Proceedings of the World Congress on Engineering 2011 Vol I
WCE 2011, July 6 - 8, 2011, London, U.K.

Multinomial Least Angle Regression with Application to Web Personalization

Ilya Gluhovsky

***Abstract*—Keerthi and Shevade (2007) proposed an efficient algorithm for constructing an approximate LARS solution path for logistic regression as a function of the regularization parameter. In this paper we extend their approach to multinomial regression. We show that a brute-force approach leads to a multivariate approximation problem resulting in an infeasible path tracking algorithm. Instead, we introduce a non-canonical link function thereby a) repeatedly reusing the univariate approximation of Keerthi and Shevade and b) producing an optimization objective with a block-diagonal Hessian. We carry out a simulation study that shows the computational efficiency of the proposed technique. A Matlab implementation is available from the author upon request. We apply this technique to a web personalization problem in corporate marketing.**

***Index Terms*—Machine learning, supervised learning, generalized linear models (GLM), least angle regression and LASSO (LARS), least absolute shrinkage and selection operator (LASSO), large-scale regression, solution path tracking.**

I. Introduction

We consider a multinomial regression solution to a K-class classification problem. Let $\{(\mathbf{x}_i, y_i),\ 1 \leq i \leq N\}$ denote the training data set. It is assumed to be an independent, identically distributed sample of $(\mathbf{X}, Y)$, a data pair of a predictor vector $\mathbf{X}$ and a response Y that takes values in the set $\{1, 2, \cdots, K\}$ of class labels. Furthermore, the conditional distribution of Y is assumed to be

$$Y|\mathbf{X} = \mathbf{x} \sim \text{Multinomial}(p_1(\mathbf{x}), \cdots, p_K(\mathbf{x})),$$

where $p_k(\mathbf{x})$ are unknown conditional class probabilities

$$p_k(\mathbf{x}) = P\{Y = k|\mathbf{X} = \mathbf{x}\}$$

for a predictor realization $\mathbf{x}$. Our goal is to estimate the $p_k(\mathbf{x})$.

We adopt a generalized linear model [1] with link function ρ, so that

$$p_k(\mathbf{x}, \beta) = P\{Y = k|\mathbf{x}, \beta) = \rho_k\left(\mathbf{x}^T\beta^{(1)}, \cdots, \mathbf{x}^T\beta^{(K)}\right).$$

The purpose of the link function in a generalized linear model is to map the linear predictors, in this case $\mathbf{x}^T\beta^{(k)}, 1 \leq k \leq K$, into the mean of the response distribution, in this case $p_k(\mathbf{x}, \beta), 1 \leq k \leq K$. In particular, the p_k are now parameterized by $\beta = \left(\beta^{(1)}, \cdots, \beta^{(K)}\right)$, the unknown parameter vector that determines the model, where $\beta^{(k)}$ denotes the parameter vector corresponding to class k with $\beta^{(K)} \equiv 0$ by convention to remove ambiguity introduced by the constraint $\sum_k p_k(\mathbf{x}, \beta) = 1$. A typical choice of ρ is the canonical link [1]

$$p_k(\mathbf{x}, \beta) = \frac{\exp\left(\mathbf{x}^T\beta^{(k)}\right)}{\sum_{l=1}^{K}\exp\left(\mathbf{x}^T\beta^{(l)}\right)}. \tag{1}$$

I. Gluhovsky is Chief Scientist, Ancestry.com Inc., San Francisco, CA 94107, ilyagluk@gmail.com, igluhovsky@ancestry.com

Manuscript received February 02, 2011; revised April 19, 2011.

As a criterion of finding β, we pose the penalized likelihood problem

$$\min_{\beta} f_\lambda(\beta), \tag{2}$$

where

$$f_\lambda(\beta) = f_0(\beta) + \lambda||\beta||_1 \tag{3}$$

and

$$f_0(\beta) = \frac{\mu}{2}\beta^T A\beta - \sum_{i=1}^{N}\log p_{y_i}(\mathbf{x}_i, \beta) \tag{4}$$

is the sum of an L_2 regularization penalty and a negative log-likelihood. The regularization penalty matrix A is any positive semidefinite matrix, such as the identity or a kernel penalty matrix [2]. Thus, our regularization introduces both an L_1 penalty in (3) and an L_2 penalty in (4), a so-called *elastic net* proposed in [3]. While μ is fixed at a small value as in [3], our task is to compute the solution $\hat{\beta}(\lambda)$ as a function of the L_1 penalty weight λ known as a *regularization parameter* as the latter varies over the nonnegative numbers. We will refer to $\hat{\beta}(\lambda)$ as the solution path.

