Math & Optimization for the Polytomous Logistic Regression Model

(Draft of 27-October-2004)

David D. Lewis

1 Introduction

For each i we let $\boldsymbol{x_i} = (x_{i1}, ...x_{ij}, ...x_{id})$ be a d-dimensional vector of real-valued feature values and $y_{i1}, \ldots, y_{ik}, \ldots, y_{ir}$ be binary (0/1) variables encoding a 1-of-r category assignment associated with example i. Exactly one of the y_{ik} will be 1, while the rest will be 0.[1]

A polytomous logistic regression model is described by a parameter vector $\boldsymbol{\beta}$:

$$\begin{aligned}\boldsymbol{\beta} &= (\boldsymbol{\beta}_1, \ldots, \boldsymbol{\beta}_k, \ldots, \boldsymbol{\beta}_r) \\ &= (\beta_{11}, \ldots, \beta_{j1}, \ldots, \beta_{d1}, \beta_{12}, \ldots \beta_{1k}, \ldots, \beta_{jk}, \ldots, \beta_{dk}, \ldots \beta_{1r}, \ldots, \beta_{jr}, \ldots, \beta_{dr})\end{aligned}$$

The output of the model is a vector of values $\hat{p}_{ik}$, one for each k, computed as follows:

$$\hat{p}_{ik} = \frac{\exp(\boldsymbol{\beta}_k \boldsymbol{x})}{\sum_{k'=1}^{r} \exp(\boldsymbol{\beta}_{k'} \boldsymbol{x})} \tag{1}$$

When $\boldsymbol{\beta}$ is produced by maximizing the likelihood function (see below) then the value $\hat{p}_{ik}$ can be viewed as an estimate of $P(y_{ik} = 1)$, i.e. the probability that example $\boldsymbol{x_i}$belongs to category k.

The likelihood function for a set of training data is:

$$L(\boldsymbol{\beta}) = \prod_{i=1}^{n} \prod_{k=1}^{r} \left(\frac{\exp(\boldsymbol{\beta}_k \boldsymbol{x_i})}{\sum_{k'=1}^{r} \exp(\boldsymbol{\beta}_{k'} \boldsymbol{x_i})} \right)^{y_{ik}} p(\boldsymbol{\beta}) \tag{2}$$

where $p(\boldsymbol{\beta})$ is a *prior probability distribution* on $\boldsymbol{\beta}$.

For optimization purposes, one always works with the logarithm of the above function, i.e. the loglikelihood function:

[1] Encodings using +1/-1 instead of 1/0 are possible, but make no fundamental difference to the model.

$$
\begin{aligned}
l(\boldsymbol{\beta}) &= \left(\sum_{i=1}^{n}\sum_{k=1}^{r} y_{ik}\left(\boldsymbol{\beta}_k \boldsymbol{x_i} - \ln\sum_{k'=1}^{r}\exp(\boldsymbol{\beta}_{k'}\boldsymbol{x_i})\right)\right) + \ln p(\boldsymbol{\beta}) \\
&= \sum_{i=1}^{n}\left(\left(\sum_{k=1}^{r} y_{ik}\boldsymbol{\beta}_k \boldsymbol{x_i}\right) - \left(\ln\sum_{k=1}^{r}\exp(\boldsymbol{\beta}_k \boldsymbol{x_i})\right)\right) + \ln p(\boldsymbol{\beta}) \qquad (3)
\end{aligned}
$$

The goal of optimization is to find the $\boldsymbol{\beta}$ (the MAP estimate) that maximizes this function.

Critical quantities in finding the optimal $\boldsymbol{\beta}$ (or, rather, a sufficiently close approximation of it) are the derivatives of the loglikelihood. The first derivative of the loglikelihood function with respect to β_{jk} is[2]

$$
\frac{d}{d\,\beta_{jk}} l(\boldsymbol{\beta}) = \left(\sum_{i=1}^{n} x_{ij}(y_{ik} - \hat{p}_{ik})\right) + \frac{d}{d\,\beta_{jk}} \ln p(\boldsymbol{\beta}). \qquad (4)
$$

