VARIATIONAL BAYESIAN INFERENCE
FOR LINEAR AND LOGISTIC REGRESSION

JAN DRUGOWITSCH

ABSTRACT. The article describe the model, derivation, and implementation of variational Bayesian inference for linear and logistic regression, both with and without automatic relevance determination. It has the dual function of acting as a tutorial for the derivation of variational Bayesian inference for simple models, as well as documenting, and providing brief examples for the MATLAB functions that implement this inference. These functions are freely available online.

1. INTRODUCTION

Linear and logistic regression are essential workhorses of statistical analysis, whose Bayesian treatment has received much recent attention (Gelman et al., 2013; Bishop, 2006; Murphy, 2012; Hastie et al., 2011). These allow specifying the a-priori uncertainty and infer a-posteriori uncertainty about regression coefficients explicity and hierarchically, by, for example, specifying how uncertain we are a-priori that these coefficients are small. However, Bayesian inference in such hierarchical models quickly becomes intractable, such that recent effort has focused on approximate inference, like Markov Chain Monte Carlo methods (Gilks et al., 1995), or variational Bayesian approximation (Beal, 2003; Bishop, 2006; Murphy, 2012).

Here, we describe such a variational treatment and implementation of Bayesian hierarchical models for both linear and logistic regression. Even though neither the statistical models nor their Bayesian approximation are particularly novel, the article provides a tutorial-style introduction to the derivation of their algorithms, together with a MATLAB implementation of these algorithms. As such, it bridges the gap between theory and practice of derivation and implementation.

The presentation of the variational inference derivation is closely aligned to that of Bishop (2006), but with essential differences. Specifically, both models include a variant with automatic relevance determination (ARD), which consists of assigning an individual hyper-prior to each regression coefficient separately. These hyper-priors are adjusted to eventually prune irrelevant coefficients (Wipf and Nagarajan, 2007) without the need for a separate validation set, unlike comparable sparsity-inducing methods like the Lasso (Tibshirani, 1996). Bishop (2006) describes ARD only in the context of type-II maximum likelihood (MacKay, 1992; Neal, 1996; Tipping, 2001), where it (hyper-)parameters are tuned by maximizing the marginal likelihood (or *model evidence*). Here, instead, we apply the full Bayesian treatment, and find the ARD hyper-posteriors by variational Bayesian inference.

The model underlying linear regression is closely related to Griffin and Brown (2010), where the authors analyze the influence of prior choice on regression coefficient shrinkage. They promote priors in the form of scale mixtures of zero-mean normals, which allow for a larger difference in regression coefficients than would

be possible under a normal prior. The authors proceed by suggesting a normal-gamma prior and analyze its shrinkage properties in detail. The prior used in this work is also member of the scale mixtures of normals, and thus shares its advantageous properties. However, instead of a normal-gamma prior, this work uses a normal inverse-gamma prior in combination with another inverse-gamma hyper-prior, as its conjugacy to the likelihood is advantageous for use with variational Bayesian inference. The same analysis performed by Griffin and Brown (2010) for the normal-gamma case should be amendable to the normal inverse-gamma case, but has yet to be performed.

The article is structured as follows. It first described linear regression, followed by logistic regression. For each of these, it first introduces the generative model, followed by deriving the variational Bayesian approximation. After that it introduces the ARD variants, together with their required changes to variational inference. This is followed by a detailed description of the MATLAB functions that implement this inference, and a set of examples that demonstrate their use.

All functions can be found at `https://github.com/DrugowitschLab/vb_linear` and `https://github.com/DrugowitschLab/vb_logit`.

2. Linear regression

This section describes inference in a model performing linear regression. It is similar to that in Bishop (2006) by assuming a hyper-prior α on the regression coefficients $\boldsymbol{w}$. However, rather than just inferring the posterior $\boldsymbol{w}$, as done in Bishop (2006), it additionally puts an inverse-gamma prior on the variance τ^{-1} and infers the joint posterior of $\boldsymbol{w}$ and τ jointly. Furthermore, Bishop (2006) utilizes type-II maximum likelihood to deal with automatic relevance determination for linear regression. Here, we use the variational Bayesian approximation instead.

2.1. **The model.** The model assumes a linear relation between D-dimensional inputs $\boldsymbol{x}$ and outputs y and constant-variance Gaussian noise, such that the data likelihood is given by

$$\mathsf{P}(y|\boldsymbol{x}, \boldsymbol{w}, \tau) = \mathcal{N}(y|\boldsymbol{w}^\top \boldsymbol{x}, \tau^{-1}) = \left(\frac{\tau}{2\pi}\right)^{1/2} \exp\left(-\frac{\tau}{2}(y - \boldsymbol{w}^\top \boldsymbol{x})^2\right). \tag{1}$$

Given all data $\mathcal{D} = \{\boldsymbol{X}, \boldsymbol{Y}\}$, with $\boldsymbol{X} = \{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_N\}$ and $\boldsymbol{Y} = \{y_1, \ldots, y_N\}$, the data likelihood is

$$\mathsf{P}(\boldsymbol{Y}|\boldsymbol{X}, \boldsymbol{w}, \tau) = \prod_n \mathsf{P}(y_n|\boldsymbol{x}_n, \boldsymbol{w}, \tau). \tag{2}$$

The prior on $\boldsymbol{w}$ and τ is conjugate normal inverse-gamma

$$\begin{aligned} \mathsf{P}(\boldsymbol{w}, \tau|\alpha) &= \mathcal{N}(\boldsymbol{w}|0, (\tau\alpha)^{-1}\boldsymbol{I})\mathrm{Gam}(\tau|a_0, b_0) \\ &= \left(\frac{\alpha}{2\pi}\right)^{D/2} \frac{b_0^{a_0}}{\Gamma(a_0)} \tau^{D/2 + a_0 - 1} \exp\left(-\frac{\tau}{2}(\alpha\boldsymbol{w}^\top\boldsymbol{w} + 2b_0)\right), \end{aligned} \tag{3}$$

parametrized by α. As in Griffin and Brown (2010), this prior is member of the scale mixtures of normals. In this prior, τ appears as τ^{-1} in the variance of the zero-mean normal on $\boldsymbol{w}$. Due to the gamma on τ, this τ^{-1} is inverse-gamma with shape a_0, scale b_0, and moments $\mathsf{E}\left(\tau^{-1}\right) = b_0/(a_0 - 1)$ for $a_0 > 1$ and $\mathsf{VAR}\left(\tau^{-1}\right) = b_0^2/\left((a_0 - 1)^2(a_0 - 2)\right)$ for $a_0 > 2$. The hyper-parameter α is assigned the hyper-prior

$$\mathsf{P}(\alpha) = \mathrm{Gam}(\alpha|c_0, d_0) = \frac{1}{\Gamma(c_0)} d_0^{c_0} \alpha^{c_0 - 1} \exp(-d_0\alpha), \tag{4}$$

with moments of α^{-1} analogous to τ^{-1}. Due to the hyper-prior, there is no analytic solution to the posteriors, and variational Bayesian inference will be applied.

2.2. **Variational Bayesian inference.** The variational posteriors are found by maximizing the variational bound

$$\mathcal{L}(\mathsf{Q}) = \iiint \mathsf{Q}(\boldsymbol{w}, \tau, \alpha) \ln \frac{\mathsf{P}(\boldsymbol{Y}|\boldsymbol{X}, \boldsymbol{w}, \tau)\mathsf{P}(\boldsymbol{w}, \tau|\alpha)\mathsf{P}(\alpha)}{\mathsf{Q}(\boldsymbol{w}, \tau, \alpha)} \mathrm{d}\boldsymbol{w}\mathrm{d}\tau\mathrm{d}\alpha \le \ln \mathsf{P}(\mathcal{D}), \quad (5)$$

where $\mathsf{P}(\mathcal{D})$ is the model evidence. To maximize this bound, we assume that the variational distribution $\mathsf{Q}(\boldsymbol{w}, \tau, \alpha)$, which approximates the posterior $\mathsf{P}(\boldsymbol{w}, \tau, \alpha|\mathcal{D})$, factors into $\mathsf{Q}(\boldsymbol{w}, \tau)\mathsf{Q}(\alpha)$.

Using standard results from variational Bayesian inference (Beal, 2003; Bishop, 2006), the variational posterior for $\boldsymbol{w}, \tau$ that maximizes the variational bound $\mathcal{L}(\mathsf{Q})$ while holding $\mathsf{Q}(\alpha)$ fixed, is given by

$$\begin{aligned}
\ln \mathsf{Q}^*(\boldsymbol{w}, \tau) &= \ln \mathsf{P}(\boldsymbol{Y}|\boldsymbol{X}, \boldsymbol{w}, \tau) + \mathsf{E}_\alpha(\ln \mathsf{P}(\boldsymbol{w}, \tau|\alpha)) + \mathrm{const.} \\
&= \left(\frac{D}{2} + a_0 - 1 + \frac{N}{2}\right) \ln \tau \\
&\quad - \frac{\tau}{2}\left(\boldsymbol{w}^\top\left(\mathsf{E}_\alpha(\alpha)\boldsymbol{I} + \sum_n \boldsymbol{x}_n\boldsymbol{x}_n^\top\right)\boldsymbol{w} \right. \\
&\quad \left. + \sum_n y_n^2 - 2\boldsymbol{w}^\top\sum_n \boldsymbol{x}_n y_n + 2b_0\right) + \mathrm{const.} \\
&= \ln \mathcal{N}(\boldsymbol{w}|\boldsymbol{w}_N, \tau^{-1}\boldsymbol{V}_N)\mathrm{Gam}(\tau|a_N, b_N), \quad (6)
\end{aligned}$$

with

$$\boldsymbol{V}_N^{-1} = \mathsf{E}_\alpha(\alpha)\boldsymbol{I} + \sum_n \boldsymbol{x}_n\boldsymbol{x}_n^\top, \quad (7)$$

$$\boldsymbol{w}_N = \boldsymbol{V}_N \sum_n \boldsymbol{x}_n y_n, \quad (8)$$

$$a_N = a_0 + \frac{N}{2}, \quad (9)$$

$$\begin{aligned}
b_N &= b_0 + \frac{1}{2}\left(\sum_n y_n^2 - \boldsymbol{w}_N^\top\boldsymbol{V}_N^{-1}\boldsymbol{w}_N\right) \\
&= b_0 + \frac{1}{2}\left(\sum_n (y_n - \boldsymbol{w}_N^\top\boldsymbol{x}_n)^2 + \mathsf{E}_\alpha(\alpha)\boldsymbol{w}_N^\top\boldsymbol{w}_N\right). \quad (10)
\end{aligned}$$

The variational posterior for α is

$$\begin{aligned}
\ln \mathsf{Q}^*(\alpha) &= \mathsf{E}_{\boldsymbol{w}, \tau}(\ln \mathsf{P}(\boldsymbol{w}, \tau|\alpha)) + \ln \mathsf{P}(\alpha) + \mathrm{const.} \\
&= \left(c_0 - 1 + \frac{D}{2}\right) \ln \alpha - \alpha\left(d_0 + \frac{1}{2}\mathsf{E}_{\boldsymbol{w}, \tau}(\tau\boldsymbol{w}^\top\boldsymbol{w})\right) + \mathrm{const.} \\
&= \ln \mathrm{Gam}(\alpha|c_N, d_N), \quad (11)
\end{aligned}$$

with

$$c_N = c_0 + \frac{D}{2}, \quad (12)$$

$$d_N = d_0 + \frac{1}{2}\mathsf{E}_{\boldsymbol{w}, \tau}(\tau\boldsymbol{w}^\top\boldsymbol{w}). \quad (13)$$