When λ is very large, $\hat{\beta}(\lambda) = \mathbf{0}$ as the L_1 penalty dominates. As λ decreases, more and more coordinates of β become nonzero, typically one at a time leading to sparse solutions for *interesting* values of λ. This is contrasted with using a single L_2 penalty where typically all coefficients $\hat{\beta}(\lambda)$ become nonzero simultaneously [4], [5]. The latter choice results in many small coefficients which are typically estimated with high variance. Therefore, L_1 penalty solutions are often superior in estimation accuracy [5]. A sparse solution also offers important practical advantages as a production system that uses one needs to concern itself with generating only a subset of predictors in real time, so that online model evaluations become faster. Finally, as λ approaches zero, $\hat{\beta}(\lambda)$ approaches the minimizer of (4), which is the maximum likelihood solution subject to the small L_2 penalty.

Spearheaded by seminal research [4], efficient algorithms for constructing the *entire* solution path for all values of λ were proposed in [2], [5], [6]. Advantages of solution path tracking include a) sorting the predictors by their importance based on the order in which they enter the model as λ decreases and b) choosing the best fit in the entire solution path (e.g. by cross-validation) having yielded *all* the parameter vectors, which are calculated in one pass by the procedure. The computational complexity of path-tracking algorithms is comparable to that of a single-λ optimization [5]. This result is concurred in practice by [7] showing that running times of path tracking are not much worse than those of single-λ competitors. Since selecting the λ typically involves running the latter with many trial values, path tracking offers a significantly better quality/running-time tradeoff than a

ISBN: 978-988-18210-6-5
ISSN: 2078-0958 (Print); ISSN: 2078-0966 (Online)

one-λ-at-a-time optimization heuristic. Our empirical study in Section V verifies this property for the proposed path tracking algorithm.

Inclusion of the L_1 penalty into a fitting criterion was first treated in detail in [8], where it was dubbed *LASSO*. [4], [5] proposed a path-tracking method called *least angle regression and LASSO* (LARS) for squared-error loglikelihood (as in linear regression solved by ordinary least squares) whose simple modification in particular yields the entire LASSO solution path. [6] extended the technique to generalized regression models with piecewise-quadratic loglikelihood. [2] proposed an efficient approximation method that reduces the logistic regression problem to the setting in [6].

Path tracking is based on the following observation. Whenever the function f_0 in (4) is piecewise quadratic in β, the solution path $\hat{\beta}(\lambda)$ is piecewise linear. At each step, [6] calculates the direction of a new segment and the extent to which the solution path follows the segment. [2] specializes to logistic regression, which is a special case of multinomial regression with the number of classes $K = 2$. In order to use the path tracking algorithm in [6], [2] proposes a particular piecewise quadratic approximation to the loglikelihood in (4). Additionally, [2] proposes a pseudo-Newton correction process applied after the original path construction to reduce the impact of the approximation and move closer to the true path.

In this paper we propose an approximate path tracking method for multinomial regression. To the best of our knowledge, this is the first multinomial least angle regression method. We now outline the specifics of our contribution. To find a piecewise quadratic approximation to $f_0(\beta)$ in (4), it is sufficient to approximate the summands $\log p_{y_i}(\mathbf{x}_i, \beta)$. Using the canonical link (1), the latter is equivalent to approximating

$$\log\left[\exp\left(\mathbf{x}^T\beta^{(1)}\right) + \cdots + \exp\left(\mathbf{x}^T\beta^{(K-1)}\right) + 1\right] \quad (5)$$

since the logarithm of the numerator of (1) is already linear in β and $\beta^{(K)} \equiv 0$. In the logistic case of $K = 2$, [2] finds a tight piecewise quadratic approximant $\hat{l}(r)$ to a univariate function

$$l(r) = \log\left(1 + e^{-r}\right) \quad (6)$$

depicted in Figure I. Since $r = -\mathbf{x}^T\beta^{(1)}$ is linear in β, this not only offers the sought-after approximation, but also leads to an efficient step for determining the approximation segment for any particular $r_i = \mathbf{x}_i^T\beta^{(1)}$. This efficiency is critical for computational feasibility because the direction of the solution path is recomputed every time any of the r_i hits a segment endpoint along with the magnitude of the extension of the path in the new direction. Therefore, the extension computation is frequent and has to be performed for every data point every time. For medium-size datasets in Section V, a typical number of such computations is 100 million for constructing the entire solution path.

Let us now examine an analogous approach in the case of $K > 2$. With $r_k = \mathbf{x}^T\beta^{(k)}$ in (5) we would have to introduce an approximation to a $(K-1)$-variate function

$$l_K(r_1, \cdots, r_{K-1}) = \log(1 + e^{-r_1} + \cdots + e^{-r_{K-1}}). \quad (7)$$

Dealing with complex $(K-1)$-dimensional approximation regions in place of one-dimensional segments would lead to an infeasible algorithm for data sets of moderate sizes as discussed in Section IV.

