Likelihood calculations for vsn

Wolfgang Huber

April 30, 2008

Contents

1 Introduction 1
2 Setup and Notation 1
3 Likelihood for Incremental Normalization 2
4 Profile Likelihood 3
5 Summary 5

1 Introduction

This vignette contains the computations that underlie the numerical code of vsn. If you are a new user and looking for an introduction on how to use vsn, please refer to the vignette Robust calibration and variance stabilization with vsn, which is provided separately.

2 Setup and Notation

Consider the model

\[
\text{arsinh} \left( f(b_i) \cdot y_{ki} + a_i \right) = \mu_k + \varepsilon_{ki}
\]  

(1)

where \( \mu_k \), for \( k = 1, \ldots, n \), and \( a_i, b_i \), for \( i = 1, \ldots, d \) are real-valued parameters, \( f \) is a function \( \mathbb{R} \rightarrow \mathbb{R} \) (see below), and \( \varepsilon_{ki} \) are i.i.d. Normal with mean 0 and variance \( \sigma^2 \). \( y_{ki} \) are the data. In applications to \( \mu \text{array} \) data, \( k \) indexes the features and \( i \) the arrays and/or colour channels.

Examples for \( f \) are \( f(b) = b \) and \( f(b) = e^b \). The former is the most obvious choice; in that case we will usually need to require \( b_i > 0 \). The choice \( f(b) = e^b \) assures that the factor in front of \( y_{ki} \) is positive for all \( b \in \mathbb{R} \), and as it turns out, simplifies some of the computations.
In the following calculations, I will also use the notation
\[
Y \equiv Y(y, a, b) = f(b) \cdot y + a
\] (2)
\[
h \equiv h(y, a, b) = \text{arsinh}(f(b) \cdot y + a).
\] (3)

The probability of the data \((y_{ki})_{k=1\ldots n, i=1\ldots d}\) lying in a certain volume element of \(y\)-space (hyperrectangle with sides \([y_{ki}^\alpha, y_{ki}^\beta}]\) is
\[
P = \prod_{k=1}^{n} \prod_{i=1}^{d} \int_{y_{ki}^\alpha}^{y_{ki}^\beta} \frac{1}{\sqrt{2\pi}\sigma^2} \exp\left(-\frac{(x - \mu)^2}{2\sigma^2}\right) \cdot \frac{dh}{dy}\,(y_{ki}).
\] (4)

where \(\mu_k\) is the expectation value for feature \(k\) and \(\sigma^2\) the variance.

With
\[
p_{\text{Normal}}(x, \mu, \sigma^2) = \frac{1}{\sqrt{2\pi}\sigma^2} \exp\left(-\frac{(x - \mu)^2}{2\sigma^2}\right)
\] (5)
the likelihood is
\[
L = \left(\frac{1}{\sqrt{2\pi}\sigma^2}\right)^{nd} \prod_{k=1}^{n} \prod_{i=1}^{d} \exp\left(-\frac{(h(y_{ki}) - \mu_k)^2}{2\sigma^2}\right) \cdot \frac{dh}{dy}\,(y_{ki}).
\] (6)

For the following, I will need the derivatives
\[
\frac{\partial Y}{\partial a} = 1
\] (7)
\[
\frac{\partial Y}{\partial b} = y \cdot f'(b)
\] (8)
\[
\frac{dh}{dy} = \frac{f(b)}{\sqrt{1 + (f(b)y + a)^2}} = \frac{f(b)}{\sqrt{1 + Y^2}}
\] (9)
\[
\frac{\partial h}{\partial a} = \frac{1}{\sqrt{1 + Y^2}},
\] (10)
\[
\frac{\partial h}{\partial b} = \frac{y}{\sqrt{1 + Y^2}} \cdot f'(b).
\] (11)

Note that for \(f(b) = b\), we have \(f'(b) = 1\), and for \(f(b) = e^b\), \(f'(b) = f(b) = e^b\).

### 3 Likelihood for Incremental Normalization

Here, *incremental normalization* means that the model parameters \(\mu_1, \ldots, \mu_n\) and \(\sigma^2\) are already known from a fit to a previous set of \(\mu\) arrays, i.e. a set of reference arrays. See Section 4 for the profile likelihood approach that is used if \(\mu_1, \ldots, \mu_n\) and \(\sigma^2\) are not
known and need to be estimated from the same data. Versions $\geq 2.0$ of the \textit{vsn} package implement both of these approaches; in versions $1.X$ only the profile likelihood approach was implemented, and it was described in the initial publication [1].

First, let us note that the likelihood (6) is simply a product of independent terms for different $i$. We can optimize the parameters $(a_i, b_i)$ separately for each $i = 1, \ldots, d$. From the likelihood (6) we get the $i$-th negative log-likelihood

$$ -\log(L) = \sum_{i=1}^{d} -LL_i $$ (12)

$$ -LL_i = \frac{n}{2} \log(2\pi\sigma^2) + \sum_{k=1}^{n} \left( \frac{(h(y_{ki}) - \mu_k)^2}{2\sigma^2} + \log \frac{1 + Y_{ki}^2}{f(b_i)} \right) $$ (13)

$$ = \frac{n}{2} \log(2\pi\sigma^2) - n \log f(b_i) + \sum_{k=1}^{n} \left( \frac{(h(y_{ki}) - \mu_k)^2}{2\sigma^2} + \frac{1}{2} \log(1 + Y_{ki}^2) \right) $$ (14)