The expectations are evaluated with respect to the variational posterior and are given by

$$\mathsf{E}_{\boldsymbol{w},\tau}(\tau \boldsymbol{w}^\top \boldsymbol{w}) \;=\; \frac{a_N}{b_N} \boldsymbol{w}_N^\top \boldsymbol{w}_N + \mathrm{Tr}(\boldsymbol{V}_N), \tag{14}$$

$$\mathsf{E}_{\alpha}(\alpha) \;=\; \frac{c_N}{d_N}. \tag{15}$$

The variational bound itself consists of

$$
\begin{aligned}
\mathcal{L}(\mathsf{Q}) \;=\;& \mathsf{E}_{\boldsymbol{w},\tau}(\ln \mathsf{P}(\boldsymbol{Y}|\boldsymbol{X},\boldsymbol{w},\tau)) + \mathsf{E}_{\boldsymbol{w},\tau,\alpha}(\ln \mathsf{P}(\boldsymbol{w},\tau|\alpha)) \\
& + \mathsf{E}_{\alpha}(\ln \mathsf{P}(\alpha)) - \mathsf{E}_{\boldsymbol{w},\tau}(\ln \mathsf{Q}(\boldsymbol{w},\tau)) \\
& - \mathsf{E}_{\alpha}(\ln \mathsf{Q}(\alpha)),
\end{aligned} \tag{16}
$$

$$
\begin{aligned}
\mathsf{E}_{\boldsymbol{w},\tau}(\ln \mathsf{P}(\boldsymbol{Y}|\boldsymbol{X},\boldsymbol{w},\tau)) \;=\;& \frac{N}{2}\left(\psi(a_N) - \ln b_N - \ln 2\pi\right) \\
& - \frac{1}{2}\sum_n \left(\frac{a_N}{b_N}(y_n - \boldsymbol{w}_N^\top \boldsymbol{x}_n)^2 + \boldsymbol{x}_n^\top \boldsymbol{V}_N \boldsymbol{x}_n\right),
\end{aligned} \tag{17}
$$

$$
\begin{aligned}
\mathsf{E}_{\boldsymbol{w},\tau,\alpha}(\ln \mathsf{P}(\boldsymbol{w},\tau|\alpha)) \;=\;& \frac{D}{2}\left(\psi(a_N) - \ln b_N + \psi(c_N) - \ln d_N - \ln 2\pi\right) \\
& - \frac{1}{2}\frac{c_N}{d_N}\left(\frac{a_N}{b_N}\boldsymbol{w}_N^\top \boldsymbol{w}_N + \mathrm{Tr}(\boldsymbol{V}_N)\right) \\
& - \ln \Gamma(a_0) + a_0 \ln b_0 \\
& + (a_0 - 1)(\psi(a_N) - \ln b_N) - b_0 \frac{a_N}{b_N},
\end{aligned} \tag{18}
$$

$$
\begin{aligned}
\mathsf{E}_{\alpha}(\ln \mathsf{P}(\alpha)) \;=\;& -\ln \Gamma(c_0) + d_0 \ln c_0 \\
& + (c_0 - 1)(\psi(c_N) - \ln d_N) - d_0 \frac{c_N}{d_N},
\end{aligned} \tag{19}
$$

$$
\begin{aligned}
\mathsf{E}_{\boldsymbol{w},\tau}(\ln \mathsf{Q}(\boldsymbol{w},\tau)) \;=\;& \frac{D}{2}(\psi(a_N) - \ln b_N - \ln 2\pi - 1) - \frac{1}{2}\ln|\boldsymbol{V}_N| \\
& - \ln \Gamma(a_N) + a_N \ln b_N \\
& + (a_N - 1)(\psi(a_N) - \ln b_N) - a_N,
\end{aligned} \tag{20}
$$

$$\mathsf{E}_{\alpha}(\ln \mathsf{Q}(\alpha)) \;=\; -\ln \Gamma(c_N) + (c_N - 1)\psi(c_N) + \ln d_N - c_N. \tag{21}$$

In combination, this gives

$$
\begin{aligned}
\mathcal{L}(\mathsf{Q}) \;=\;& -\frac{N}{2}\ln 2\pi - \frac{1}{2}\sum_n \left(\frac{a_N}{b_N}(y_n - \boldsymbol{w}_N^\top \boldsymbol{x}_n)^2 + \boldsymbol{x}_n^\top \boldsymbol{V}_N \boldsymbol{x}_n\right) + \frac{1}{2}\ln|\boldsymbol{V}_N| + \frac{D}{2} \\
& - \ln \Gamma(a_0) + a_0 \ln b_0 - b_0 \frac{a_N}{b_N} + \ln \Gamma(a_N) - a_N \ln b_N + a_N \\
& - \ln \Gamma(c_0) + c_0 \ln d_0 + \ln \Gamma(c_N) - c_N \ln d_N
\end{aligned} \tag{22}
$$

This bound is maximized by iterating over the updates for $\boldsymbol{V}_N$, $\boldsymbol{w}_N$, a_N, b_N, c_N, and d_N until $\mathcal{L}(\mathsf{Q})$ reaches a plateau.

2.3. **Predictive density.** The predictive density is evaluated by approximating the posterior $\mathsf{P}(\boldsymbol{w}, \tau | \mathcal{D})$ by its variational counterpart $\mathsf{Q}(\boldsymbol{w}, \tau)$, to get

$$
\begin{aligned}
\mathsf{P}(y|\boldsymbol{x}, \mathcal{D}) &= \iint \mathsf{P}(y|\boldsymbol{x}, \boldsymbol{w}, \tau) \mathsf{P}(\boldsymbol{w}, \tau | \mathcal{D}) \mathrm{d}\boldsymbol{w} \mathrm{d}\tau \\
&\approx \iint \mathsf{P}(y|\boldsymbol{x}, \boldsymbol{w}, \tau) \mathsf{Q}(\boldsymbol{w}, \tau) \mathrm{d}\boldsymbol{w} \mathrm{d}\tau \\
&= \iint \mathcal{N}(y|\boldsymbol{w}^\top \boldsymbol{x}, \tau^{-1}) \mathcal{N}(\boldsymbol{w}|\boldsymbol{w}_N, \tau^{-1} \boldsymbol{V}_N) \mathrm{Gam}(\tau|a_N, b_N) \mathrm{d}\boldsymbol{w} \mathrm{d}\tau \\
&= \int \mathcal{N}(y|\boldsymbol{w}_N^\top \boldsymbol{x}, \tau^{-1}(1 + \boldsymbol{x}^\top \boldsymbol{V}_N \boldsymbol{x})) \mathrm{Gam}(\tau|a_N, b_N) \mathrm{d}\tau \\
&= \mathrm{St}\left(y|\boldsymbol{w}_N^\top \boldsymbol{x}, (1 + \boldsymbol{x}^\top \boldsymbol{V}_N \boldsymbol{x})^{-1} \frac{a_N}{b_N}, 2a_N\right),
\end{aligned}
\tag{23}
$$

where standard results of convolving Gaussians with other Gaussians and Gamma distributions where used (Bishop, 2006; Murphy, 2012). The resulting distribution is a Student's t distribution with mean $\boldsymbol{w}_N^\top \boldsymbol{x}$, precision $(1 + \boldsymbol{x}^\top \boldsymbol{V}_N \boldsymbol{x})^{-1} a_N / b_N$, and $2a_N$ degrees of freedom. The resulting predictive variance is $(1 + \boldsymbol{x}^\top \boldsymbol{V}_N \boldsymbol{x}) b_N / (a_N - 1)$.

2.4. **Using automatic relevance determination.** Automatic relevance determination (ARD) determines the relevance of the elements of the input to determine the output by assigning a separate shrinkage prior to each element of the weight vector, which is in turn adjusted by a hyper-prior. While the data likelihood remains unchanged, the prior on $\boldsymbol{w}, \tau$ is modified to be

$$
\begin{aligned}
\mathsf{P}(\boldsymbol{w}, \tau | \boldsymbol{\alpha}) &= \mathcal{N}(\boldsymbol{w}|\boldsymbol{0}, (\tau \boldsymbol{A})^{-1}) \mathrm{Gam}(\tau|a_0, b_0) \\
&= \frac{|\boldsymbol{A}|^{1/2}}{\sqrt{2\pi}^D} \frac{b_0^{a_0}}{\Gamma(a_0)} \tau^{D/2 + a_0 - 1} \exp\left(-\frac{\tau}{2}(\boldsymbol{w}^\top \boldsymbol{A} \boldsymbol{w} + 2b_0)\right),
\end{aligned}
\tag{24}
$$

where the vector $\boldsymbol{\alpha} = (\alpha_1, \ldots, \alpha_D)^\top$ forms the diagonal of $\boldsymbol{A}$. All of the α's are independent, such that the hyper-prior is given by

$$
\mathsf{P}(\boldsymbol{\alpha}) = \prod_i \mathrm{Gam}(\alpha_i | c_0, d_0) = \prod_i \frac{1}{\Gamma(c_0)} d_0^{c_0} \alpha_i^{c_0 - 1} \exp(-d_0 \alpha_i).
\tag{25}
$$

Variational Bayesian inference is performed as before, resulting in the variational posteriors

$$
\mathsf{Q}^*(\boldsymbol{w}, \tau) = \mathcal{N}(\boldsymbol{w}|\boldsymbol{w}_N, \tau^{-1} \boldsymbol{V}_N) \mathrm{Gam}(\tau|a_N, b_N), \quad \mathsf{Q}^*(\boldsymbol{\alpha}) = \prod_i \mathrm{Gam}(\alpha_i | c_N, d_{Ni}),
\tag{26}
$$

with parameters

$$
\boldsymbol{V}_N^{-1} \;=\; \mathsf{E}_\alpha(\boldsymbol{A}) + \sum_n \boldsymbol{x}_n \boldsymbol{x}_n^\top, \tag{27}
$$

$$
\boldsymbol{w}_N \;=\; \boldsymbol{V}_N \sum_n \boldsymbol{x}_n y_n, \tag{28}
$$

$$
a_N \;=\; a_0 + \frac{N}{2} \tag{29}
$$

$$
b_N \;=\; b_0 + \frac{1}{2}\left(\sum_n y_n^2 - \boldsymbol{w}_N^\top \boldsymbol{V}_N^{-1} \boldsymbol{w}_N\right)
$$

$$
\;=\; b_0 + \frac{1}{2}\left(\sum_n (\boldsymbol{w}_N^\top \boldsymbol{x}_N - y_n)^2 + \boldsymbol{w}_N^\top \mathsf{E}_\alpha(\boldsymbol{A})\boldsymbol{w}_N\right), \tag{30}
$$

$$
c_N \;=\; c_0 + \frac{1}{2}, \tag{31}
$$

$$
d_{Ni} \;=\; d_0 + \frac{1}{2}\mathsf{E}_{\boldsymbol{w},\tau}(\tau w_i^2), \tag{32}
$$

with expectations $\mathsf{E}_{\boldsymbol{w},\tau}(\tau w_i^2) = w_{Ni}^2 a_N/b_N + (\boldsymbol{V}_N)_{ii}$, and $\mathsf{E}_\alpha(\boldsymbol{A}) = \boldsymbol{A}_N$ is a diagonal matrix with elements $\mathsf{E}_\alpha(\alpha_i) = c_N/d_{Ni}$.

The variational bound changes to

$$
\mathcal{L}(\mathsf{Q}) \;=\; -\frac{N}{2}\ln 2\pi - \frac{1}{2}\sum_n \left(\frac{a_N}{b_N}(y_n - \boldsymbol{w}_N^\top \boldsymbol{x}_n)^2 + \boldsymbol{x}_n^\top \boldsymbol{V}_N \boldsymbol{x}_n\right) + \frac{1}{2}\ln|\boldsymbol{V}_N| + \frac{D}{2}
$$

$$
-\ln\Gamma(a_0) + a_0\ln b_0 - b_0\frac{a_N}{b_N} + \ln\Gamma(a_N) - a_N\ln b_N + a_N
$$

$$
+\sum_i \left(-\ln\Gamma(c_0) + c_0\ln d_0 + \ln\Gamma(c_N) - c_N\ln d_{Ni}\right). \tag{33}
$$

The predictive distribution remains unchanged, as the prior does not appear in the expression for the variational posterior $\mathsf{P}(\boldsymbol{w},\tau)$.

2.5. **Implementation.** The scripts that compute the variational posterior parameters without and with ARD are vb_linear_fit .m and vb_linear_fit_ard .m, respectively. vb_linear_pred.m computes the predictive density parameters for a set of input vectors.

2.5.1. *Variational posterior parameters without ARD.* The function vb_linear_fit .m computes the variational posterior parameters without ARD, and has syntax

[w, V, invV, logdetV, an, bn, E_a, L] =
    vb_linear_fit (X, y[, a0, b0, c0, d0])

where X is the $N \times D$ input matrix with $\boldsymbol{x}_n^\top$ as its rows, and y is a column vector, containing all y_n's. The optional parameters, a0, b0, c0, and d0, specify the prior parameters a_0, b_0, c_0, and d_0, respectively. If not given, they default to $a_0 = 10^{-2}$, $b_0 = 10^{-4}$, $c_0 = 10^{-2}$, and $d_0 = 10^{-4}$. For these values, the mean of τ^{-1} is undefined, but its mode is at $b_0/(a_0 + 1) \approx 10^{-4}$, implying the a-prior most likely variance of the prior on $\boldsymbol{w}$ to be small. The variance of τ^{-1} is also undefined for $a_0 \leq 2$, but the related variance on τ is $\mathsf{VAR}(\tau) = a_0/b_0^2 = 10^6$. Thus, even though the default prior on τ implies some shrinkage of $\boldsymbol{w} \to \boldsymbol{0}$, this shrinkage is weak due to the prior's large width. Furthermore, the update Eq. (9) of a_N reveals that a_0

can be interpreted as the half the a-prior number of observations. Thus the prior has the weight of 2×10^{-2} observations, and thus loses its influence with very few observations. The same applies for the prior on α.

The returned values, w, V, an, and bn correspond to the normal inverse-gamma parameters $\boldsymbol{w}_N$, $\boldsymbol{V}_N$, a_N, and b_N, respectively, of the variational posterior $\mathsf{Q}^*(\boldsymbol{w}, \tau)$, Eq. (6). The variational parameters for $\mathsf{Q}^*(\alpha)$ are summarized by the returned E_a $= \mathsf{E}_\alpha(\alpha)$. The function additionally returns invV $= \boldsymbol{V}_N^{-1}$, and logdetV $= \ln |\boldsymbol{V}_N|$, such that, if required, these values do not need to be re-computed. The returned L is the variational bound $\mathcal{L}(\mathsf{Q})$, Eq. (22), evaluated at the returned parameters.