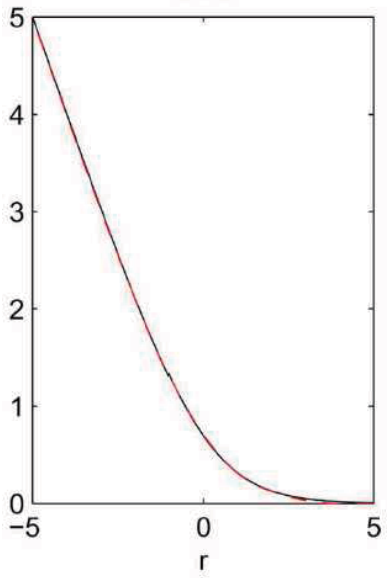

Fig. 1. $l(r)$ (black solid) and its piecewise quadratic approximant $\hat{l}(r)$ (dashed red). Reproduced from [2].

TABLE I
APPROXIMATION SEGMENTS USED TO DEFINE $\hat{l}(r)$ IN (8) ALONG WITH THE CORRESPONDING VALUES OF THE PIECEWISE CONSTANT COEFFICIENT $a(r)$.

r	$(-\infty, -4]$	$(-4, -1.65]$	$(-1.65, 1.65]$	$(1.65, 4]$	$(4, \infty)$
$a(r)$	0	0.0618	.215	0.0618	0

We solve this problem by proposing a particular non-canonical link function in place of (1). This allows us to continue using a univariate approximation to (6) applied up to $K-1$ times for each data point. This link function also produces an optimization objective with a block-diagonal Hessian with $K-1$ blocks. These two benefits lead to algorithmic efficiency similar to that of [2] in the multinomial setting.

In Section II we give details of the proposed link function and the ensuing piecewise-quadratic approximation to the loglikelihood. The path tracking algorithm is described in Section III. In Section IV we further contrast our algorithm with a brute-force approach leading to the $(K-1)$-variate approximation outlined above. In Section V we carry out a brief simulation study that illustrates the computational efficiency of the proposed algorithm as discussed above across various numbers N, P, and K of observations, predictor variables, and class labels respectively. In Section VI we apply the techinique to a web personalization problem in corporate marketing. We conclude in Section VII.

II. New Link Function and a Piecewise Quadratic Approximation to Loglikelihood

[2] proposes a particular nonnegative, convex, continuously differentiable, and piecewise quadratic approximant $\hat{l}(r)$ to $l(r)$ in (6). These properties meet the requirements of the path tracking algorithm in [6]. Figure I illustrates its tightness achieved with just 5 approximation segments that make up the horizontal axis and are given in the first row in

ISBN: 978-988-18210-6-5
ISSN: 2078-0958 (Print); ISSN: 2078-0966 (Online)

Table I. $\hat{l}(r)$ is given by

$$\hat{l}(r) = (1/2)a(r)r^2 + b(r)r + c(r) \tag{8}$$

with $a(r), b(r)$, and $c(r)$ being constant within each segment. In particular, $\hat{l}(r)$ is linear over $(-\infty, -4]$ with the slope of -1 and is zero over $[4, +\infty)$. Only the values of $a(r)$ given in Table I are used in the path tracking algorithm in Section 3. Other details are available in [2].

We now introduce the new link function that allows us to approximate the multinomial loglikelihood by only involving this univariate $\hat{l}(r)$ and not its multivariate counterparts as outlined above and detailed in Section IV. It is defined by

$$\frac{\sum_{l=1}^{k} p_l(\mathbf{x}, \beta)}{\sum_{l=1}^{k+1} p_l(\mathbf{x}, \beta)} = \frac{1}{1 + \exp\left(-\mathbf{x}^T \beta^{(k)}\right)}, k = 1, \cdots, K-1 \tag{9}$$

or equivalently,

$$\frac{p_{k+1}(\mathbf{x}, \beta)}{\sum_{l=1}^{k+1} p_l(\mathbf{x}, \beta)} = \frac{1}{1 + \exp\left(\mathbf{x}^T \beta^{(k)}\right)}, k = 1, \cdots, K-1.$$

Thus, the linear predictor $\eta_k = \mathbf{x}^T \beta^{(k)}$ determines the fraction of $P\{Y = k+1|\mathbf{x}, \beta\}$ in $P\{Y \leq k+1|\mathbf{x}, \beta\}$. This is known as *stick-breaking construction* often used in conjunction with Dirichlet processes[1]. In particular, any distribution $\{p_1(\mathbf{x}, \beta), \cdots, p_K(\mathbf{x}, \beta)\}$ can be represented by (9) for the appropriate choice of linear predictors $\{\eta_1, \cdots, \eta_K\}$ and, conversely, any choice of the linear predictors corresponds to a valid distribution $\{p_1(\mathbf{x}, \beta), \cdots, p_K(\mathbf{x}, \beta)\}$ obtained through (9).