The expressions for the second derivatives are:

$$
\begin{aligned}
\frac{d^2}{d^2\,\beta_{jk}} l(\boldsymbol{\beta}) &= \left(-\sum_{i=1}^{n} x_{ij}^2 \hat{p}_{ik}(1-\hat{p}_{ik})\right) + \frac{d^2}{d^2\,\beta_{jk}} \ln p(\boldsymbol{\beta}) \qquad (5) \\
\frac{d^2}{d\beta_{jk}\,d\beta_{jk'}} l(\boldsymbol{\beta}) &= \left(\sum_{i=1}^{n} x_{ij}^2 \hat{p}_{ik}\hat{p}_{ik'}\right) + \frac{d^2}{d\beta_{jk}\,d\beta_{jk'}} \ln p(\boldsymbol{\beta}) \qquad (6) \\
\frac{d^2}{d\beta_{jk}\,d\beta_{j'k}} l(\boldsymbol{\beta}) &= \left(-\sum_{i=1}^{n} x_{ij}x_{ij'} \hat{p}_{ik}(1-\hat{p}_{ik})\right) + \frac{d^2}{d\beta_{jk}\,d\beta_{j'k}} \ln p(\boldsymbol{\beta}) \qquad (7) \\
\frac{d^2}{d\beta_{jk}\,d\beta_{j'k'}} l(\boldsymbol{\beta}) &= \left(\sum_{i=1}^{n} x_{ij}x_{ij'} \hat{p}_{ik}\hat{p}_{ik'}\right) + \frac{d^2}{d\beta_{jk}\,d\beta_{j'k'}} \ln p(\boldsymbol{\beta}) \qquad (8)
\end{aligned}
$$

where here we assume $j \neq j', k \neq k'$.

2 Priors

In the Bayesian statistical framework the prior probability, $p(\boldsymbol{\beta})$, expresses our belief (prior to seeing the data to be fitted) that certain $\boldsymbol{\beta}$'s are more likely than others. Here we are

[2]Note that this equation was incorrect in my May 2004 version of this document, even taking into account the use of $+1/-1$ for y_{ik}. This also led me to an incorrect belief about being able to tell that some parameters are 0 or identical without doing fitting.

Name	Distribution $p(\beta_{jk}\|\boldsymbol{\lambda})$	Log distribution $\ln p(\beta_{jk}\|\boldsymbol{\lambda})$	First derivative $\frac{d}{d\beta_{jk}} \ln p(\beta_{jk}\|\boldsymbol{\lambda})$
Gaussian	$\frac{1}{\sqrt{2\pi}\sigma_{jk}} \exp(\frac{-\beta_{jk}^2}{2\sigma_{jk}^2})$	$-\ln \sigma_{jk} - \frac{\ln 2\pi}{2} - \frac{\beta_{jk}^2}{\sigma_{jk}^2}$	$-\frac{2}{\sigma_{jk}^2}\beta_{jk}$
Laplace	$\frac{\lambda_{jk}}{2} \exp(-\lambda_{jk}\|\beta_j\|)$	$-\ln 2 + \ln \lambda_{jk} - \lambda_{jk}\|\beta_j\|$	$-\lambda_{jk}\text{sgn}(\beta_{jk}), \beta_{jk} \neq 0$
exponential	$\lambda_{jk} \exp(-\lambda_{jk}\beta_j)$	$\ln \lambda_{jk} - \lambda_{jk}\beta_j$	$-\lambda_{jk}$

Table 1: *Three univariate densities that we currently use for the prior distribution of individual coefficients of a logistic regression model. The Laplace and Gaussian take on nonzero values for all real β_{jk}, while the exponential is defined only for $\beta_{jk} >= 0$.*

only concerned with the mathematical form that $\ln p(\boldsymbol{\beta})$ takes, not its origin.

Table 1 presents three univariate distributions which are convenient to use as the prior distribution for a coefficient β_{jk} of a logistic regression model. To define a prior for $\boldsymbol{\beta}$ as a whole, we take the product of one such distribution for each coefficient.