After initializing the required data structures, initializing parameters, and pre-computing some constants, the function finds the variational posterior parameters by updating $\mathsf{Q}^*(\boldsymbol{w}, \tau)$ (lines 60–68) and $\mathsf{Q}^*(\alpha)$ (lines 70–72). After each update, it evaluates the parameter-dependent components of the variational bound $\mathcal{L}(\mathsf{Q})$ in lines 74–77. This is repeated until either the change in $\mathcal{L}(\mathsf{Q})$ between two consecutive iterations drops below 0.001%, or the number of iterations exceeds 500.

To update $\boldsymbol{V}_N^{-1}$, $\boldsymbol{w}_N$, b_n, and $\mathcal{L}(\mathsf{Q})$, the function vectorizes operations over n by

$$\sum_n \boldsymbol{x}_n \boldsymbol{x}_n^\top = \text{X'} * \text{X}, \tag{34}$$

$$\sum_n \boldsymbol{x}_n y_n = \text{X'} * \text{y}, \tag{35}$$

$$\boldsymbol{w}_N^\top \boldsymbol{x}_n = (\text{X} * \text{w})_n, \tag{36}$$

$$\sum_n \boldsymbol{x}_n^\top \boldsymbol{V}_N \boldsymbol{x}_n = \mathbf{sum}(\mathbf{sum}(\text{X} .* (\text{X} * \text{V}))). \tag{37}$$

The rest of the code follows closely the update equations derived further above.

2.5.2. *Variational posterior parameters with ARD.* The function vb_linear_fit_ard .m computes the variational parameters with ARD, and has syntax

[w, V, invV, logdetV, an, bn, E_a, L] =
    vb_linear_fit_ard (X, y [, a0, b0, c0, d0])

where the arguments X and y, and optional prior and hyper-prior parameters a0, b0, c0, d0, have the same structure as for vb_linear_fit .m. If not given, the prior / hyper-prior parameters default, as for vb_linear_fit .m, to $a_0 = 10^{-2}$, $b_0 = 10^{-4}$, $c_0 = 10^{-2}$, and $d_0 = 10^{-4}$. The return values w, V, invV, logdetV, an, and bn again correspond to the parameters of the variational posterior $\mathsf{Q}^*(\boldsymbol{w}, \tau)$, and L to the variational bound $\mathcal{L}(\mathsf{Q})$. The only difference to vb_linear_fit .m is that E_a is a vector with D elements, returning the diagonal elements of $\mathsf{E}_\alpha(\boldsymbol{A})$.

The structure of vb_linear_fit .m is similar to vb_linear_fit .m, updating the parameters of $\mathsf{Q}^*(\boldsymbol{w}, \tau)$ and $\mathsf{Q}^*(\boldsymbol{\alpha})$ in lines 61–69 and lines 71–73, respectively, and computing the variational bound, $\mathcal{L}(\mathsf{Q})$ in lines 75–78. This is again repeated until either $\mathcal{L}(\mathsf{Q})$ changes by less than 0.001% between two consecutive iterations, or the number of iterations exceeds 500.

2.5.3. *Predictive density parameters.* For a given set of variational posterior parameters, the function vb_linear_pred.m computes the parameters of the predictive density, Eq. (23), and has syntax

[mu, lambda, nu] = vb_linear_pred(X, w, V, an, bn)

Here X is the $M \times D$ matrix with $\boldsymbol{x}_m^\top$ as its rows. The other arguments correspond to the variational posterior parameters returned by vb_linear_fit .m or vb_linear_fit_ard .m. The function returns the vectors mu and lambda with M elements, and the scalar nu. These variables specify the mean, precision, and the degrees of freedom of the predictive Student's t distribution for y_m (Eq. (23), corresponding to $\boldsymbol{x}_m$) by the mth element of mu and lambda, and by nu, respectively.

2.6. **Examples.**

2.6.1. *Estimation of regression coefficients.* Assuming inputs

```
>> X = [ones(100, 1) randn(100, 3)];
>> y = X * [1 2 3 5]' + randn(100, 1);
```

The mean regression coefficient estimates are found by

```
>> vb_linear_fit(X, y)
ans =

    0.9269
    2.0220
    2.9915
    4.9798
```

Due to the additive noise, these estimates are close to, but do not exactly match the generative coefficients, $\boldsymbol{w} = (1, 2, 3, 5)^\top$.

2.6.2. *Higher-dimensional linear regression, and predictive accuracy.* Let us now consider a 100-dimensional case with only 150 observations, that is $D = 100$ and $N = 150$. We generate the training and testing set by

```
>> D = 100; N = 150; N_test = 50;
>> w = randn(D, 1);
>> X = rand(N, D) − 0.5;
>> X_test = rand(N_test, D) − 0.5;
>> y = X * w + randn(N, 1);
>> y_test = X_test * w + randn(N_test, 1);
```

such that $\boldsymbol{w}$ is drawn from a zero-mean unit variance Gaussian (corresponding to the assumptions of the Bayesian model), and the $\boldsymbol{x}_n$'s have elements drawn from a uniform distribution on $[-0.5, 0.5]$. We train a regression model by both variational Bayesian inference and maximum likelihood by

```
>> [w_VB, V_VB, ~, ~, an_VB, bn_VB] = vb_linear_fit(X, y);
>> y_VB = vb_linear_pred(X, w_VB, V_VB, an_VB, bn_VB);
>> [y_test_VB, lam_VB, nu_VB] = vb_linear_pred(X_test, w_VB, V_VB, an_VB, bn_VB);
>> [w_ML, wint_ML] = regress(y, X);
>> y_ML = X * w_ML;
>> y_test_ML = X_test * w_ML;
```

Measuring the mean squared error for both the training and the test set, we find

```
>> fprintf('Training set MSE: ML = %f, VB = %f\n', ...
        mean((y − y_ML).^2), mean((y − y_VB).^2));
Training set MSE: ML = 0.363473, VB = 0.446073
>> fprintf('Test     set MSE: ML = %f, VB = %f\n', ...
```
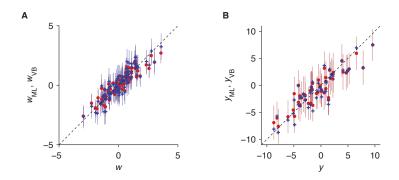
A

B

Figure 1. Coefficient estimates and output predictions for 100-dimensional linear regression example. A) True coefficients vs. coefficient estimates (mean $\pm$ 95% CIs) for variational Bayesian inference (red) and maximum likelihood (blue). The estimates of both inference approaches are horizontally shifted, such that their CIs can be distinguished. B) True outputs vs. test set output predictions (mean, $\pm$ 95% CIs when available) for variational Bayesian inference (red) and maximum likelihood (blue). For both coefficients and output predictions, the variational Bayesian estimates are on average closer to the true values.

$$\textbf{mean}((\text{y\_test} - \text{y\_test\_ML}).\hat{~}2), \textbf{mean}((\text{y\_test} - \text{y\_test\_VB}).\hat{~}2));$$

Test   **set** MSE: ML = 3.622844, VB = 3.221452

Clearly, the maximum likelihood estimate over-fits the training data, as reflected by a small training set error and a large test set error. Variational Bayesian regression also shows signs of over-fitting, but to a lesser extent than maximum likelihood.

We visualize the fit of the regression coefficients by

```
>> figure; hold on;
>> for i = 1:D
       plot(w(i) * [1 1] − 0.01, w_VB(i) + sqrt(V_VB(i,i)) * 1.96 * [−1 1], ...
            '−', 'LineWidth', 0.25, 'Color', [0.8 0.5 0.5]);
       plot(w(i) * [1 1] + 0.01, wint_ML(i,:), ...,
            '−', 'LineWidth', 0.25, 'Color', [0.5 0.5 0.8]);
   end
>> plot(w − 0.01, w_VB, 'o', 'MarkerSize', 3, ...
        'MarkerFaceColor', [0.8 0 0], 'MarkerEdgeColor', 'none');
>> plot(w + 0.01, w_ML, '+', 'MarkerSize', 3, ...
        'MarkerFaceColor', 'none', 'MarkerEdgeColor', [0 0 0.8], 'LineWidth', 1);
>> xymin = min([min(xlim) min(ylim)]); xymax = max([max(xlim) max(ylim)]);
>> plot([xymin xymax], [xymin xymax], 'k−−', 'LineWidth', 0.5);
>> set(gca, 'Box','off', 'PlotBoxAspectRatio', [1 1 1], ...
        'TickDir', 'out', 'TickLength', [1 1]*0.02);
>> xlabel('w'); ylabel('w_{ML}, w_{VB}');
```

which results in Fig. 1A. As can be seen, the better fit of variational Bayesian inference is also reflected in a better estimate of the regression coefficients. Visualizing the test set predictions in the same way results in Fig. 1B. As for the regression
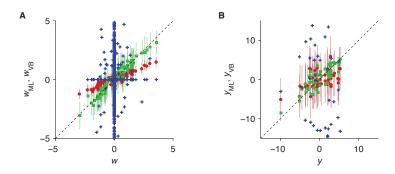
FIGURE 2. Regression coefficients and predictions for variational
Bayesian inference without and with ARD, and maximum like-
lihood estimation, for 1000-dimensional regression problem with
100 informative dimensions. A) True coefficients vs. coefficient
estimates (mean, ± 95% CIs where available), computed by maxi-
mum likelihood (blue, most outside of plotted range), and by vari-
ational Bayesian inference without (red) and with ARD (green).
While maximum likelihood fails to correctly estimate these coeffi-
cients, variational Bayesian inference without ARD applies overly
strong shrinkage to informative dimensions, which causes a bias
towards small coefficient estimates. Only with ARD is it able to
modulate the amount of shrinkage applied to different dimensions
by their informativeness. B) Test set predictions (mean, ± 95%
CIs where available) vs. true values, as estimated by maximum
likelihood (blue, most outside of plotted range), and by variational
Bayesian inference without (red) and with ARD (green).

coefficients, variational Bayesian inference can be seen to provide better predictions
than maximum likelihood.

The code for this example is available in vb_linear_example_highdim.m.

2.6.3. *High-dimensional regression with uninformative input dimensions.* To demon-
strate the effect of Automated Relevance Determination, consider a high-dimensional
input space in which most of the input dimensions are informative (that is, for
which the generative regression coefficients are zero). Specifically, we assume 1000
dimensions, of which only 100 are informative. We generate training and test data
by

```
>> D = 1000; D_eff = 100; N = 500; N_test = 50;
>> w = [randn(D_eff, 1); zeros(D − D_eff, 1)];
>> X = rand(N, D) − 0.5;
>> X_test = rand(N_test, D) − 0.5;
>> y = X * w + randn(N, 1);
>> y_test = X_test * w + randn(N_test, 1);
```

Thus, only the first $D_{\text{eff}} = 100$ elements of the 1000-dimensional $\boldsymbol{w}$ are non-zero.

We find the coefficients by variational Bayesian inference (without and with
ARD) and maximum likelihood by

```
>> [w_VB, V_VB, ˜, ˜, an_VB, bn_VB] = vb_linear_fit(X, y);
```

```
>> y_VB = vb_linear_pred(X, w_VB, V_VB, an_VB, bn_VB);
>> [y_test_VB, lam_VB, nu_VB] = ...
        vb_linear_pred(X_test, w_VB, V_VB, an_VB, bn_VB);
>> [w_VB2, V_VB2, ~, ~, an_VB2, bn_VB2] = vb_linear_fit_ard(X, y);
>> y_VB2 = vb_linear_pred(X, w_VB2, V_VB2, an_VB2, bn_VB2);
>> [y_test_VB2, lam_VB2, nu_VB2] = ...
        vb_linear_pred(X_test, w_VB2, V_VB2, an_VB2, bn_VB2);
>> [w_ML, wint_ML] = regress(y, X);
Warning: X is rank deficient to within machine precision.
> In regress at 84
> In vb_linear_example_sparse at 37
>> y_ML = X * w_ML;
>> y_test_ML = X_test * w_ML;
```

MATLAB's regress function correctly identifies the rank deficiency of X. Due to the shrinkage prior on w, this rank deficiency is not a problem for Bayesian inference. The resulting mean squared prediction errors are found by

```
>> fprintf('Training set MSE: ML = %f, VB = %f, VB w/ ARD = %f\n', ...
           mean((y − y_ML).^2), mean((y − y_VB).^2), mean((y − y_VB2).^2));
Training set MSE: ML = 0.000000, VB = 0.270214, VB w/ ARD = 0.000000
>> fprintf('Test    set MSE: ML = %f, VB = %f, VB w/ ARD = %f\n', ...
           mean((y_test − y_test_ML).^2), mean((y_test − y_test_VB).^2), ...
           mean((y_test − y_test_VB2).^2));
Test    set MSE: ML = 277.108556, VB = 7.164384, VB w/ ARD = 3.230588
```

As in the previous example, the maximum likelihood estimator shows clear signs of over-fitting, as reflected in the large test set error. Variational Bayesian inference does not suffer from over-fitting to the same extent, as is illustrated by the significantly smaller test set error. When used with ARD, this training set error shrinks further, which indicates that ARD is better able to identify and ignore uninformative input dimensions.