Below we use the convention that an empty product is one and an empty sum is zero. The p_k can be recovered by

$$p_1(\mathbf{x}, \beta) = \prod_{m=1}^{K-1} \frac{1}{1 + \exp\left(-\mathbf{x}^T \beta^{(m)}\right)},$$

and for $k \geq 2$,

$$p_k(\mathbf{x}, \beta) = \sum_{l=1}^{k} p_l(\mathbf{x}, \beta) - \sum_{l=1}^{k-1} p_l(\mathbf{x}, \beta) =$$

$$\prod_{m=k}^{K-1} \frac{1}{1 + \exp\left(-\mathbf{x}^T \beta^{(m)}\right)} - \prod_{m=k-1}^{K-1} \frac{1}{1 + \exp\left(-\mathbf{x}^T \beta^{(m)}\right)} =$$

$$= \prod_{m=k}^{K-1} \frac{1}{1 + \exp\left(-\mathbf{x}^T \beta^{(m)}\right)} \left[1 - \frac{1}{1 + \exp\left(-\mathbf{x}^T \beta^{(k-1)}\right)}\right] =$$

$$= \prod_{m=k}^{K-1} \frac{1}{1 + \exp\left(-\mathbf{x}^T \beta^{(m)}\right)} \left[\frac{1}{1 + \exp\left(\mathbf{x}^T \beta^{(k-1)}\right)}\right].$$

These factorizations enable us to express the loglikelihood summands in (4) in terms of l in (6) by

$$\log p_1(\mathbf{x}, \beta) = -\sum_{m=1}^{K-1} l\left(\mathbf{x}^T \beta^{(m)}\right),$$

$$\log p_k(\mathbf{x}, \beta) = -\sum_{m=k}^{K-1} l\left(\mathbf{x}^T \beta^{(m)}\right) - l\left(-\mathbf{x}^T \beta^{(k-1)}\right), \; k \geq 2 \tag{10}$$

[1] http://en.wikipedia.org/wiki/Dirichlet_process#The_stick-breaking_process

Thus, a piecewise quadratic approximation to the univariate $l(r)$ induces that to the penalized loglikelihood in (4). This allows for a computationally efficient path tracking algorithm that we describe next.

III. Path Tracking Algorithm

The hat quantities in the new minimization objective

$$\min_{\beta} \hat{f}_\lambda(\beta), \tag{11}$$

with

$$\hat{f}_\lambda(\beta) = \hat{f}_0(\beta) + \lambda ||\beta||_1 \tag{12}$$

are obtained from (2)-(4) by using the approximation $\hat{l}(r)$ in place of $l(r)$ in (10) and substituting these into (4):

$$\hat{f}_0(\beta) = \frac{\mu}{2} \beta^T A \beta + \sum_{i=1}^{N} \sum_{k=\max(1, y_i - 1)}^{K-1} \hat{l}(r_{ik}), \tag{13}$$

where $r_{ik} = s_{ik} \mathbf{x}_i \beta^{(k)}$ with $s_{ik} = 1$ if $y_i \leq k$ and $s_{ik} = -1$ if $y_i = k+1$. The gradient $\hat{g}$ and the Hessian $\hat{H}$ of $\hat{f}_0$ are given by

$$\hat{g}\left(\beta^{(k)}\right) = \mu A_k \beta^{(k)} + \sum_{i=1}^{N} \mathbf{1}_{\{y_i \leq k+1\}} s_{ik} \mathbf{x}_i [a(r_{ik}) r_{ik} + b(r_{ik})] \tag{14}$$

and

$$\hat{H}\left(\beta^{(k)}, \beta^{(k)}\right) = \mu A_k + \sum_{i=1}^{N} \mathbf{1}_{\{y_i \leq k+1\}} \mathbf{x}_i \mathbf{x}_i^T a(r_{ik})$$

$$\hat{H}\left(\beta^{(k)}, \beta^{(l)}\right) = \mathbf{0}, \; k \neq l, \tag{15}$$

where $\hat{g}\left(\beta^{(k)}\right)$ refers to the vector of partial derivatives with respect to $\beta^{(k)}$, $\hat{H}\left(\beta^{(k)}, \beta^{(l)}\right)$ is the block of $\hat{H}$ with rows (columns) corresponding to $\beta^{(k)}$ ($\beta^{(l)}$), and we assume that the L_2 penalty matrix A is block-diagonal with the rows and columns of block A_k corresponding to $\beta^{(k)}$. We also used the fact that $\hat{l}$ is piecewise quadratic with coefficients a, b, and c (the latter is not used) as given by (8).