The loglikelihood function and first derivative in the Gaussian case is:

$$l(\boldsymbol{\beta}) = \sum_{i=1}^{n} \left(\left(\sum_{k=1}^{r} y_{ik} \boldsymbol{\beta}_k \boldsymbol{x_i} \right) - \left(\ln \sum_{k=1}^{r} \exp(\boldsymbol{\beta}_k \boldsymbol{x_i}) \right) \right) - \sum_{j=1}^{d} \sum_{k=1}^{r} \lambda_{jk} \beta_{jk}^2 \quad (9)$$

$$\frac{d}{d\,\beta_{jk}} l(\boldsymbol{\beta}) = \left(\sum_{i=1}^{n} x_{ij} (y_{ik} - \hat{p}_{ik}) \right) - 2\lambda_{jk}\beta_{jk} \quad (10)$$

Note that $\lambda_{jk} = -2/\sigma_{jk}^2$, where σ_{jk}^2 is the variance of the Gaussian

Laplace priors give this form:

$$l(\boldsymbol{\beta}) = \sum_{i=1}^{n} \left(\left(\sum_{k=1}^{r} y_{ik} \boldsymbol{\beta}_k \boldsymbol{x_i} \right) - \left(\ln \sum_{k=1}^{r} \exp(\boldsymbol{\beta}_k \boldsymbol{x_i}) \right) \right) - \sum_{j=1}^{d} \sum_{k=1}^{r} \lambda_{jk} |\beta_{jk}| \quad (11)$$

$$\frac{d}{d\,\beta_{jk}} l(\boldsymbol{\beta}) = \left(\sum_{i=1}^{n} x_{ij} (y_{ik} - \hat{p}_{ik}) \right) - \lambda_{jk}\text{sgn}(\beta_{jk}),\ \beta_{jk} \neq 0 \quad (12)$$

Note that the first derivative of the loglikelikhood with the Laplace prior is not defined at $\beta_{jk} = 0$, though fortunately a standard trick (Section 6) keeps this from being a problem for optimization.

Exponential priors give this form:

$$l(\boldsymbol{\beta}) = \sum_{i=1}^{n} \left(\left(\sum_{k=1}^{r} y_{ik} \boldsymbol{\beta}_k \boldsymbol{x_i} \right) - \left(\ln \sum_{k=1}^{r} \exp(\boldsymbol{\beta}_k \boldsymbol{x_i}) \right) \right) - \sum_{j=1}^{d} \sum_{k=1}^{r} \lambda_{jk} \beta_{jk} \quad (13)$$

$$\frac{d}{d\,\beta_{jk}} l(\boldsymbol{\beta}) = \left(\sum_{i=1}^{n} x_{ij} (y_{ik} - \hat{p}_{ik}) \right) - \lambda_{jk} \quad (14)$$

Note that the exponential prior, and thus its loglikelihood and first derivative of loglikelihood, are defined only when all $\beta_{jk} >= 0$.

We often, but not always, use the same value λ for all λ_{jk}. In all cases the λ_{jk} are treated as constants during optimization.[3]

We also sometimes are interested in variations of these priors where the penalty term takes the form $(\beta_{jk} - \mu_{jk})^2$ or $|\beta_{jk} - \mu_{jk}|$ for specified constants μ_{jk}. This modification does not change the optimization problem in any fundamental way. We may also be interested in using different forms of priors for different features. This does not introduce any optimization issues beyond those of dealing with the individual forms.

3 Characteristics of the Optimization Problem

The major characteristics of the optimization problem seem to be:

- Convexity: The loglikelihood function is convex with any of the above priors, or with no prior.
- Continuity: The loglikelihood function is continuous with the Gaussian prior. With the Laplace and exponential priors, it is continuous except at points with one or more $\beta_{jk} = 0$.
- Finite solution: With any of above priors the optimal $\boldsymbol{\beta}$ is finite. (This is not true in the absence of a prior.)

[3] Actually we *are* sometimes interested in solving the problem for each of a set of λ values, or in finding the λ that optimizes some other criterion, but this is not currently handled by another mechanism. See item 3 in Section 5.

- Almost linear: While this is a nonlinear optimization problem, it is a well-behaved one. This results because the logistic function is often nearly linear in interesting portions of the $\boldsymbol{\beta}$ space, and because we take the sum of many terms.

- Almost least squares: The optimization problem has similarities to a least squares problem. The standard algorithm for maximizing the loglikelihood function in statistical software is Newton-Raphson, which ends up being implemented as a succession of weighted least squares problems.

- Large data: We are interested in 10^2 to 10^6 $\boldsymbol{x_i}$'s, each a sparse vector of length 10^3 to 10^7 and being mapped to a binary output vector of length 10^1 to 10^4. Total number of parameters in $\boldsymbol{\beta}$ may be 10^3 to 10^{11}.