A look at the regression coefficients in Fig. 2A (plotted as in the previous example) confirms this property. It illustrates that maximum likelihood was unable to detect the relevant dimensions, whereas variational Bayesian inference without ARD applied overly strong shrinkage to all dimensions. ARD, in contrast, determined the amount of shrinkage for each input dimension separately, and this way was able selectively suppress a subset of these. This is also reflected in the model predictions in Fig. 2B, which inference without ARD underestimates due to overly strong shrinkage of its regression coefficient estimates. With ARD, in contrast, the amount of bias due to shrinkage is reduced.

The code for this example is available in vb_linear_example_sparse.m.

2.6.4. *Model selection by maximizing variational bound.* An appealing property of variational Bayesian inference is that the variational bound $\mathcal{L}(Q)$ lower-bounds the log-model evidence, $\ln P(\mathcal{D})$, and can thus be used for model selection. Here, this feature is demonstrated on the basis of finding the order of the polynomial that has generated the observations. First, let us generate the data by

```
>> D = 3; N = 10; D_ML = 6; Ds = 1:10;
>> x_range = [−5 5];
>> w = randn(D, 1);
```

```
>> x = x_range(1) + (x_range(2) − x_range(1)) * rand(N, 1);
>> x_test = linspace(x_range(1), x_range(2), 100)';
>> gen_X = @(x, d) bsxfun(@power, x, 0:(d−1));
>> X = gen_X(x, D);
>> y = X * w + randn(N, 1);
>> y_test = gen_X(x_test, D) * w;
```

In the above, $D-1$ determines the order of the generative polynomial (which in this case has 2nd order), D_{ML} specifies the order assumed by the maximum likelihood estimate, and Ds is the range of orders tested by model selection. We only generate $N = 10$ training data points to make identifying the correct polynomial order difficult.

Model selection is performed by computing the training data variational bound for a set of orders, to find the order that minimizes this bound,

```
Ls = NaN(1, length(Ds));
>> for i = 1:length(Ds)
        [˜, ˜, ˜, ˜, ˜, ˜, ˜, Ls(i)] = vb_linear_fit (gen_X(x, Ds(i)), y);
    end
>> [˜, i] = max(Ls);
>> D_best = Ds(i)
D_best =

    3
```

As can be seen, model selection correctly identified the order of the generative polynomial. Plotting the variational bound for all tested polynomial orders (Fig. 3A) shows that this selection was unambiguous.

We test model prediction on the previously generated training set for variational Bayesian inference with the inferred polynomial order, and by maximum likelihood with D_{ML}, by

```
>> [w_VB, V_VB, ˜, ˜, an_VB, bn_VB] = vb_linear_fit(gen_X(x, D_best), y);
>> [y_VB, lam_VB, nu_VB] = ...
        vb_linear_pred(gen_X(x_test, D_best), w_VB, V_VB, an_VB, bn_VB);
>> w_ML = regress(y, gen_X(x, D_ML));
>> y_ML = gen_X(x_test, D_ML) * w_ML;
>> fprintf('Test set MSE, ML = %f, VB = %f\n', ...
        mean((y_test − y_ML).^2), mean((y_test − y_VB).^2));
Test set MSE, ML = 16.601449, VB = 0.603299
```

The wrong model ($D_{\mathrm{ML}} = 6$ rather than 3) caused the maximum likelihood estimate to perform badly on the test set, as illustrated by its large mean squared error. Plotting the prediction of both model reveals that the data under-constraints the maximum likelihood, such that its output predictions deviate noticeably from the true outputs for larger inputs (Fig. 3B). For the variational Bayesian model, in contrast, the true, generative polynomial remains within the model prediction's 95% credible intervals.

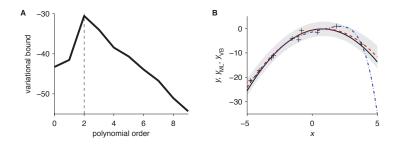The code for this example is available in vb_linear_example_modelsel.m.

FIGURE 3. Variational bound and output prediction for identifying the order of the data-generating polynomial by Bayesian model selection. A) The variational bound, $\mathcal{L}(\mathsf{Q})$, for different generative models, as indexed by the order of the assumed data-generating polynomial. This bound lower-bounds the log-model evidence, $\ln \mathsf{P}(\mathcal{D})$, and shows a clear peak at order 2. The dashed line indicates the order of the true data-generating polynomial. B) Predicted outputs over inputs for the maximum likelihood estimate (blue) and variational Bayesian inference (dashed red). The black curve shows the noise-free data-generating polynomial, and the black crosses are the 10 data points based upon which the regression is performed. The shaded area indicates the 95% CIs for the output prediction of the Bayesian model.

3. LOGISTIC REGRESSION

This section describes how to perform variational Bayesian inference for logistic regression with a hyper-prior on $\boldsymbol{w}$. The basic generative model is the same as that used in Bishop (2006). In addition to this, we here provide a variant that performs automatic relevance determination.

3.1. **The model.** The data y is, dependent on the D-dimensional input $\boldsymbol{x}$, assumed to be of either class $y = -1$ or $y = 1$. The log-likelihood ratio $\ln(\mathsf{P}(y = 1|\boldsymbol{x}, \boldsymbol{w})/\mathsf{P}(y = -1|\boldsymbol{x}, \boldsymbol{w}))$ is assumed to be linear in $\boldsymbol{x}$, such that the conditional likelihood for $y = 1$ is given by the sigmoid

$$\mathsf{P}(y = 1|\boldsymbol{x}, \boldsymbol{w}) = \frac{1}{1 + \exp(-\boldsymbol{w}^\top \boldsymbol{x})} = \sigma(\boldsymbol{w}^\top \boldsymbol{x}). \tag{38}$$

Equally, $\mathsf{P}(y = -1|\boldsymbol{x}, \boldsymbol{w}) = 1 - \mathsf{P}(y = 1|\boldsymbol{x}, \boldsymbol{w}) = 1/(1 + \exp(\boldsymbol{w}^\top \boldsymbol{x}))$, such that

$$\mathsf{P}(y|\boldsymbol{x}, \boldsymbol{w}) = \sigma(y\boldsymbol{w}^\top \boldsymbol{x}). \tag{39}$$

Given some data $\mathcal{D} = \{\boldsymbol{X}, \boldsymbol{Y}\}$, where $\boldsymbol{X} = \{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_N\}$ and $\boldsymbol{Y} = \{y_1, \ldots, y_N\}$ are N input/output pairs, the aim is to find the posterior $\mathsf{P}(\boldsymbol{w}|\mathcal{D})$, given some prior $\mathsf{P}(\boldsymbol{w})$. Unfortunately, the sigmoid data likelihood does not admit a conjugate-exponential prior. Therefore, approximations need to be applied to find an analytic expression for the posterior.

The approximation that will be used is quadratic in $\boldsymbol{w}$ in the exponential, such that the conjugate Gaussian prior

$$\mathsf{P}(\boldsymbol{w}|\alpha) = \mathcal{N}(\boldsymbol{w}|\boldsymbol{0}, \alpha^{-1}\boldsymbol{I}) = \left(\frac{\alpha}{2\pi}\right)^{D/2} \exp\left(-\frac{\alpha}{2}\boldsymbol{w}^\top \boldsymbol{w}\right) \tag{40}$$

can be used. This prior is parametrized by the hyper-parameter $\boldsymbol{\alpha}$ that is modeled by a conjugate Gamma distribution

$$P(\alpha) = \text{Gam}(\alpha|a_0, b_0) = \frac{1}{\Gamma(a_0)} b_0^{a_0} \alpha^{a_0-1} \exp(-b_0\alpha). \tag{41}$$

This hyper-prior implies the a-prior variance, α^{-1} of the zero-mean $\boldsymbol{w}$ to be inverse-gamma with moments $\mathsf{E}\left(\alpha^{-1}\right) = b_0/(a_0 - 1)$ for $a_0 > 1$, and $\mathsf{VAR}\left(\alpha^{-1}\right) = b_0^2/\left((a_0 - 1)^2(a_0 - 2)\right)$ for $a_0 > 2$.

3.2. **Variational Bayesian inference.** Variational Bayesian inference is based on maximizing a lower bound on the marginal data log-likelihood

$$\ln P(\boldsymbol{Y}|\boldsymbol{X}) = \ln \iint P(\boldsymbol{Y}|\boldsymbol{X}, \boldsymbol{w})P(\boldsymbol{w}|\alpha)P(\alpha)\mathrm{d}\boldsymbol{w}\mathrm{d}\alpha. \tag{42}$$

This lower bound is given by

$$\ln P(\boldsymbol{Y}|\boldsymbol{X}) \geq \mathcal{L}(\mathsf{Q}) = \iint \mathsf{Q}(\boldsymbol{w}, \alpha) \ln \frac{P(\boldsymbol{Y}|\boldsymbol{X}, \boldsymbol{w})p(\boldsymbol{w}|\alpha)P(\alpha)}{\mathsf{Q}(\boldsymbol{w}, \alpha)} \mathrm{d}\boldsymbol{w}\mathrm{d}\alpha, \tag{43}$$

where the variational distribution $\mathsf{Q}(\boldsymbol{w}, \alpha)$, approximating the posterior $P(\boldsymbol{w}, \alpha|\mathcal{D})$, is assumed to factor into $\mathsf{Q}(\boldsymbol{w}, \alpha) = \mathsf{Q}(\boldsymbol{w})\mathsf{Q}(\alpha)$. This approximation leads to analytic posterior expressions if the model structure is conjugate-exponential.

The data likelihood

$$P(\boldsymbol{Y}|\boldsymbol{X}, \boldsymbol{w}) = \prod_n P(y_n|\boldsymbol{x}_n, \boldsymbol{w}) = \prod_n \sigma(y_n \boldsymbol{w}^\top \boldsymbol{x}_n) \tag{44}$$

does not admit a conjugate prior in the exponential family and will be approximated by the use of

$$\sigma(z) \geq \sigma(\xi) \exp\left((z - \xi)/2 - \lambda(\xi)(z^2 - \xi^2)\right), \quad \lambda(\xi) = \frac{1}{2\xi}\left(\sigma(\xi) - \frac{1}{2}\right), \tag{45}$$

which is a tight lower bound on the sigmoid, with one additional parameter ξ per datum (Jaakkola and Jordan, 2000). Applying this bound, the data log-likelihood is lower-bounded by

$$\begin{aligned}
\ln P(\boldsymbol{Y}|\boldsymbol{X}, \boldsymbol{w}) &\geq& \ln h(\boldsymbol{w}, \boldsymbol{\xi}) \\
&=& \boldsymbol{w}^\top \sum_n \frac{y_n}{2}\boldsymbol{x}_n - \boldsymbol{w}^\top \left(\sum_n \lambda(\xi_n)\boldsymbol{x}_n\boldsymbol{x}_n^\top\right)\boldsymbol{w} \\
&& + \sum_n \left(\ln \sigma(\xi_n) - \frac{\xi_n}{2} + \lambda(\xi_n)\xi_n^2\right),
\end{aligned} \tag{46}$$

with one local variation parameter ξ_n per datum. This results in the new variational bound

$$\tilde{\mathcal{L}}(\mathsf{Q}, \boldsymbol{\xi}) = \iint \mathsf{Q}(\boldsymbol{w}, \alpha) \ln \frac{h(\boldsymbol{w}, \boldsymbol{\xi})P(\boldsymbol{w}|\alpha)P(\alpha)}{\mathsf{Q}(\boldsymbol{w}, \alpha)} \mathrm{d}\boldsymbol{w}\mathrm{d}\alpha, \tag{47}$$

which is a lower bound on the original variational bound, that is $\tilde{\mathcal{L}}(\mathsf{Q}, \boldsymbol{\xi}) \leq \mathcal{L}(\mathsf{Q})$.