Given the gradient and the Hessian, we can apply the path tracking algorithm of [2], [6]. We now outline its justification and main steps. The Karush-Kuhn-Tucker conditions [6] imply that for any parameter β_j, either it is *inactive* with $\beta_j = 0$ and $|\hat{g}_j| \leq \lambda$ or it is active ($\beta_j \neq 0$) with

$$\hat{g}_j = -\lambda \text{sgn}(\beta_j), \tag{16}$$

where $\text{sgn}(\beta_j)$ is the sign of β_j. Let $\mathcal{A}$ be the set of active parameters and use notation $\mathbf{x}_\mathcal{A}$ and $B_\mathcal{A}$ to denote the vector $x_j, j \in \mathcal{A}$ and the block of matrix B with rows and columns from $\mathcal{A}$ respectively.

Since $\hat{f}_0$ is piecewise quadratic (or from (14) and (15)), a change in β and $\hat{g}$ are related by $\delta\hat{g} = \hat{H}\delta\beta$. Setting $\delta\lambda = -1$, (16) implies that

$$\delta\beta_\mathcal{A} = \hat{H}_\mathcal{A}^{-1} \delta\hat{g}_\mathcal{A} = \hat{H}_\mathcal{A}^{-1} \text{sgn}(\beta_\mathcal{A}) = -\hat{H}_\mathcal{A}^{-1} \text{sgn}(\hat{g}_\mathcal{A}).$$

Thus, as λ decreases, the solution path extends in the direction above implying in particular its piecewise linearity. The linear extension continues until either a) a new parameter j becomes active with $|\hat{g}_j| = \lambda$; b) a parameter reaches zero; or c) the path reaches a piecewise quadratic approximation segment endpoint for any of the

ISBN: 978-988-18210-6-5
ISSN: 2078-0958 (Print); ISSN: 2078-0966 (Online)

r_{ik} in (13). When either of these events happens, the direction of the path is recalculated. The complete algorithm whose description closely parallels that in [2] is given below.

Path Tracking Algorithm

Initialize: $\beta = \mathbf{0}$, $r_{ik} = 0$, $a_i = a(0)\ \forall i$, $\hat{g}\left(\beta^{(k)} = 0\right) = -(1/2)\sum_{i=1}^{N} \mathbf{1}_{\{y_i \leq k+1\}} s_{ik}\mathbf{x}_i$, $\mathcal{A} = \text{argmax}_j |\hat{g}_j|$, $\lambda = \max_j |\hat{g}_j|$, $\delta\beta_{\mathcal{A}} = -\hat{H}_{\mathcal{A}}^{-1}\text{sgn}(\hat{g}_{\mathcal{A}})$, $\delta\beta_{\mathcal{A}^c} = \mathbf{0}$, $\delta r_{ik} = s_{ik}\mathbf{x}_i\delta\beta^{(k)}$ for $i:\ y_i \leq k+1$, and $\delta\hat{g} = \hat{H}\delta\beta$.

While $(\lambda > 0)$

a) $d_1 = \min\{d > 0 :\ |\hat{g}_j + d\delta\hat{g}_j| = \lambda - d, j \in \mathcal{A}^c\}$;

b) $d_2 = \min\{d > 0 : \beta_j + d\delta\beta_j = 0, j \in \mathcal{A}\}$;

c) $d_3 = \min\{d > 0 : r_{ik} + d\delta r_{ik}$ reaches an approximation segment endpoint for some $i, k\}$;

d) $d = \min\{d_1, d_2, d_3\}, \lambda \leftarrow \lambda - d, \beta \leftarrow \beta + \delta\beta, r \leftarrow r + \delta r, \hat{g} \leftarrow d\delta\hat{g}$;

e) if $d = d_1$, add the parameter attaining equality at d to $\mathcal{A}$;

f) if $d = d_2$, remove the coefficient attaining 0 at d from $\mathcal{A}$;

g) if $d = d_3$ at (i^*, k^*), set $a(r_{i^*k^*})$ to the value in the new segment;

h) set $\delta\beta_{\mathcal{A}} = -\hat{H}_{\mathcal{A}}^{-1}\text{sgn}(\hat{g}_{\mathcal{A}}), \delta\beta_{\mathcal{A}^c} = \mathbf{0}$, $\delta r_{ik} = s_{ik}\mathbf{x}_i\delta\beta^{(k)}$ for $i:\ y_i \leq k+1$, and $\delta\hat{g} = \hat{H}\delta\beta$.

At each iteration, most of the time is spent updating the inverse of the Hessian in h). This is typically done by maintaining and updating the Cholesky decomposition of the Hessian. Because of the block-diagonal form of the Hessian in (15), one only needs to manipulate and invert the individual blocks $\hat{H}\left(\beta^{(k)}, \beta^{(k)}\right)$. Also, a single parameter addition and removal in steps e) and f) as well as a single change for a (i^*, k^*) pair in g) entails an update of only a single block of the Hessian since only a single k^* is affected. By contrast, if the canonical link (1) were to be used, it could be seen that differentiating (5) with respect to $\partial\beta^{(k)}\partial\beta^{(l)},\ k \neq l$ would not produce zero. Therefore, one does not obtain a block-diagonal Hessian in the canonical link case.