- Sparse targets: For each i, only one y_{ik} is nonzero. This is critical, but straightforward, to take into account in efficient implementations.

- Sparse inputs: The individual $\boldsymbol{x_i}$'s (rows of the data matrix) are sparse. Most, but not all, of the columns of the data matrix are sparse as well. This is important to make use of in efficient implementations, and the issues in doing so vary among optimization algorithms.

- Sparse solutions: When the Laplace or exponential prior, i.e. an L_1 penalty, is used, the vast majority of parameters are typically 0 at the solution. Taking advantage of this requires clever implementation and probably would benefit from fundamental insights.

- Structured Hessian: The average size of an entry in the Hessian is heavily dependent on whether $j = j'$ and $k = k'$. If $j \neq j'$, i.e. the two parameters are for different features, then the Hessian entry is affected by the product (for each training example) of two different feature values. Since features are sparse, these products will often be 0, and it will not be unusual for the Hessian entry to be 0. If $k \neq k'$, i.e. the two parameters are for different classes, then the Hessian entry is affected by the product $\hat{p}_{ik}\hat{p}_{ik'}$ (for each training example) of the posterior probabilities for two different classes. For finite values of $\boldsymbol{\beta}$ we have $\hat{p}_{ik} + \hat{p}_{ik'} < 1$. This contrasts with the product $\hat{p}_{ik}(1 - \hat{p}_{ik})$, with $\hat{p}_{ik} + (1 - \hat{p}_{ik}) = 1$, that figures in entries for $k = k'$. This knowledge may be useful in some optimization algorithms.

4 Published Results

In this section I briefly note some published results on optimization algorithms for logistic regression. It's important to distinguish papers which studied the polytomous problem from those that studied only the easier binary problem. (The binary problem corresponds to a polytomous problem with r=2 and one of the $\boldsymbol{\beta_k}$ forced to be all 0's.) The binary problem is easier from an optimization standpoint because correlations between coefficients are much weaker.

Malouf studied natural language processing problems with very large numbers of classes (up to r = 8.6 million) and sparse inputs (with up to d = 260,000 features). He appears not to have used a regularization term (!), instead relying on terminating when successive loglikelihood values were sufficient close together. He found that for the largest problem a limited memory Quasi-Newton method was 8 times faster than the second best method, a Polak-Ribere-Positive version of conjugate gradient. The particular limited-memory Quasi-Newton method used not described clearly, but appears to have been one available in the C++ TAO optimization package from Argonne.

Sha and Pereira also studied very large language processing problems (r = 3 classes, and d = 820,000 to 3.8 million features) and found limited-memory BFGS (with 3-10 pairs of previous gradients and updates saved) and preconditioned conjugate gradient performed similarly, and much better than iterative scaling or plain conjugate gradient. They used a Gaussian penalty on the loglikelihood.

Komarek did a detailed study of binary logistic regression on a range of datasets with varying size and sparsity. He found iteratively reweighted least squares (Newton's method) with conjugate gradient to solve the least squares problem to be highly efficient, as well using conjugate gradient directly. Both were found more efficient that BFGS, but of course this was a binary problem. He found that a Gaussian penalty contributed greatly to numerical stability and efficiency.

Minka experimented on moderate sized dense, artificial data sets. He studied binary logistic regression with no regularization term. He found conjugate gradient and a cyclic coordinate descent method (similar to Zhang-Oles) beat Newton's method and several other algorithms including a quasi-Newton method. However, on an artificial dataset designed so that parameters would be highly correlated he found Newton's method was best, followed by a

quasi-Newton method.

Krishnapuram, Carin, Figueiredo, and Hartemink experimented on small, dense classification problems from the Irvine archive. They studied polytomous logistic regression with an L1 penalty, equivalent to a Laplace prior. They claimed a cyclic coordinate descent method (very similar to Zhang-Oles) beat conjugate gradient by "orders of magnitude" but provided no quantitative data.

5 Thoughts

Here are my nonexpert thoughts/questions on approaches that might lead to a good optimization algorithm.

1. Newton's method and quasi-Newton methods applied to the full parameter vector are probably impossible for us, due to Hessian being too big, even taking sparsity into account. Methods that use only the gradient, such as limited memory quasi-Newton or conjugate gradient would be possible.