The variational posteriors are evaluated by standard variational methods for factorized distributions. The variational posterior for $\boldsymbol{w}$ is given by

$$
\begin{aligned}
\ln \mathsf{Q}^*(\boldsymbol{w}) &= \ln h(\boldsymbol{w}, \boldsymbol{\xi}) + \mathsf{E}_\alpha(\ln \mathsf{P}(\boldsymbol{w}|\alpha)) + \text{const.} \\
&= \boldsymbol{w}^\top \sum_n \frac{y_n}{2} \boldsymbol{x}_n - \frac{1}{2} \boldsymbol{w}^\top \left(\mathsf{E}_\alpha(\alpha)\boldsymbol{I} + 2\sum_n \lambda(\xi_n)\boldsymbol{x}_n\boldsymbol{x}_n^\top \right) \boldsymbol{w} + \text{const.} \\
&= \ln \mathcal{N}(\boldsymbol{w}|\boldsymbol{w}_N, \boldsymbol{V}_N),
\end{aligned}
\tag{48}
$$

with parameters

$$
\boldsymbol{V}_N^{-1} = \mathsf{E}_\alpha(\alpha)\boldsymbol{I} + 2\sum_n \lambda(\xi_n)\boldsymbol{x}_n\boldsymbol{x}_n^\top,
\tag{49}
$$

$$
\boldsymbol{w}_N = \boldsymbol{V}_N \sum_n \frac{y_n}{2}\boldsymbol{x}_n.
\tag{50}
$$

The variational posterior for α results in

$$
\begin{aligned}
\ln \mathsf{Q}^*(\alpha) &= \mathsf{E}_{\boldsymbol{w}}(\ln \mathsf{P}(\boldsymbol{w}|\alpha)) + \ln \mathsf{P}(\alpha) + \text{const.} \\
&= \left(a_0 - 1 + \frac{D}{2} \right) \ln \alpha - \left(b_0 + \frac{1}{2}\mathsf{E}_{\boldsymbol{w}}(\boldsymbol{w}^\top \boldsymbol{w}) \right) \alpha + \text{const.} \\
&= \ln \text{Gam}(\alpha|a_N, b_N),
\end{aligned}
\tag{51}
$$

with

$$
a_N = a_0 + \frac{D}{2},
\tag{52}
$$

$$
b_N = b_0 + \frac{1}{2}\mathsf{E}_{\boldsymbol{w}}(\boldsymbol{w}^\top \boldsymbol{w}).
\tag{53}
$$

The expectations are evaluated with respect to the variational posteriors and result in

$$
\mathsf{E}_\alpha(\alpha) = \frac{a_N}{b_N},
\tag{54}
$$

$$
\mathsf{E}_{\boldsymbol{w}}(\boldsymbol{w}^\top \boldsymbol{w}) = \boldsymbol{w}_N^\top \boldsymbol{w}_N + \text{Tr}(\boldsymbol{V}_N).
\tag{55}
$$

The variational bound itself is given by

$$
\begin{aligned}
\tilde{\mathcal{L}}(\mathsf{Q}, \boldsymbol{\xi}) &= \mathsf{E}_{\boldsymbol{w}}(\ln h(\boldsymbol{w}, \boldsymbol{\xi})) + \mathsf{E}_{\boldsymbol{w},\alpha}(\ln \mathsf{P}(\boldsymbol{w}|\alpha)) + \mathsf{E}_\alpha(\ln \mathsf{P}(\alpha)) \\
&\quad - \mathsf{E}_{\boldsymbol{w}}(\ln \mathsf{Q}(\boldsymbol{w})) - \mathsf{E}_\alpha(\ln \mathsf{Q}(\alpha)),
\end{aligned}
\tag{56}
$$

$$
\begin{aligned}
\mathsf{E}_{\boldsymbol{w}}(\ln h(\boldsymbol{w}, \boldsymbol{\xi})) &= \frac{1}{2}\boldsymbol{w}_N^\top \boldsymbol{V}_N^{-1} \boldsymbol{w}_N - \frac{D}{2} + \frac{1}{2}\frac{a_N}{b_N}\left(\boldsymbol{w}_N^\top \boldsymbol{w}_N + \text{Tr}(\boldsymbol{V}_N) \right) \\
&\quad + \sum_n \left(\ln \sigma(\xi_n) - \frac{\xi_n}{2} + \lambda(\xi_n)\xi_n^2 \right),
\end{aligned}
\tag{57}
$$

$$
\begin{aligned}
\mathsf{E}_{\boldsymbol{w},\alpha}(\ln \mathsf{P}(\boldsymbol{w}|\alpha)) &= -\frac{D}{2}\ln 2\pi + \frac{D}{2}(\psi(a_N) - \ln b_N) \\
&\quad - \frac{1}{2}\frac{a_N}{b_N}\left(\boldsymbol{w}_N^\top \boldsymbol{w}_N + \text{Tr}(\boldsymbol{V}_N) \right),
\end{aligned}
\tag{58}
$$

$$
\mathsf{E}_\alpha(\ln \mathsf{P}(\alpha)) = -\ln \Gamma(a_0) + a_0 \ln b_0 + (a_0 - 1)(\psi(a_N) - \ln b_N) - b_0 \frac{a_N}{b_N},
\tag{59}
$$

$$
\mathsf{E}_{\boldsymbol{w}}(\ln \mathsf{Q}(\boldsymbol{w})) = -\frac{1}{2}\ln |\boldsymbol{V}_N| - \frac{D}{2}(1 + \ln 2\pi),
\tag{60}
$$

$$
\mathsf{E}_\alpha(\ln \mathsf{Q}(\alpha)) = -\ln \Gamma(a_N) + (a_N - 1)\psi(a_N) + \ln b_N - a_N,
\tag{61}
$$

where $\psi(\cdot)$ is the digamma function. In combination, this gives

$$\tilde{\mathcal{L}}(\mathsf{Q}, \boldsymbol{\xi}) \;=\; \frac{1}{2}\boldsymbol{w}_N^\top \boldsymbol{V}_N^{-1}\boldsymbol{w}_N + \frac{1}{2}\ln|\boldsymbol{V}_N| + \sum_n \left(\ln\sigma(\xi_n) - \frac{\xi_n}{2} + \lambda(\xi_n)\xi_n^2 \right)$$
$$- \ln\Gamma(a_0) + a_0\ln b_0 - b_0\frac{a_N}{b_N} - a_N\ln b_N + \ln\Gamma(a_N) + a_N. \quad (62)$$

This bound is to be maximized in order to find the variational posteriors for $\boldsymbol{w}$ and α. The expressions that maximize this bound with respect to $\mathsf{Q}(\boldsymbol{w})$ and $\mathsf{Q}(\boldsymbol{\alpha})$, while keeping all other parameters fixed, are given by $\mathsf{Q}^*(\boldsymbol{w})$ and $\mathsf{Q}^*(\alpha)$ respectively. To find the local variational parameters ξ_n that maximize $\tilde{\mathcal{L}}(\mathsf{Q}, \boldsymbol{\xi})$, its derivative with respect to ξ_n is set to zero (see Bishop (2006)), resulting in

$$(\xi_n^{\text{new}})^2 = \boldsymbol{x}_n^\top \left(\boldsymbol{V}_N + \boldsymbol{w}_N\boldsymbol{w}_N^\top \right) \boldsymbol{x}_n. \quad (63)$$

The variational bound is maximized by iterating over the update equations for $\boldsymbol{w}_N$, $\boldsymbol{V}_N$, a_N, b_N and $\boldsymbol{\xi}$, until $\tilde{\mathcal{L}}(\mathsf{Q}, \boldsymbol{\xi})$ reaches a plateau. A lower bound on the marginal data log-likelihood $\ln\mathsf{P}(\mathcal{D})$ is given by the variational bound itself, as $\ln\mathsf{P}(\mathcal{D}) \geq \mathcal{L}(\mathsf{Q}) \geq \tilde{\mathcal{L}}(\mathsf{Q}, \boldsymbol{\xi})$.

3.3. **Predictive density.** In order to get the predictive density, the posterior $\mathsf{P}(\boldsymbol{w}|\mathcal{D})$ is approximated by the variational posterior $\mathsf{Q}(\boldsymbol{w})$, and the sigmoid is lower-bounded by above bound, such that

$$\mathsf{P}(y = 1|\boldsymbol{x}, \mathcal{D}) \;=\; \int \mathsf{P}(y = 1|\boldsymbol{x}, \boldsymbol{w})\mathsf{P}(\boldsymbol{w}|\mathcal{D})\mathrm{d}\boldsymbol{w}$$
$$\approx \int \mathsf{P}(y = 1|\boldsymbol{x}, \boldsymbol{w})\mathsf{Q}(\boldsymbol{w})\mathrm{d}\boldsymbol{w}, \quad (64)$$
$$\geq \int \sigma(\xi)\exp\left(\frac{\boldsymbol{w}^\top\boldsymbol{x} - \xi}{2} - \lambda(\xi)\boldsymbol{w}^\top\boldsymbol{x}\boldsymbol{x}^\top\boldsymbol{w} + \lambda(\xi)\xi^2 \right) \mathsf{Q}(\boldsymbol{w})\mathrm{d}\boldsymbol{w}.$$

The integral is solved by noting that the lower bound is exponentially quadratic in $\boldsymbol{w}$, such that the Gaussian can be completed, to give

$$\ln\mathsf{P}(y = 1|\boldsymbol{x}, \mathcal{D}) \approx \frac{1}{2}\ln\frac{|\tilde{\boldsymbol{V}}|}{|\boldsymbol{V}_N|} - \frac{1}{2}\boldsymbol{w}_N^\top\boldsymbol{V}_N^{-1}\boldsymbol{w}_N + \frac{1}{2}\tilde{\boldsymbol{w}}^\top\tilde{\boldsymbol{V}}^{-1}\tilde{\boldsymbol{w}} + \ln\sigma(\xi) - \frac{\xi}{2} + \lambda(\xi)\xi^2,$$
$$(65)$$

with

$$\tilde{\boldsymbol{V}}^{-1} \;=\; \boldsymbol{V}_N^{-1} + 2\lambda(\xi)\boldsymbol{x}\boldsymbol{x}^\top, \quad (66)$$
$$\tilde{\boldsymbol{w}} \;=\; \tilde{\boldsymbol{V}}\left(\boldsymbol{V}_N^{-1}\boldsymbol{w}_N + \frac{\boldsymbol{x}}{2} \right). \quad (67)$$

The bound parameter ξ that maximizes $\ln\mathsf{P}(y = 1|\boldsymbol{x}, \mathcal{D})$ is given by

$$(\xi^{\text{new}})^2 = \boldsymbol{x}^\top\left(\tilde{\boldsymbol{V}} + \tilde{\boldsymbol{w}}\tilde{\boldsymbol{w}}^\top \right)\boldsymbol{x}. \quad (68)$$

Thus, the predictive density is found by iterating over the updates for $\tilde{\boldsymbol{w}}$, $\tilde{\boldsymbol{V}}$ and ξ until $\ln\mathsf{P}(y = 1|\boldsymbol{x}, \mathcal{D})$ reaches a plateau. The hyper-prior $\mathsf{P}(\alpha)$ does not need to be considered as it does not appear in the variational posterior $\mathsf{Q}(\boldsymbol{w})$.

3.4. **Using automatic relevance determination.** To use automatic relevance determination (ARD), each element of the prior of $\boldsymbol{w}$ is assigned a separate prior,

$$\mathsf{P}(\boldsymbol{w}|\boldsymbol{\alpha}) = \mathcal{N}(\boldsymbol{w}|\boldsymbol{0}, \boldsymbol{A}^{-1}) = \frac{|\boldsymbol{A}|^{1/2}}{\sqrt{2\pi}^D} \exp\left(-\frac{1}{2}\boldsymbol{w}^\top \boldsymbol{A}\boldsymbol{w}\right), \tag{69}$$

where $\boldsymbol{A}$ is the diagonal matrix with the vector $\boldsymbol{\alpha} = (\alpha_1, \ldots, \alpha_D)^\top$ along its diagonal. The conjugate hyper-prior $\mathsf{P}(\boldsymbol{\alpha})$ is given by

$$\mathsf{P}(\boldsymbol{\alpha}) = \prod_i \mathrm{Gam}(\alpha_i|a_0, b_0). \tag{70}$$

Note that α_i determines the precision (inverse variance) of the ith element of $\boldsymbol{w}$. A low precision makes the prior uninformative, whereas a high precision tells us that the associated element in $\boldsymbol{w}$ is most likely zero and the associated input element is therefore irrelevant for the prediction of y. Thus, such a prior structure automatically determines the relevance of each element of the input to predict the class of the output.

Using the same variational Bayes inference as before, the variational posteriors are given by

$$\mathsf{Q}^*(\boldsymbol{w}) = \mathcal{N}(\boldsymbol{w}|\boldsymbol{w}_N, \boldsymbol{V}_N), \quad \mathsf{Q}^*(\boldsymbol{\alpha}) = \prod_i \mathrm{Gam}(\alpha_i|a_N, b_{Ni}), \tag{71}$$

with

$$\boldsymbol{V}_N^{-1} = \mathsf{E}_\alpha(\boldsymbol{A}) + 2\sum_n \lambda(\xi_n)\boldsymbol{x}_n\boldsymbol{x}_n^\top, \tag{72}$$

$$\boldsymbol{w}_N = \boldsymbol{V}_N \sum_n \frac{y_n}{2}\boldsymbol{x}_n, \tag{73}$$

$$a_N = a_0 + \frac{1}{2}, \tag{74}$$

$$b_{Ni} = b_0 + \frac{1}{2}\mathsf{E}_{\boldsymbol{w}}(w_i^2), \tag{75}$$

where w_i is the ith element of $\boldsymbol{w}$, and $\boldsymbol{A}_N = \mathsf{E}_\alpha(\boldsymbol{A})$ is a diagonal matrix with its ith diagonal element given by $\mathsf{E}_\alpha(\alpha_i) = a_N/b_{Ni}$. $\mathsf{E}_{\boldsymbol{w}}(w_i^2)$ evaluates to $\mathsf{E}_{\boldsymbol{w}}(w_i^2) = \mathsf{E}_{\boldsymbol{w}}(w_i)^2 + \mathsf{VAR}_{\boldsymbol{w}}(w_i) = (\boldsymbol{w}_N)_i^2 + (\boldsymbol{V}_N)_{ii}$.