Finding the d in c) for each (i^*, k^*) pair is a simple closed-form computation. This is thanks to using the univariate approximation whose approximation regions are segments. In the next section we contrast this simplicity with the complexity of using $(K-1)$-variate approximation regions which would ensue if the canonical link (1) were to be employed.

IV. Multivariate Approximation with Canonical Link

To further emphasize the importance of choosing the proposed link function, we pause here to review the difficulties which would be caused by using the canonical link (1). As discussed above, a parallel approach would be to approximate l_K in (7) with a piecewise quadratic function. In the case of $K = 3$, Figure II depicts $l_3(r_1, r_2)$. l_3 can be seen to be highly non-additive. That is, there are no functions $a_1(r_1)$ and $a_2(r_2)$ whose sum adequately approximates $l_3(r_1, r_2)$. For if it were the case, the graphs of $l_3(-5, r_2)$ and $l_3(5, r_2)$ against r_2 marked by arrows in Figure II would have approximately the same shape, namely that of $a_2(r_2)$, and would only differ in their vertical positions. Therefore, an additive expansion similar to (10) enjoyed in the proposed link case which would allow for univariate approximations to $a_1(r_1)$ and $a_2(r_2)$ is not possible.

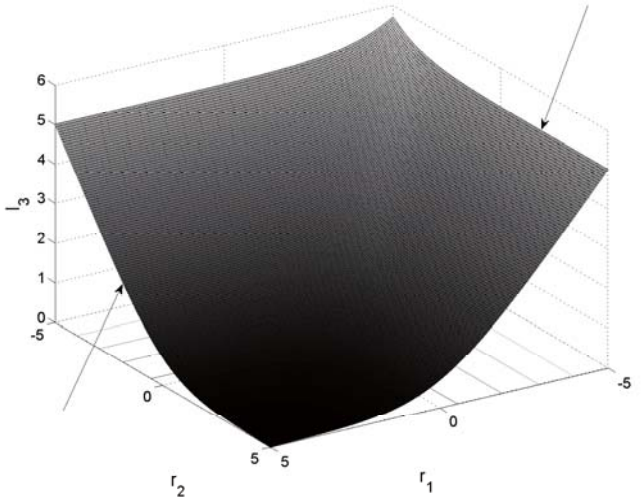

Fig. 2. $l_3(r_1, r_2)$, a special case of (7) for $K = 3$

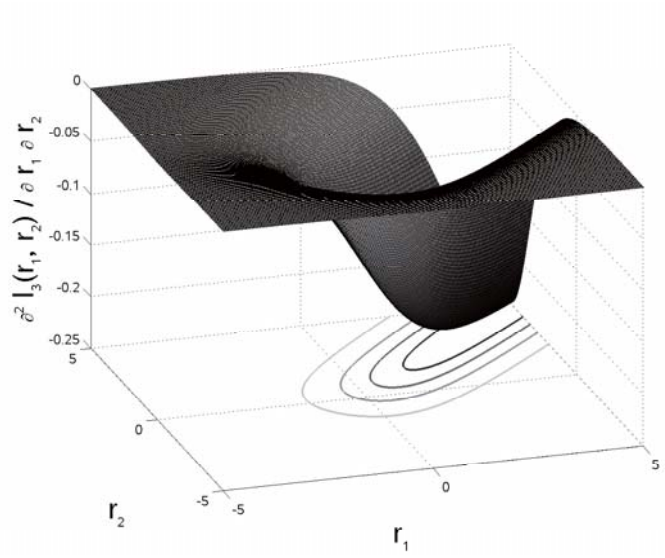

Fig. 3. Mixed partial derivative $\partial l_3(r_1, r_2)/\partial r_1\partial r_2$

Alternatively, one may attempt to develop a $(K-1)$-dimensional piecewise quadratic approximation, where the "pieces" are proper $(K-1)$-dimensional regions. A computational infeasibility of this approach for large training datasets lies in the step c) of the algorithm above, where the distance d to reach the boundary of the current "piece" has to be computed for every training data point. Since tracking multivariate boundaries is tricky regardless of their shapes, this is not a scalable proposition. Nevertheless, let us examine the mixed second derivative $\partial l_3(r_1, r_2)/\partial r_1\partial r_2$ depicted in Figure III. A piecewise quadratic approximation has a piecewise constant second derivative. Therefore, the

ISBN: 978-988-18210-6-5
ISSN: 2078-0958 (Print); ISSN: 2078-0966 (Online)

TABLE II
AVERAGE RUNNING TIMES FOR $mnrfit$ (MAXIMUM LIKELIHOOD ONLY, NO REGULARIZATION), TMPM (L_1 REGULARIZATION FOR A SINGLE λ), AND MULTINOMIAL LARS AS DESCRIBED IN THIS WORK ACROSS SIMULATED DATASETS VARYING IN THE NUMBER OF OBSERVATIONS N, PREDICTOR VARIABLES P, AND CLASS LABELS K.