2. We might be able to use knowledge of which Hessian entries are largest:

   - In a quasi-Newton method we might start with a matrix better than the identity.
   - I wonder about holding most parameters constant on any given iteration, and optimizing only on groups of the more highly correlated parameters. Of the non-diagonal entries, the largest ones are likely to be those that have $j = j', k \neq k'$, i.e. the set of parameters for the same feature, across classes. This is particularly true for the more dense j's.

3. Our regularization penalties are of the same form as penalties used to solve optimization problems under constraints.[4] In solving problems of those sorts, I saw in a textbook that one wants to solve a succession of problems, starting with small λ's and increasing the λ's until the solution comes close enough to satisfying the constraints. The rationale is that using very large λ's at the start creates a function that is hard for optimizers to

[4] Actually, while I've seen textbook descriptions of L_2 penalties to enforce constraints, I haven't seen L_1 penalties used for that purpose. Is that done?

work with. We of course have a particular value of λ we want to find a solution for[5], but perhaps we would want to start with small values of λ and gradually increase it until we reach our desired value, to aid optimization. I also wonder if we should get a rough solution with an L_2 penalty before switching to L_1 when that's what we want.

6 Plan

Our current plan is to use a limited memory quasi-Newton (LMQN) algorithm that supports bounds on variables (i.e. box constraints). This is motivated both by Malouf's results, which are on the data that is the most similar to ours, and by the availability of good implementations of bound-constrained LMQN algorithms, in particular Steve Benson's *blmvm* C code from Argonne.[6]

In the Gaussian prior case we do not need to use bounds on the variables. Values of $\beta_{jk} < 0$ are forbidden by an exponential prior, so there we would use a lower bound of 0, but no upper bound.

The Laplace prior at first seems problematic, since it lacks a derivative at 0, a point in the range of legal values. But a simple trick (apparently widely known in linear programming) fixes this problem. We replace each original feature value x_{ij} with two new features $x_{ijp} = x_{ij}$ and $x_{ijq} = -x_{ij}$. We correspondingly replace each coefficient β_{jk} in the coefficient vector with two new coefficients β_{jkp} and β_{jkq}. If $\frac{1}{2}\lambda_{jk} \exp(\lambda_{jk})|\beta_{jk}|$ was the prior on the original β_{jk}, we use exponential priors $\lambda_{jk} \exp(\lambda_{jk}|\beta_{jkp}|)$ and $\lambda_{jk} \exp(\lambda_{jk}|\beta_{jkq}|)$ on the two new coefficients.

[5] Actually, often as part of experimentation we want to try a set of values of λ, which would be another reason to use this technique.

[6] Available at *http://www-unix.mcs.anl.gov/~benson/blmvm/*. Also available in TAO (*http://www-unix.mcs.anl.gov/tao/*).

The loglikelihood and first derivative expressions for the reformulated problem are:

$$\begin{aligned} l(\boldsymbol{\beta}) &= \left(\sum_{i=1}^{n}\left(\sum_{k=1}^{r} y_{ik}(\sum_{j=1}^{d}\beta_{jkp}x_{ijp} + \beta_{jkq}x_{ijq})\right)\right) \\ &\quad - \left(\ln\sum_{k=1}^{r}\exp(\sum_{j=1}^{d}\beta_{jkp}x_{ijp} + \beta_{jkq}x_{ijq})\right) - \sum_{j=1}^{d}\sum_{k=1}^{r}(\lambda_{jkp}\beta_{jkp} + \lambda_{jkn}\beta_{jkn}) \quad (15) \\ \frac{d}{d\,\beta_{jk}}l(\boldsymbol{\beta}) &= \left(\sum_{i=1}^{n} x_{ij}(y_{ik} - \hat{p}_{ik})\right) - (\lambda_{jkn} + \lambda_{jkp}) \quad (16) \end{aligned}$$

where as usual with the exponential prior we require all coefficients of the solution (in this case β_{jkp}'s and β_{jkq}'s) to be $>= 0$. As with the exponential prior, this can be handled by using an optimizer that supports box constraints.

We prove the following claims in the Appendix:

- Let $\tilde{\boldsymbol{\beta}}$ be the $2dr$-dimensional MAP solution for the transformed problem. Then for each pair j, k, at most one of $\tilde{\beta}_{jkp}$ and $\tilde{\beta}_{jkq}$ is nonzero.
- Define a dr-dimensional $\hat{\boldsymbol{\beta}}$ by letting, for each jk, $\hat{\beta}_{jk} = \tilde{\beta}_{jkp}$ if $\tilde{\beta}_{jkp} > 0$, or $\hat{\beta}_{jk} = -\tilde{\beta}_{jkq}$ if $\tilde{\beta}_{jkq} > 0$. Then $\hat{\boldsymbol{\beta}}$ is the MAP solution to the original problem.