The new expectations to evaluate the variation bound are

$$
\begin{aligned}
\mathsf{E}_{\boldsymbol{w}}(\ln h(\boldsymbol{w},\boldsymbol{\xi})) \;=\;& \frac{1}{2}\boldsymbol{w}_N^\top \boldsymbol{V}_N^{-1}\boldsymbol{w}_N - \frac{D}{2} + \frac{1}{2}\left(\mathrm{Tr}(\boldsymbol{A}_N\boldsymbol{V}_N) + \boldsymbol{w}_N^\top \boldsymbol{A}_N\boldsymbol{w}_N\right) \\
& + \sum_n \left(\ln\sigma(\xi_n) - \frac{\xi_n}{2} + \lambda(\xi_n)\xi_n^2\right),
\end{aligned}
\tag{76}
$$

$$
\begin{aligned}
\mathsf{E}_{\boldsymbol{w},\boldsymbol{\alpha}}(\ln \mathsf{P}(\boldsymbol{w}|\boldsymbol{\alpha})) \;=\;& \frac{1}{2}\sum_i \left(\psi(a_N) - \ln b_{Ni}\right) \\
& - \frac{D}{2}\ln 2\pi - \frac{1}{2}\left(\mathrm{Tr}(\boldsymbol{A}_N\boldsymbol{V}_N) + \boldsymbol{w}_N^\top \boldsymbol{A}_N\boldsymbol{w}_N\right),
\end{aligned}
\tag{77}
$$

$$
\begin{aligned}
\mathsf{E}_{\boldsymbol{\alpha}}(\ln \mathsf{P}(\boldsymbol{\alpha})) \;=\;& \sum_i \Big(-\ln\Gamma(a_0) + a_0\ln b_0 \\
& + (a_0-1)(\psi(a_N) - \ln b_{Ni}) - b_0\frac{a_N}{b_{Ni}}\Big),
\end{aligned}
\tag{78}
$$

$$
\mathsf{E}_{\boldsymbol{\alpha}}(\ln \mathsf{Q}(\boldsymbol{\alpha})) \;=\; \sum_i \left(-\ln\Gamma(a_N) + (a_N-1)\psi(a_N) + \ln b_{Ni} - a_N\right),
\tag{79}
$$

resulting in

$$
\begin{aligned}
\tilde{\mathcal{L}}(\mathsf{Q},\boldsymbol{\xi}) \;=\;& \frac{1}{2}\boldsymbol{w}_N^\top \boldsymbol{V}_N^{-1}\boldsymbol{w}_N + \frac{1}{2}\ln|\boldsymbol{V}_N| + \sum_n\left(\ln\sigma(\xi_n) - \frac{\xi_n}{2} + \lambda(\xi_n)\xi_n^2\right) \\
& + \sum_i \Big(-\ln\Gamma(a_0) + a_0\ln b_0 - b_0\frac{a_N}{b_{Ni}} - a_N\ln b_{Ni} + \ln\Gamma(a_N) + a_N\Big).
\end{aligned}
\tag{80}
$$

As the variational posterior $\mathsf{Q}^*(\boldsymbol{w})$ is independent of the hyper-parameters, the predictive density is evaluated as before.

3.5. **Implementation.** The scripts that compute the variational posterior parameters without and with ARD are vb_logit_fit .m and vb_logit_fit_ard .m, respectively. vb_logit_fit_iter .m is a slower version of vb_logit_fit .m that features a slightly simplified generative model without the hyper-prior, and iterates over updating each ξ_n separately rather updating them for all $\boldsymbol{x}_n$'s simultaneously. vb_logit_pred .m computes the predictive density parameters for a set of input vectors. vb_logit_pred_iter .m does so as well, but again iterates over updating ξ_n rather than updating them all at the same time.

3.5.1. *Variational posterior parameters without ARD, iterative implementation.* The function vb_logit_fit_iter .m deviates from the generative model given by Eqs. (38)-(41) as it does not use a hyper-prior on α. Instead, it uses the conjugate Gaussian prior

$$
\mathsf{P}(\boldsymbol{w}) = \mathcal{N}\left(\boldsymbol{w}|\boldsymbol{0}, D^{-1}\boldsymbol{I}\right),
\tag{81}
$$

where D is the input dimensionality. This prior was chosen to increase shrinkage with the number of input dimensions. The function is called by

[w, V, invV, logdetV] = vb_logit_fit_iter (X, y)

where X is the $N \times D$ input matrix with $\boldsymbol{x}_n^\top$ as its nth row, and y is the output column vector with N elements that are either -1 or 1. The returned w, V specify the parameters $\boldsymbol{w}_N$, $\boldsymbol{V}_N$ of the variational posterior Eq. (48). The function additionally returns invV $= \boldsymbol{V}_N^{-1}$ and logdetV $= \ln|\boldsymbol{V}_n|$, such that, if required, these values do not need to be re-computed.

The function computes these parameters incrementally by adding the observations $\boldsymbol{x}_n, y_n$ one by one, while optimizing ξ_n for each of these observations separately. Let $\boldsymbol{V}_j$ and $\boldsymbol{w}_j$ denote the parameters of $\mathsf{Q}^*(\boldsymbol{w})$ after j observations have been made. Starting with $\boldsymbol{w}_0 = \boldsymbol{0}$, $\boldsymbol{V}_0 = D^{-1}\boldsymbol{I}$, $\boldsymbol{V}_0^{-1} = D\boldsymbol{I}$, and $\ln|\boldsymbol{V}_0^{-1}| = -D\ln D$ according to the prior, $\boldsymbol{V}_j$ follows the incremental update

$$\boldsymbol{V}_j^{-1} = \boldsymbol{V}_{j-1}^{-1} + 2\lambda(\xi_j)\boldsymbol{x}_j\boldsymbol{x}_j^\top. \tag{82}$$

The incremental update of $\boldsymbol{w}_j$ is slightly more complex, but from observing that

$$\boldsymbol{V}_j^{-1}\boldsymbol{w}_j = \sum_n^j \frac{y_n}{2}\boldsymbol{x}_n = \frac{y_j}{2}\boldsymbol{x}_j + \sum_n^{j-1} \frac{y_n}{2}\boldsymbol{x}_n = \boldsymbol{V}_{j-1}^{-1}\boldsymbol{w}_{j-1} + \frac{y_j}{2}\boldsymbol{x}_j, \tag{83}$$

it is easy to see that

$$\boldsymbol{w}_j = \boldsymbol{V}_j \left(\boldsymbol{V}_{j-1}^{-1}\boldsymbol{w}_{j-1} + \frac{y_j}{2}\boldsymbol{x}_j \right). \tag{84}$$

The script avoids taking the inverse of $\boldsymbol{V}$ by updating $\boldsymbol{V}^{-1}$ and $\boldsymbol{V}$ in parallel, where the latter is based on an application of the Sherman-Morrison formula on the $\boldsymbol{V}^{-1}$ update, resulting in

$$\boldsymbol{V}_j = \left(\boldsymbol{V}_{j-1}^{-1} + 2\lambda(\xi_j)\boldsymbol{x}_j\boldsymbol{x}_j^\top \right)^{-1} = \boldsymbol{V}_{j-1} - \frac{2\lambda(\xi_j)\boldsymbol{V}_{j-1}\boldsymbol{x}_j\boldsymbol{x}_j^\top\boldsymbol{V}_{j-1}}{1 + 2\lambda(\xi_j)\boldsymbol{x}_j^\top\boldsymbol{V}_{j-1}\boldsymbol{x}_j}. \tag{85}$$

$\ln|\boldsymbol{V}_j|$ can be updated in a similar way, based on the Matrix determinant lemma,

$$|\boldsymbol{V}_j^{-1}| = |\boldsymbol{V}_{j-1}^{-1} + 2\lambda(\xi_j)\boldsymbol{x}_j\boldsymbol{x}_j^\top| = |\boldsymbol{V}_{j-1}^{-1}| \left(1 + 2\lambda(\xi_j)\boldsymbol{x}_j^\top\boldsymbol{V}_{j-1}\boldsymbol{x}_j \right), \tag{86}$$

such that, using $\ln|\boldsymbol{V}_j| = -\ln|\boldsymbol{V}_j^{-1}|$,

$$\ln|\boldsymbol{V}_j| = \ln|\boldsymbol{V}_{j-1}| - \ln\left(1 + 2\lambda(\xi_j)\boldsymbol{x}_j^\top\boldsymbol{V}_{j-1}\boldsymbol{x}_j \right). \tag{87}$$

The function initializes $\boldsymbol{V}_0$ and $\boldsymbol{w}_0$ in lines 37–40, and then updates its values by iterating over the $\boldsymbol{x}_j$'s for $j = 1, \ldots, N$, starting in line 44. For each j, it first updates $\boldsymbol{V}$ and $\boldsymbol{w}$ under the assumption that $\xi_n = 0$ and $\lambda(\xi_n) = 1/8$, leading to a simplified initial step,

$$\boldsymbol{V}_j^{-1}(\xi_j) \quad =_{\xi_j=0} \quad \boldsymbol{V}_{j-1}^{-1} + \frac{1}{4}\boldsymbol{x}_j\boldsymbol{x}_j^\top, \tag{88}$$

$$\boldsymbol{V}_j(\xi_j) \quad =_{\xi_j=0} \quad \boldsymbol{V}_{j-1} - \frac{\boldsymbol{V}_{j-1}\boldsymbol{x}_j\boldsymbol{x}_j^\top\boldsymbol{V}_{j-1}}{4 + \boldsymbol{x}_j^\top\boldsymbol{V}_{j-1}\boldsymbol{x}_j}, \tag{89}$$

$$\ln|\boldsymbol{V}_j(\xi_j)| \quad =_{\xi_j=0} \quad \ln|\boldsymbol{V}_{j-1}| - \ln\left(1 + \frac{1}{4}\boldsymbol{x}_j^\top\boldsymbol{V}_{j-1}\boldsymbol{x}_j \right). \tag{90}$$

Then, in lines 62–86, it alternates between updating ξ_j, and $\boldsymbol{V}_j(\xi_j)$ and $\boldsymbol{w}_j(\xi_j)$ while monitoring how these updates change the variational bound $\mathcal{L}(\xi_j)$. The updates are performed until the variational bound changes less than 0.001% between two consecutive updates of all parameters, or the number of iterations exceeds 500. The variational bound itself is, without the hyper-prior, given by

$$\mathcal{L}_j(\xi_j) = \frac{1}{2}\boldsymbol{w}_j^\top(\xi_j)\boldsymbol{V}_j^{-1}(\xi_j)\boldsymbol{w}_j(\xi_j) + \frac{1}{2}\ln|\boldsymbol{V}_j(\xi_j)| + \ln\sigma(\xi_j) - \frac{\xi_j}{2} + \lambda(\xi_j)\xi_j^2. \tag{91}$$

3.5.2. *Variational posterior parameters without ARD, batch implementation.* Rather than updating all ξ_n in turn, vb_logit_fit .m estimates the variational posterior parameters by updating all ξ_n's at once. Furthermore, it differs from vb_logit_fit_iter .m in that it assumes the full generative mode, Eqs. (38)-(41), including the hyper-prior on α with associated parameters a and b. The function is called by

[w, V, invV, logdetV, E_a, L] = vb_logit_fit (X, y [, a0, b0])

where X and y specify inputs and outputs as for vb_logit_fit_iter .m. The optional a0 and b0 specify the hyper-prior parameters a_0 and b_0. If not given, they default to $a_0 = 10^{-2}$ and $b_0 = 10^{-4}$, corresponding to an un-informative hyper-prior (see Sec. 2.5). The returned w and V correspond to the posterior parameters $\boldsymbol{w}_N$ and $\boldsymbol{V}_N$ of the variational posterior of $\boldsymbol{w}$. E_a is $\mathsf{E}_\alpha(\alpha)$ of the posterior α, and L is the variational bound $\tilde{\mathcal{L}}(\mathsf{Q}, \boldsymbol{\xi})$ at these parameters and the last-used $\boldsymbol{\xi}$. The function additionally returns invV $= \boldsymbol{V}_N^{-1}$ and logdetV $= \ln|\boldsymbol{V}_N|$, such that, if required, these values do not need to be re-computed.

Within the function, all ξ_n are stored in the vector xi and are updated simultaneously. The script start in line 47 by assuming $\xi_n = 0$ for all n, such that $\lambda(\xi_n) = 1/8$. Additionally, it pre-computes w_t $= \sum_n \boldsymbol{x}_n y_n / 2$. The initial update of $\boldsymbol{V}_N(\boldsymbol{\xi})$, $\boldsymbol{w}_N(\boldsymbol{\xi})$, $b_N(\boldsymbol{\xi})$, and $\tilde{\mathcal{L}}(\mathsf{Q}, \boldsymbol{\xi})$ in lines 49–55 is computed outside of the loop. After that, the script iterates in lines 59–88 over first updating $\boldsymbol{\xi}$, then $b_N(\boldsymbol{\xi})$, followed by $\boldsymbol{V}_N(\boldsymbol{\xi})$ and $\boldsymbol{w}_N(\boldsymbol{\xi})$. The iteration stops if either $\tilde{\mathcal{L}}(\mathsf{Q}, \boldsymbol{\xi})$ does not change more than 0.001% between two consecutive iterations, or the number of iterations exceeds 500.