Datasets			Running Times		
N	P	K	$mnrfit$	TMPM	LARS
100K	50	3	1,004.2	690.9	1,971.7
50K	20	6	174.9	640.4	1,600.8
10K	10	24	133.6	3433.3	1306.2
10K	20	12	104.6	369.9	630.1
10K	100	3	82.4	25.2	80.2
4K	10	12	15.5	216.1	35.6
10K	20	3	6.85	17.89	3.18
2K	10	6	1.87	5.61	1.43
1K	10	3	.421	.562	.090

boundaries of an accurate approximation should generally follow the contour curves in Figure III. A convoluted shape of these contour curves only adds to the boundary tracking problem. These difficulties are expected to become even more pronounced as K grows.

V. Empirical Evaluation

In this section we report the results of a brief computational performance comparison study of our Matlab implementation of the path-tracking algorithm with the proposed link function against two competitors: the standard Matlab multinomial regression routine *mnrfit* and a publicly available Matlab implementation[2] of a two-metric projection method (TMPM) with non-negative variables [7], [9]. Note that *mnrfit* aims at finding the maximum likelihood solution, but does not carry out any L_1 regularization. TMPM does carry out L_1 regularization, but only attempts to solve the regularized problem (2) for a *particular* λ as opposed to finding the full solution path as in our technique. Therefore, our primary question in this study is how much more computationally expensive it is to obtain the full regularized solution path.

We generate simulation datasets for various numbers N, P, and K of observations, predictor variables, and class labels respectively. In order to run these simulations on a commodity laptop with 2GB of RAM, we generate sparse matrices X of size $N \times P$ with the 2.5% fraction of nonzero entries sampled uniformly from $[0, 1]$. Furthermore, upon generating the true coefficient vector β also sampled uniformly from $[0, 1]$, for each row $\mathbf{x}_i$ of X we apply the link function to the linear predictors $\mathbf{x}_i^T \beta^{(k)}, 1 \leq k \leq K-1$ to calculate multinomial probabilities and then draw from the latter to finally generate response y_i. For each triple (N, P, K) we repeat this process m times, each time regenerating X, β, and y, measure the m running times across the three techniques, and then record the average for each technique over the m runs. Note that the same m simulated datasets are used for each of the three techniques.

Table II summarizes the results. When choosing the dataset triples (N, P, K) we tried to balance the number of observations N and the number of parameters $P(K-1)$. This is typical of real-life applications where recording more observations often affords an opportunity to add predictors to the model in an effort to explain more subtle patterns in the data. We used $m = 30$ repetitions for the upper 3 rows and $m = 100$ for the rest. The rows are sorted in the descending average running time for both $mnrfit$ and LARS.

We observe that for the small bottom three configurations LARS is actually the most computationally efficient method. That is, not only does the proposed LARS method carry out L_1 regularization and compute the whole solution path, but it also does this in the least amount of time among the three techniques! Furthermore, having a large number of class labels K hurts TMPM most. It came significantly behind LARS for two out of the three configuration triples with $K \geq 12$. For the medium-size triple (10K, 100, 3) LARS ran about as fast as $mnrfit$, while for the two largest-size triples it performed less than three times worse than TMPM. To put this factor of three into proper context, if one were to vary λ as part of model selection, one would be better off running LARS to obtain the whole solution path than picking 3 different λ's to choose from and running TMPM. Since one typically explores a much greater number of λ's, LARS would be a heavy favorite from the computational perspective over TMPM.

VI. Application to Web Personalization

We apply the proposed technique to a corporate marketing web personalization problem. A *carousel* widget, a yellow popin that appears in the middle right in Figure 4, was deployed across a large segment of Sun Microsystems web properties, including www.sun.com, docs.sun.com, developers.sun.com, and so on. The carousel shows up to 12 items (4 frames, 3 items per frame) pointing to various pieces of marketing collateral, such as white papers, webinars, demos, etc. The goal of presenting the collateral is not only to serve as an initial research tool at the awareness and early consideration stages of a prospective customer lifecycle, but also to let the users self-select into interested parties, submit registrations to possibly become sales leads, and initiate inbound interaction with sales [10].

The key problem is item selection for populating the carousel. To this end, we first apply the proposed technique to classify a user into one of the following categories: Developer, System Administrator, Retail Partner, Federal, Enterprise, Small/Medium Business, Startup, and Student. This multinomial model is fitted on approximately 15K users who classified themselves as part of their engagement with the company. The category was regressed on user actions recorded on and off the web, such as visited web page content, software downloads, registrations, sales interaction, etc.

