(diag(length(teams)),P)
    penw <- abs(P %*% M$coef) + 1 # Very kludgy
    yp <- c(y, rep(0, length(penw)))
    Xp <- rbind(X[,-1], lambda * P)
    wp <- c(w, 1/penw)
    f <- rq(yp ~ Xp - 1, weights = wp)
    coef <- c(0,f$coef)
    names(coef) <- levels(D$T1)
    list(coef = coef, resid = f$resid[1:n], lambda = lambda)
}
```

References

Adamaszek, M. (2021), Google groups mosek conversations. Personal Communication: Available at: `https://groups.google.com/g/mosek/c/T2MYKTc8uD4`.

Bradley, R. A. and Terry, M. E. (1952), 'Rank analysis of incomplete block designs: I. the method of paired comparisons', *Biometrika* **39**, 324–345.

Cattelan, M. (2012), 'Models for Paired Comparison Data: A Review with Emphasis on Dependent Data', *Statistical Science* **27**, 412 – 433.

David, H. A. (1988), *The method of paired comparisons*, 2nd edn, Oxford University Press.

Glickman, M. E. (2013), Introductory note to 1928. Available from `http://www.glicko.net/research/preface-z28.pdf`.

Gu, J. and Volgushev, S. (2019), 'Panel data quantile regression with grouped fixed effects', *Journal of Econometrics* **213**, 68–91.

Hocking, T. D., Joulin, A., Bach, F. and Vert, J.-P. (2011), 'Clusterpath: an algorithm for clustering using convex fusion penalties', *Proceedings of the 28th International Conference on International Conference on Machine Learning* pp. 745–752.

Koenker, R. (2016), Lecture 20: Binary response models. Available from: `http://www.econ.uiuc.edu/~roger/courses/508/lectures/L20.pdf`.

Koh, K., Kim, S.-J. and Boyd, S. (2007), 'An interior-point method for large-scale ℓ_1-regularized logistic regression', *J. of Machine Learning Research* **8**, 1519–1555.

Masarotto, G. and Varin, C. (2012), 'The ranking lasso and its application to sport tournaments', *The Annals of Applied Statistics* **6**, 1949 – 1970.

Mosek ApS (2021*a*), MOSEK modeling cookbook. Available from `http://www.mosek.com`.

Mosek ApS (2021*b*), MOSEK Rmosek package. Available from `http://www.mosek.com`.

Turner, H. and Firth, D. (2012), 'Bradley-Terry models in R: The BradleyTerry2 package', *Journal of Statistical Software* **48**, 1–21.

Zermelo, E. (1929), 'Die berechnung der turnier-ergebnisse als ein maximum-problem der wahrscheinlichkeitsrechnung', *Mathematische Zeitschrift* **29**, 436–460.