The script employs a few short-cuts and vectorizations, which will be discussed here. In particular, the initial $\tilde{\mathcal{L}}(\mathsf{Q}, \boldsymbol{\xi})$ at $\boldsymbol{\xi} = \boldsymbol{0}$ is simplified by using

$$\sum_n \left(\ln \sigma(\xi_n) - \frac{\xi_n}{2} + \lambda(\xi_n) \xi_n^2 \right) =_{\boldsymbol{\xi}=\boldsymbol{0}} -N \ln 2. \tag{92}$$

Also, as the scripts computes $\boldsymbol{V}_N(\boldsymbol{\xi})$ by inverting $\boldsymbol{V}_N^{-1}(\boldsymbol{\xi})$, it computes $\ln|\boldsymbol{V}_N(\boldsymbol{\xi})|$ from $\boldsymbol{V}_N^{-1}(\boldsymbol{\xi})$ for better stability, using $\ln|\boldsymbol{V}_N(\boldsymbol{\xi})| = -\ln|\boldsymbol{V}_N^{-1}(\boldsymbol{\xi})|$. In addition, the following vectorized operations are used:

$$\sum_n \frac{y_n}{2} \boldsymbol{x}_n = 0.5 * \mathbf{sum}(\text{bsxfun}(@\text{times, X, y}), 1)', \tag{93}$$

$$2 \sum_n \lambda(\xi_n) \boldsymbol{x}_n \boldsymbol{x}_n^\top = 2 * X' * \text{bxsfun}(@\text{times, X, lam\_xi}), \tag{94}$$

$$\boldsymbol{x}_n^\top \left(\boldsymbol{V}_N + \boldsymbol{w}_N \boldsymbol{w}_N^\top \right) \boldsymbol{x}_n = \left(\mathbf{sum}(X .* (X * (V + w * w')), 2) \right)_n. \tag{95}$$

Using a hyper-prior comes at the cost of having to explicitly invert $\boldsymbol{V}^{-1}$ at each iteration. For this reason, vb_logit_fit .m might be numerically less stable than vb_logit_fit_iter .m for problems will ill-conditioned inputs.

3.5.3. *Variational posterior parameters with ARD.* Estimating the variational posterior parameters with ARD is implemented in the function vb_logit_fit_ard .m, which is called by

[w, V, invV, logdetV, E_a, L] = vb_logit_fit_ard (X, y [, a0, b0])

The inputs X and y, the optional inputs a0 and b0, and the outputs w, V, invV, logdetV, and L have the same meaning as for vb_logit_fit .m. The only difference

is that the returned E_a is now a vector that contains the posterior $\mathsf{E}_{\boldsymbol{\alpha}}(\alpha_i)$'s as its elements.

The implementation of vb_logit_fit_ard .m differs from vb_logit_fit .m only by the variables b_n and E_a, holding the b_{Ni}'s and $\mathsf{E}_{\boldsymbol{\alpha}}(\alpha_i)$'s, now being vectors rather than scalars. Their update and use is adjusted accordingly, in line with what has been described above.

3.5.4. *Predictive density parameters.* Two scripts are available to compute the predictive probability $\mathsf{P}(y = 1|\boldsymbol{x}, \mathcal{D})$, namely vb_logit_pred_iter .m and vb_logit_pred .m. Their only difference is that the latter is a vectorized form of the former. Both are called by

out = vb_logit_pred[_incr](X, w, V, invV)

where X is the $M \times D$ matrix with the input vector $\boldsymbol{x}_m^\top$ as its mth row. The variational posterior parameters w, V, and invV are those returned by either of the estimation function discussed above. The returned vector out with M elements contains the estimated $\mathsf{P}(y_m = 1|\boldsymbol{x}_m, \mathcal{D})$ as its mth element.

In terms of implementation, let us first consider vb_logit_pred_iter .m. This function iterates over all given $\boldsymbol{x}_m$, and optimizes $\boldsymbol{\xi}_m$ for each of these separately. In order to do so, it employs a simplification to logdetV_xi = $\ln |\tilde{\boldsymbol{V}}|/|\boldsymbol{V}_N|$, appearing in $\ln \mathsf{P}(y = 1|\boldsymbol{x}, \mathcal{D})$. From the expression for $\tilde{\boldsymbol{V}}^{-1}$ and the Matrix determinant lemma it can be shown that

$$
\begin{aligned}
\ln |\tilde{\boldsymbol{V}}| &= -\ln |\tilde{\boldsymbol{V}}^{-1}| \\
&= -\ln |\boldsymbol{V}_N^{-1}| - \ln \left(1 + 2\lambda(\xi)\boldsymbol{x}^\top \boldsymbol{V}_N \boldsymbol{x}^\top\right) \\
&= \ln |\boldsymbol{V}_N| - \ln \left(1 + 2\lambda(\xi)\boldsymbol{x}^\top \boldsymbol{V}_N \boldsymbol{x}\right).
\end{aligned}
\tag{96}
$$

Thus, $\ln |\tilde{\boldsymbol{V}}|/|\boldsymbol{V}_N|$ results in

$$
\ln \frac{|\tilde{\boldsymbol{V}}|}{|\boldsymbol{V}_N|} = -\ln \left(1 + 2\lambda(\xi)\boldsymbol{x}^\top \boldsymbol{V}_N \boldsymbol{x}\right)
\tag{97}
$$

Additionally, the Sherman-Morrison formula can be applied to avoid inverting $\tilde{\boldsymbol{V}}^{-1}$ by using

$$
\tilde{\boldsymbol{V}} = \boldsymbol{V}_N - \frac{2\lambda(\xi)\boldsymbol{V}_N \boldsymbol{x}\boldsymbol{x}^\top \boldsymbol{V}_N}{1 + 2\lambda(\xi)\boldsymbol{x}^\top \boldsymbol{V}_N \boldsymbol{x}}
\tag{98}
$$

instead.

The script initially starts with $\xi_m = 0$ for each $\boldsymbol{x}_m$ in lines 36–40, using some initial simplifications based on $\lambda(\xi) = 1/8$, as already previously discussed. It then iterates over updating ξ, $\tilde{\boldsymbol{V}}$ and $\tilde{\boldsymbol{w}}$ in lines 43–68 until the variational bound either changes less than 0.001% between two consecutive iterations, or the number of iterations exceeds 500. The variational bound is computed as a simplified version of $\ln \mathsf{P}(y = 1|\boldsymbol{x}, \mathcal{D})$, omitting all terms that are independent of ξ_m.

The vectorized script vb_logit_pred .m follows exactly the same principles, but optimizes $\boldsymbol{\xi}$ for all $\boldsymbol{x}$ at the same time, by maximizing a sum of the individual variational bounds.
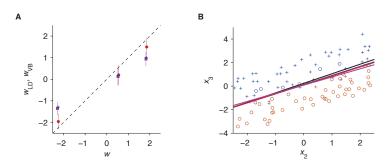
3.6. **Examples.**

FIGURE 4. Estimated coefficients and separating hyperplanes for classifiers trained on low-dimensional training set. A) true coefficients vs. their estimates (mean, $\pm$ 95% CIs where available) for variational Bayesian logistic regression with hyper-prior on $\boldsymbol{w}$ (red), variational Bayesian logistic regression without such hyper-prior (purple), and Fisher's linear discriminant (blue). B) Separating hyperplanes (defined by $w_1 + x_2 w_2 + x_3 w_3 = 0$) for the three classifiers (same colors) and generative separating hyperplane (black). The shown observations (circles / crosses) represent the training data, with symbol type indicating class membership, symbol color indicating class probability (blue barely visible, as hidden by red line).

3.6.1. *Estimation of coefficients and separating hyperplane.* As a first example, consider 50 observations from a 3-dimensional input space, such that $N = 50$ and $D = 3$. The observations are generated by

```
>> D = 3; N = 100; N_test = 1000;  X_scale = 5;
>> w = randn(D, 1);
>> X = [ones(N, 1) (X_scale*(rand(N, 1)−0.5))];
>> X = [X (X_scale*(rand(N, 1)−0.5) − (w(1) + X(:,2) * w(2)) / w(3))];
>> X_test = [ones(N_test, 1) (X_scale*(rand(N_test, 1)−0.5))];
>> X_test = [X_test (X_scale*(rand(N_test, 1)−0.5) − ...
                      (w(1) + X_test(:,2) * w(2)) / w(3))];
>> py = 1 ./ (1 + exp(− X * w));
>> y = 2 * (rand(N, 1) < py) − 1;
>> y_test = 2 * (rand(N_test, 1) < 1 ./ (1 + exp(− X_test * w))) − 1;
```

The different $\boldsymbol{x}_n$'s are generated such that roughly half of all outputs are 1 and the other half are -1. This is achieved by setting the first element, x_{n1}, of each $\boldsymbol{x}_n$ to one, and by drawing the second, $x_{n2} \sim \mathcal{U}(-2.5, 2.5)$ from a uniform distribution over the range $[-2.5, 2.5]$. The third element is then set to $\mathcal{U}(-2.5, 2.5) - (w_1 + x_{n2} w_2)/w_3$, such that, with 50% change, $w_1 + x_{n2} w_2 + x_{n3} w_3 > 0$. The outputs are drawn from $\{-1, 1\}$ according to their probabilities of being 1.

We train two variational Bayesian logistic regression classifiers, one with and one without a hyper-prior on $\boldsymbol{w}$

```
>> [w_VB, V_VB, invV_VB] = vb_logit_fit(X, y);
>> py_VB = vb_logit_pred(X, w_VB, V_VB, invV_VB);
>> py_test_VB = vb_logit_pred(X_test, w_VB, V_VB, invV_VB);
```

```
>> [w_VB1, V_VB1, invV_VB1] = vb_logit_fit_iter(X, y);
>> py_VB1 = vb_logit_pred(X, w_VB1, V_VB1, invV_VB1);
>> py_test_VB1 = vb_logit_pred(X_test, w_VB1, V_VB1, invV_VB1);
```

In addition, we train Fisher's linear discriminant (Bishop, 2006) by

```
>> y1 = y == 1;
>> w_LD = NaN(D, 1);
>> w_LD(2:end) = (cov(X(y1, 2:end)) + cov(X(~y1, 2:end))) \ ...
                (mean(X(y1, 2:end))' − mean(X(~y1, 2:end))');
>> w_LD(1) = − 0.5 * (mean(X(y1, 2:end)) + ...
                      mean(X(~y1, 2:end))) * w_LD(2:end);
>> y_LD = 2 * (X * w_LD > 0) − 1;
>> y_test_LD = 2 * (X_test * w_LD > 0) − 1;
```

These classifiers feature training and test set errors

```
>> fprintf('training set 0−1 loss: FLD = %f, VB = %f, VBiter = %f\n', ...
           mean(y_LD ~= y), ...
           mean(2 * (py_VB > 0.5) − 1 ~= y), ...
           mean(2 * (py_VB1 > 0.5) − 1 ~= y));
training set 0−1 loss: FLD = 0.140000, VB = 0.140000, VBiter = 0.160000
>> fprintf('test     set 0−1 loss: FLD = %f, VB = %f, VBiter = %f\n', ...
           mean(y_test_LD ~= y_test), ...
           mean(2 * (py_test_VB > 0.5) − 1 ~= y_test), ...
           mean(2 * (py_test_VB1 > 0.5) − 1 ~= y_test));
test     set 0−1 loss: FLD = 0.142000, VB = 0.143000, VBiter = 0.149000
```

As can be seen, for this realization of training and test set, all classifiers perform roughly equally well. Their similar performance is also reflected in the similarity of their coefficient estimates (Fig. 4A) and their class-separating hyperplanes (Fig. 4B).

The code for this example is available in vb_logit_example_coeff.m.

3.6.2. *High-dimensional logistic regression with uninformative input dimensions.* To illustrate the use of shrinkage priors on all/individual dimensions, let us consider an example in which the majority of input dimensions are uninformative. We generate the data by

```
>> D = 1000; D_eff = 100; N = 2000; N_test = 10000;
>> w = [randn(D_eff, 1); zeros(D − D_eff, 1)];
>> X = rand(N, D) − 0.5;
>> X_test = rand(N_test, D) − 0.5;
>> py = 1 ./ (1 + exp(− X * w));
>> y = 2 * (rand(N, 1) < py) − 1;
>> py_test = 1 ./ (1 + exp(−X_test * w));
>> y_test = 2 * (rand(N_test, 1) < py_test) − 1;
```

In this case, only the first $D_{\text{eff}} = 100$ out of the $D = 1000$ elements of each $\boldsymbol{x}_n$ are informative about the class label y_n.