This is what we want to optimize as a function of $a_i$ and $b_i$. The optimizer benefits from the derivatives. The derivative with respect to $a_i$ is

$$ \frac{\partial}{\partial a_i}(-LL_i) = \sum_{k=1}^{n} \left( \frac{h(y_{ki}) - \mu_k}{\sigma^2} + \frac{Y_{ki}}{\sqrt{1 + Y_{ki}^2}} \right) \cdot \frac{1}{\sqrt{1 + Y_{ki}^2}} $$ (15)

and with respect to $b_i$

$$ \frac{\partial}{\partial b_i}(-LL_i) = -n \frac{f'(b_i)}{f(b_i)} + \sum_{k=1}^{n} \left( \frac{h(y_{ki}) - \mu_k}{\sigma^2} + \frac{Y_{ki}}{\sqrt{1 + Y_{ki}^2}} \right) \cdot \frac{y_{ki}}{\sqrt{1 + Y_{ki}^2}} \cdot f'(b_i) $$ (16)

Here, I have introduced the following shorthand notation for the “intermediate results” terms

$$ r_{ki} = h(y_{ki}) - \mu_k $$ (17)

$$ A_{ki} = \frac{1}{\sqrt{1 + Y_{ki}^2}} $$ (18)

Variables for these intermediate values are also used in the C code to organise the computations of the gradient.
4 Profile Likelihood

If $\mu_1, \ldots, \mu_n$ and $\sigma^2$ are not already known, we can plug in their maximum likelihood estimates, obtained from optimizing $LL$ for $\mu_1, \ldots, \mu_n$ and $\sigma^2$:

$$\hat{\mu}_k = \frac{1}{d} \sum_{j=1}^{d} h(y_{kj}) \quad (19)$$

$$\hat{\sigma}^2 = \frac{1}{nd} \sum_{k=1}^{n} \sum_{j=1}^{d} (h(y_{kj}) - \hat{\mu}_k)^2 \quad (20)$$

into the negative log-likelihood. The result is called the negative profile log-likelihood

$$-PLL = \frac{nd}{2} \log(2\pi \hat{\sigma}^2) + \frac{nd}{2} - n \sum_{j=1}^{d} \log f(b_j) + \frac{1}{2} \sum_{k=1}^{n} \sum_{j=1}^{d} \log \sqrt{1 + Y_{kj}^2}. \quad (21)$$

Note that this no longer decomposes into a sum of terms for each $j$ that are independent of each other – the terms for different $j$ are coupled through Equations (19) and (20). We need the following derivatives.

$$\frac{\partial \hat{\sigma}^2}{\partial a_i} = \frac{2}{nd} \sum_{k=1}^{n} r_{ki} \frac{\partial h(y_{ki})}{\partial a_i} = \frac{2}{nd} \sum_{k=1}^{n} r_{ki} A_{ki} \quad (22)$$

$$\frac{\partial \hat{\sigma}^2}{\partial b_i} = \frac{2}{nd} f'(b_i) \sum_{k=1}^{n} r_{ki} A_{ki} y_{ki} \quad (23)$$

So, finally

$$\frac{\partial}{\partial a_i} (-PLL) = \frac{nd}{2\hat{\sigma}^2} \cdot \frac{\partial \hat{\sigma}^2}{\partial a_i} + \sum_{k=1}^{n} A_{ki}^2 Y_{ki}$$

$$= \sum_{k=1}^{n} \left( \frac{r_{ki}}{\hat{\sigma}^2} + A_{ki} Y_{ki} \right) A_{ki} \quad (24)$$

$$\frac{\partial}{\partial b_i} (-PLL) = -n \frac{f'(b_i)}{f(b_i)} + f'(b_i) \sum_{k=1}^{n} \left( \frac{r_{ki}}{\hat{\sigma}^2} + A_{ki} Y_{ki} \right) A_{ki} y_{ki} \quad (25)$$
5 Summary

Likelihoods, from Equations (12) and (21):

\[-LL_i = \frac{n}{2} \log(2\pi \sigma^2) + \sum_{k=1}^{n} \frac{(h(y_{ki}) - \mu_k)^2}{2\sigma^2} - n \log f(b_i) + \frac{1}{2} \sum_{k=1}^{n} \log(1 + Y_{ki}^2)\] (26)

\[-PLL = \frac{nd}{2} \log(2\hat{\sigma}^2) + \sum_{i=1}^{d} \left( -n \log f(b_i) + \frac{1}{2} \sum_{k=1}^{n} \log(1 + Y_{ki}^2) \right)\] (27)

The computations in the C code are organised into steps for computing the terms “scale”, “residuals” and “jacobian”.

Partial derivatives with respect to \(a_i\), from Equations (15) and (24):

\[\frac{\partial}{\partial a_i}(-LL_i) = \sum_{k=1}^{n} \left( \frac{r_{ki}}{\sigma^2} + A_{ki}Y_{ki} \right) A_{ki}\] (28)

\[\frac{\partial}{\partial a_i}(-PLL) = \sum_{k=1}^{n} \left( \frac{r_{ki}}{\hat{\sigma}^2} + A_{ki}Y_{ki} \right) A_{ki}\] (29)

Partial derivatives with respect to \(b_i\), from Equations (16) and (25):

\[\frac{\partial}{\partial b_i}(-LL_i) = -n \frac{f'(b_i)}{f(b_i)} + f'(b_i) \sum_{k=1}^{n} \left( \frac{r_{ki}}{\sigma^2} + A_{ki}Y_{ki} \right) A_{ki}y_{ki}\] (30)

\[\frac{\partial}{\partial b_i}(-PLL) = -n \frac{f'(b_i)}{f(b_i)} + f'(b_i) \sum_{k=1}^{n} \left( \frac{r_{ki}}{\hat{\sigma}^2} + A_{ki}Y_{ki} \right) A_{ki}y_{ki}\] (31)

Note that the terms have many similarities – this is used in the implementation in the C code.

References