In other words, we can find the MAP solution to a Laplace penalty problem of size dr by finding the MAP solution to a corresopnding exponential penalty problem of size $2dr$. This doubles the size of the major data structures (gradients and coefficient vectors) used during optimization. However, the memory for the data set need not double, since the new features can be handled implicitly.) The effect on CPU time is unclear. Time to compute the loglikelihood value will change by very little (again, since the new features can be handled implicitly), nor will the number of calls to the exponentiation function (a major expense) change.

7 Implementation Issues

(DDL: Insert revised version of my old email on this, particularly the critical heuristic of zeroing coefficients corresponding to features that don't occur with a class.)

8 Appendix

8.1 Proving that Only One of β_{jkp} and β_{jkn} Can Be Nonzero in the MAP Solution

Suppose $\boldsymbol{\beta}$ maximizes (is the MAP solution to) Equation 15, but both $\beta_{lmp} > 0$, and $\beta_{lmq} > 0$. Assume that $\beta_{lmp} >= \beta_{lmq}$ (the opposite case is almost identical).

Define $\boldsymbol{\beta'}$ to be identical to $\boldsymbol{\beta}$, except that $\beta'_{lmp} = \beta_{lmp} - \beta_{lmq}$ and $\beta'_{lmq} = 0$.

Note that $\boldsymbol{\beta}$ and $\boldsymbol{\beta'}$ have identical dot products with the expanded example vector, since

$$\begin{aligned} y_{im}\beta'_{lmp}x_{ilp} + y_{im}\beta'_{lmq}x_{ilq} &= y_{im}(\beta_{lmp} - \beta_{lmq})x_{il} + y_{im}(0)(-x_{il}) \\ &= y_{im}(\beta_{lmp})x_{il} + y_{im}(-\beta_{lmq})(x_{il}) + 0 \\ &= y_{im}\beta_{lmp}x_{il} + y_{im}\beta_{lmq}(-x_{il}) \\ &= y_{im}\beta_{lmp}x_{ilp} + y_{im}\beta_{lmq}x_{ilq}, \end{aligned} \tag{17}$$

and all other coordinates of $\boldsymbol{\beta}$ and $\boldsymbol{\beta'}$ are equal. Therefore $l(\boldsymbol{\beta})$ and $l(\boldsymbol{\beta'})$ differ only in their prior component. So

$$\begin{aligned} l(\boldsymbol{\beta}) - l(\boldsymbol{\beta'}) &= \left(-\sum_{j=1}^{d}\sum_{k=1}^{r}(\lambda_{jkp}\beta'_{jkp} + \lambda_{jkq}\beta'_{jkq})\right) + \left(\sum_{j=1}^{d}\sum_{k=1}^{r}(\lambda_{jkp}\beta_{jkp} + \lambda_{jkq}\beta_{jkq})\right) \\ &= -\lambda_{lmp}\beta'_{lmp} - \lambda_{lmq}\beta'_{lmq} + \lambda_{lmp}\beta_{lmp} + \lambda_{lmq}\beta_{lmq} \\ &= -\lambda_{lmp}(\beta_{lmp} - \beta_{lmq}) - \lambda_{lmq}(0) + \lambda_{lmp}\beta_{lmp} + \lambda_{lmq}\beta_{lmq} \\ &= -\lambda_{lm}(\beta_{lmp} - \beta_{lmq}) + \lambda_{lm}\beta_{lmp} + \lambda_{lm}\beta_{lmq} \\ &= \lambda_{lm}(-\beta_{lmp} + \beta_{lmq} + \beta_{lmp} + \beta_{lmq}) \\ &= 2\lambda_{lm}\beta_{lmq} \\ &> 0 \end{aligned} \tag{18}$$

But this contradicts our assumption that $\boldsymbol{\beta}$ is the MAP solution.

8.2 Proving That The MAP Solution for a Laplace Prior Can Be Derived from the MAP Solution for the Exponential Prior on the Duplicated Vector

(To be written.)