As before, we train a variational Bayesian logistic regression classifier without ARD, and with and without hyper-prior on $\boldsymbol{w}$, and additionally one classifier with ARD, by
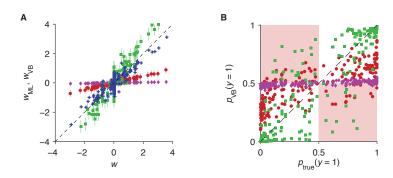
FIGURE 5. Coefficient estimates and output predictions for high-dimensional classification with few informative dimensions. A) true coefficients vs. coefficient estimates (mean, $\pm$ 95% CIs where available) for Fisher's linear discriminant (blue) and variational Bayesian logistic regression without ARD and with (red) or without hyper-priors (purple), and with ARD and hyper-priors on $\boldsymbol{w}$ (green). The figure illustrates the overly strong shrinkage of the methods without ARD (red, purple) due to the large number of uninformative dimensions. B) true class probability vs. estimated class probability for 200 observations of the training set. The colors are the same as in panel A, not showing Fisher's linear discriminant. The red shaded areas are areas in which mis-classification occurs. The strong coefficient shrinkage of methods without ARD is reflected by their predictive class probabilities being strongly biased towards 0.5.

```
>> [w_VB, V_VB, invV_VB] = vb_logit_fit(X, y);
>> py_VB = vb_logit_pred(X, w_VB, V_VB, invV_VB);
>> py_test_VB = vb_logit_pred(X_test, w_VB, V_VB, invV_VB);
>> [w_VB1, V_VB1, invV_VB1] = vb_logit_fit_iter(X, y);
>> py_VB1 = vb_logit_pred(X, w_VB1, V_VB1, invV_VB1);
>> py_test_VB1 = vb_logit_pred(X_test, w_VB1, V_VB1, invV_VB1);
>> [w_VB2, V_VB2, invV_VB2] = vb_logit_fit_ard(X, y);
>> py_VB2 = vb_logit_pred(X, w_VB2, V_VB2, invV_VB2);
>> py_test_VB2 = vb_logit_pred(X_test, w_VB2, V_VB2, invV_VB2);
```

Furthermore, we train Fisher's linear discriminant by

```
>> y1 = y == 1;
>> w_LD = (cov(X(y1, :)) + cov(X(~y1, :))) \ ...
         (mean(X(y1, :))' - mean(X(~y1, :))');
>> c_LD = 0.5 * (mean(X(y1, :)) + mean(X(~y1, :))) * w_LD;
>> y_LD = 2 * (X * w_LD > c_LD) - 1;
>> y_test_LD = 2 * (X_test * w_LD > c_LD) - 1;
```

The above computation of the linear discriminant differs slightly from the previous example, as here, no offset term was included in the input matrix $\boldsymbol{X}$.

Evaluating the classification error on the training and test sets results in

```
>> fprintf(['training set 0-1 loss: FLD  = %f, VB       = %f\n' ...
```

```
             ,                                 VBiter = %f, VB w/ ARD = %f\n'], ...
           mean(y_LD ˜= y), ...
           mean(2 ∗ (py_VB > 0.5) − 1 ˜= y), ...
           mean(2 ∗ (py_VB1 > 0.5) − 1 ˜= y), ...
           mean(2 ∗ (py_VB2 > 0.5) − 1 ˜= y));
training set 0−1 loss: FLD   = 0.041500, VB        = 0.065500
                     VBiter = 0.112500, VB w/ ARD = 0.066000
>> fprintf(['test      set 0−1 loss: FLD   = %f, VB        = %f\n' ...
           ,                                 VBiter = %f, VB w/ ARD = %f\n'], ...
           mean(y_test_LD ˜= y_test), ...
           mean(2 ∗ (py_test_VB > 0.5) − 1 ˜= y_test), ...
           mean(2 ∗ (py_test_VB1 > 0.5) − 1 ˜= y_test), ...
           mean(2 ∗ (py_test_VB2 > 0.5) − 1 ˜= y_test));
test      set 0−1 loss: FLD   = 0.282600, VB        = 0.260300
                     VBiter = 0.280200, VB w/ ARD = 0.203500
```

As can be seen, Fisher's linear discriminant over-fits the training set and thus features a higher test set loss than both variational methods with hyper-prior. To understand the worse performance of the hyper-prior-free variational method (VBiter), it is instructive to investigates its coefficient estimates. As seen in Fig. 5A (purple diamonds), its prior on $\boldsymbol{w}$ with covariance $D^{-1}\boldsymbol{I}$, together with the large input dimensionality, $D = 1000$, and low number of informative inputs $D_{\text{eff}} = 100$, caused the coefficient estimates to be close to zero, such that it severely underestimated these coefficients. This is also reflected in its predictive class probabilities being close to 0.5 (Fig. 5B, purple diamonds). The variational method without ARD but with an adjustable degree of global shrinkage fares slightly better, but still under-estimates the informative coefficients due to the larger number of uninformative inputs (Fig 5A, red circles). This is again reflected in its predictive class probabilities being biased towards 0.5 (Fig. 5B, red circles). Finally, variational Bayesian logistic regression with ARD shows little signs of shrinkage of informative input dimensions (Fig. 5A, green squares), which is also reflected in its wider spread of predictive class probabilities (Fig. 5B, green squares). Note that neither method performs perfectly, due to the task's considerable difficulty.

The code for this example is available in vb_logit_example_highdem.m.

3.6.3. *Model selection by maximizing variational bound.* As for the linear case, the variational bound $\tilde{\mathcal{L}}(\mathsf{Q}, \boldsymbol{\xi})$ can be used as a proxy for the model log-evidence, $\ln \mathsf{P}(\mathcal{D})$, to choose between models of different complexity. We assume that a 1-dimensional scalar input is mapped into a polynomial of order k, which is in turn used to produce the class labels. Given a set of observations of inputs and class labels, the task is to identify this order k and train the associated classifier. Assuming $k = 2$, we generate the data by

```
>> D = 3; N = 50; Ds = 1:10;  x_range = [−5 5];
>> w = randn(D, 1);
>> x = x_range(1) + (x_range(2) − x_range(1)) ∗ rand(N, 1);
>> x_test = linspace(x_range(1), x_range(2), 300)';
>> gen_X = @(x, d) bsxfun(@power, x, 0:(d−1));
>> X = gen_X(x, D);
>> py = 1 ./ (1 + exp(− X ∗ w));
```
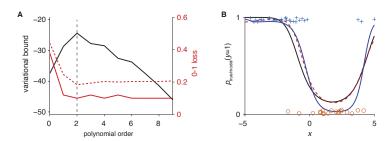
FIGURE 6. Variational bound and loss for models of different complexities, and model predictions. A) variational bound (black curve, left axis) and training/test set loss (solid/dashed red curve, right axis) for different assumed polynomial orders of the input. The black dashed line shows the true polynomial order. B) True class probability (black curve) vs. predicted class probability of variational Bayesian logistic regression (red dashed curve, using best-fit $k = 2$ from A) and Fisher's logistic discriminant (blue curve, assuming $k = 5$) over scalar input x. The blue crosses and orange circles show the provided 50 training samples. They are scattered away from 0 and 1 for better visibility.

```
>> y = 2 * (rand(N, 1) < py) − 1;
>> py_test = 1 ./ (1 + exp(− gen_X(x_test, D) * w));
>> y_test = 2 * (rand(length(py_test), 1) < py_test) − 1;
```

In the above, Ds are the different orders to be tested, and the test data constitutes of 300 inputs spanning the range from -5 to 5. The training data consists of 50 observations which are uniformly randomly distributed in the same range.

We find the variational bound and loss on training and test set for different orders of the polynomial by

```
>> Ls = NaN(1, length(Ds)); pred_loss = NaN(length(Ds), 2);
>> for i = 1:length(Ds)
       [w, V, invV, ˜, ˜, Ls(i)]  =  vb_logit_fit (gen_X(x, Ds(i)), y);
       y_pred =  2 * (vb_logit_pred(gen_X(x, Ds(i)), w, V, invV) > 0.5) − 1;
       y_test_pred  =  ...
           2 * (vb_logit_pred(gen_X(x_test, Ds(i)), w, V, invV) > 0.5) − 1;
       pred_loss(i, :)  = [mean(y_pred ˜= y) mean(y_test_pred ˜= y_test)];
   end
>> [˜, i]  = max(Ls);
>> D_best = Ds(i)
D_best =

    3
```

Thus, for this particular training set, the method was able to correctly identify the underlying complexity. Plotting the variational bound over model complexity shows a clear peak of the variational bound at this order (Fig. 6A). Figure 6A also illustrates that, even though the training set error is equally low for polynomials of

higher order (red, solid curve), Bayesian model selection takes the model complexity into account and thus selects a lower-order polynomial model.

For further testing, we train a Bayesian classifier with the identified polynomial order by

```
>> X_VB = gen_X(x, D_best);
>> [w_VB, V_VB, invV_VB] = vb_logit_fit(X_VB, y);
>> py_VB = vb_logit_pred(X_VB, w_VB, V_VB, invV_VB);
>> py_test_VB = vb_logit_pred(gen_X(x_test, D_best), w_VB, V_VB, invV_VB);
```

Furthermore, we train Fisher's linear discriminant with assume polynomial order 5, by

```
>> D_LD = 6; y1 = y == 1;
>> X_LD = gen_X(x, D_LD);
>> w_LD = NaN(D_LD, 1);
>> w_LD(2:end) = (cov(X_LD(y1, 2:end)) + cov(X_LD(~y1, 2:end))) \ ...
              (mean(X_LD(y1, 2:end))' - mean(X_LD(~y1, 2:end))');
>> w_LD(1) = - 0.5 * (mean(X_LD(y1, 2:end)) + ...
            mean(X_LD(~y1, 2:end))) * w_LD(2:end);
>> y_LD = 2 * (X_LD * w_LD > 0) - 1;
>> y_test_LD = 2 * (gen_X(x_test, D_LD) * w_LD > 0) - 1;
```

leading to training and test set losses of

```
>> fprintf('Training set MSE, LD = %f, VB = %f\n', ...
          mean(y_LD ~= y), mean(2 * (py_VB > 0.5) - 1 ~= y));
Training set MSE, LD = 0.100000, VB = 0.100000
>> fprintf('Test     set MSE, LD = %f, VB = %f\n', ...
          mean(y_test_LD ~= y_test), ...
          mean(2 * (py_test_VB > 0.5) - 1 ~= y_test));
Test     set MSE, LD = 0.190000, VB = 0.183333
```

The slightly higher test-set loss for Fisher's linear discriminant is not surprising, given that its higher model complexity is bound to lead to over-fitting the training set data. However, as illustrated in Fig. 6B, this over-fitting does not seem particularly severe in this particular case, as both the Bayesian estimate and the linear discriminant's estimate closely follow the true class probabilities.

The code for this example is available in vb_logit_example_modelsel.m.

Acknowledgments

References

Beal, M. J. (2003). *Variational Algorithms for Approximate Bayesian Inference.* PhD thesis, Gatsby Computational Neuroscience Unit, University College London.

Bishop, C. M. (2006). *Pattern Recognition and Machine Learning.* Springer-Verlag.

Gelman, A., Carlin, J. B., Sern, H. S., Dunson, D. B., Vehtari, A., and Rubin, D. B. (2013). *Bayesian Data Analysis.* CRC Texts in Statistical Science. Chapman and Hall, 3rd edition.

Gilks, W. R., Richardson, S., and Spiegelhalter, D., editors (1995). *Markov Chain Monte Carlo in Practice.* CRC Interdisiplinary Statistics. Chapman and Hall.

Griffin, J. E. and Brown, P. J. (2010). Inference with normal-gamma prior distributions in regression problems. *Bayesian Analysis*, 5(1):171–188.

Hastie, T., Tibishirani, R., and Friedman, J. (2011). *The Elements of Statistical Learning.* Springer Series in Statistics. Springer-Verlag, 2nd edition.

Jaakkola, T. S. and Jordan, M. M. (2000). Bayesian parameter estimation via variational methods. *Statistics and Computation*, 10:25–37.

MacKay, D. J. C. (1992). Bayesian interpolation. *Neural Computation*, 4(3):415–447.

Murphy, K. P. (2012). *Machine Learning: A Probabilistic Perspective.* Adaptive Computation and Machine Learning Series. The MIT Press.

Neal, R. M. (1996). *Bayesian Learning for Neural Networks.* Springer-Verlag.

Tibshirani, R. (1996). Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society B*, 58(1):267–288.

Tipping, M. E. (2001). Sparse bayesian learning and the relevance vector machine. *Journal of Machine Learning Research*, 1:211–244.

Wipf, D. P. and Nagarajan, S. S. (2007). A new view of automatic relevance determination. In Platt, J. C., Koller, D., Singer, Y., and Roweis, S. T., editors, *NIPS.* Curran Associates, Inc.

DEPARTMENT DES NEUROSCIENCES FONDAMENTALES, UNIVERSITÉ DE GENÈVE, 1211 GENÈVE, SWITZERLAND

E-mail address: `jdrugo@gmail.com`