Second, we undertake two approaches. In the *user segmentation* approach, editors hand-pick appropriate items for each category of users and the system rotates through them giving preference to items that perform better within a given user category. In the *personalization* approach, a logistic LARS model is fitted to predict the success probability for an item in the context of a user. The definition of success varies with an item, but is typically a click, a registration, initializing a chat with sales, etc. The context of a user consists of a) the aforementioned user actions where we also distinguish as to whether or not they have taken place during the current browser session; and b) the inferred multinomial probabilities from the classification above. Thus, these probabilities were explicitly included as *predictors* in the logistic model. Giving

[2] http://www.cs.ubc.ca/ schmidtm/Software/code.html

ISBN: 978-988-18210-6-5
ISSN: 2078-0958 (Print); ISSN: 2078-0966 (Online)

Fig. 4. The Sun Microsystems carousel widget (the yellow popin in the middle right) shows personalized links to marketing collateral.

specific attention to the current browser session aims at making the items relevant to the user based both on what we know about her historically and on what she is engaged in at the moment. Finally, based on the logistic probabilities and the expected payoff of a success as well as some business logic, a decision for populating the carousel is made.

While the personalization approach generates significantly more success events (by about 30%), the simpler segmentation approach fares similarly better than a baseline test of just showing popular items, user category notwithstanding. Additionally, the category probabilities proved to be some of the most significant predictors in the logistic model. Thus, this preclassification constitutes a potent method for improving a model by bringing in highly relevant and compactly packaged information. Overall, the personalization program proved a great success nearly doubling both the rate of sales lead generation from the web and the *pipeline* revenue from these leads.

VII. Conclusion

In this work we extended least angle regression and LASSO (LARS), a state-of-the-art approach to building regularized solution paths, to the multinomial regression setting. To the best of our knowledge, this is the first such extension in the literature. We started with the logistic regression algorithm in [2]. An analogous approach would entail an introduction of $(K-1)$-dimensional approximation regions and computing points of intersections of lines with these regions as part of building solution path extensions. This would result in significant complexity, computational performance degradation, and scalability limitations. Instead, we proposed an efficient method by introducing a new link function based on stick-breaking construction. Using this link function allows us to continue relying on a univariate approximation to a) efficiently carry out path extension distance computations and b) work with a block-diagonal Hessian. The latter permits inversion of only a single block of the Hessian during each step of the algorithm.

Least angle regression methods typically exhibit running times comparable to those of a single-objective optimization method, such as unpenalized maximum likelihood estimation or regularization for a single value of the regularization parameter. We were able to verify this superior quality/scalability tradeoff for the proposed technique. Our research-quality Matlab implementation is available from the author.

References

[1] T. Hastie, R. Tibshirani, and J. H. Friedman, *The Elements of Statistical Learning: Data Mining, Inference, and Prediction. 2nd Edition.* New York: Springer-Verlag, 2009.
[2] S. Keerthi and S. Shevade, "A fast tracking algorithm for generalized lars/lasso," *IEEE Transactions on Neural Networks*, vol. 18, no. 6, pp. 1826–1830, 2007.
[3] H. Zho and T. Hastie, "Regularization and variable selection via the elastic net," *J. Roy. Statist. Soc. B*, vol. 67, pp. 301–320, 2005.
[4] M. Osborne, B. Presnell, and B. Turlach, "A new approach to variable selection in least squares problems," *IMA Journal of Numerical Analysis*, vol. 20, pp. 389–404, 2000.
[5] B. Efron, T. Hastie, I. Johnstone, and R. Tibshirani, "Least angle regression," *Annals of Statistics*, vol. 32, no. 2, pp. 407–499, 2004.
[6] S. Rosset and J. Zhu, "Piecewise linear regularized solution paths," *The Annals of Statistics*, vol. 35, pp. 1012–1030, 2007.
[7] M. Schmidt, G. Fung, and R. Rosales, "Fast optimization methods for l1 regularization: A comparative study and two new approaches," *Lecture Notes in Computer Science*, vol. 4701, pp. 286–297, 2007.
[8] R. Tibshirani, "Regression shrinkage and selection via the lasso," *J. Roy. Statist. Soc. B*, vol. 58, pp. 267–288, 1996.
[9] E. Gafni and D. Bertsekas, "Two-metric projection methods for constrained optimization," *SIAM J. Contr. Optim.*, vol. 22, no. 6, pp. 936–964, 1984.
[10] B. Halligan and D. Shah, *Inbound marketing : get found using Google, social media, and blogs*. Wiley, 2010. [Online]. Available: http://www.worldcat.org/isbn/9780470499313

ISBN: 978-988-18210-6-5
ISSN: 2078-0958 (Print); ISSN: 2078-0966 (Online)