Journal of Machine Learning Research 19 (2018) 1-30 Submitted 11/17; Revised 8/18; Published 9/18

Accelerating Cross-Validation in Multinomial Logistic Regression with ℓ_1-Regularization

Tomoyuki Obuchi OBUCHI@C.TITECH.AC.JP

Yoshiyuki Kabashima KABA@C.TITECH.AC.JP

Department of Mathematical and Computing Science
Tokyo Institute of Technology
2-12-1, Ookayama, Meguro-ku, Tokyo, Japan

Editor: Manfred Opper

Abstract

We develop an approximate formula for evaluating a cross-validation estimator of predictive likelihood for multinomial logistic regression regularized by an ℓ_1-norm. This allows us to avoid repeated optimizations required for literally conducting cross-validation; hence, the computational time can be significantly reduced. The formula is derived through a perturbative approach employing the largeness of the data size and the model dimensionality. An extension to the elastic net regularization is also addressed. The usefulness of the approximate formula is demonstrated on simulated data and the ISOLET dataset from the UCI machine learning repository. MATLAB and python codes implementing the approximate formula are distributed in (Obuchi, 2017; Takahashi and Obuchi, 2017).

Keywords: classification, multinomial logistic regression, cross-validation, linear perturbation, self-averaging approximation

1. Introduction

Multinomial classification is a ubiquitous task. There are several ways to treat this task, such as the naive Bayesian methods, neural networks, decision trees, and hierarchical classification schemes (Trevor et al., 2009). Among them, in this paper, we focus on multinomial logistic regression (MLR), which is simple but powerful enough to be used in many present day applications.

Let us denote each feature vector by $\boldsymbol{x}_\mu \in \mathbb{R}^N$ and its class by $y_\mu \in \{1, \cdots, L\}$, where $\mu = 1 \cdots, M$ denotes the index of given data. The MLR uses a linear structural model with parameters $\{\boldsymbol{w}_a \in \mathbb{R}^N\}_{a=1}^L$ and computes a class-a bias as an overlap:

$$u_{\mu a} = \boldsymbol{x}_\mu^\top \boldsymbol{w}_a. \tag{1}$$

A probability such that the feature vector $\boldsymbol{x}_\mu$ belongs to the class a is computed through a softmax function ϕ as:

$$\phi\left(a \middle| \{u_{\mu b}\}_{b=1}^L\right) = \frac{e^{u_{\mu a}}}{\sum_{b=1}^L e^{u_{\mu b}}}. \tag{2}$$

These define the MLR.

©2018 Tomoyuki Obuchi and Yoshiyuki Kabashima.

License: CC-BY 4.0, see `https://creativecommons.org/licenses/by/4.0/`. Attribution requirements are provided at `http://jmlr.org/papers/v19/17-684.html`.

The maximum likelihood estimation is usually employed to train the MLR, though the learning result tends to be inefficient when the data size is not sufficiently larger than the model dimensionality or noises in relevant levels are present. A common technique to overcome this difficulty is to introduce a penalty or regularization. In this paper, we use an ℓ_1-regularization, which induces a sparse classifier as a learning result and is accepted to be effective. Given M data points $D^M \equiv \{(\boldsymbol{x}_\mu, y_\mu)\}_{\mu=1}^M$, the ℓ_1-regularized estimator is defined by the following optimization problem:

$$\{\hat{\boldsymbol{w}}_a(\lambda)\}_a = \mathop{\arg\min}_{\{\boldsymbol{w}_a\}_a} \left\{ \mathcal{H}\left(\{\boldsymbol{w}_a\}_{a=1}^L \middle| D^M, \lambda \right) \right\}, \tag{3}$$

$$\mathcal{H}\left(\{\boldsymbol{w}_a\}_{a=1}^L \middle| D^M, \lambda \right) \equiv \sum_{\mu=1}^{M} q_\mu \left(\{\boldsymbol{w}_a\}_{a=1}^L \right) + \lambda \sum_{a=1}^{L} ||\boldsymbol{w}_a||_1, \tag{4}$$

$$q_\mu \left(\{\boldsymbol{w}_a\}_{a=1}^L \right) = -\ln \phi \left(y_\mu \middle| \left\{ u_{\mu a} = \boldsymbol{x}_\mu^\top \boldsymbol{w}_a \right\}_{a=1}^L \right), \tag{5}$$

where we denote the negative log-likelihood as q_μ and define a regularized cost function or Hamiltonian $\mathcal{H}$.

The introduction of regularization causes another problem of model selection or hyperparameter estimation with respect to λ. A versatile framework providing a reasonable estimate is cross-validation (CV), but it has a disadvantage in terms of the computational cost. The literal CV requires repeated optimizations which can be a serious computational burden when the data size and the model dimensionality are large. The purpose of this paper is to resolve this problem by inventing an efficient approximation of CV.

Our technique is based on a perturbative expansion employing the largeness of the data size and the model dimensionality. Similar techniques were also developed for the Bayesian learning of simple perceptron and committee machine (Opper and Winther, 1996, 1997), for Gaussian process and support vector machine (Opper and Winther, 2000a,b; Vapnik and Chapelle, 2000), for linear regression with the ℓ_1-regularization (Obuchi and Kabashima, 2016; Rad and Maleki, 2018; Wang et al., 2018) and with the two-dimensional total variation (Obuchi et al., 2017). Actually, this perturbative approach is fairly general and can be applied to a wide class of generalized linear models with simple convex regularizations. For example in the present MLR case, it is easy to extend our result to the case where both the ℓ_1- and ℓ_2-regularizations exist (elastic net, Zou and Hastie, 2005), which is used in a common implementation (Friedman et al., 2010). The derivation of our approximate formula below is, however, conducted on the case of the ℓ_1-regularization only, for simplicity. The extension to the elastic net case is stated after the derivation.

The rest of the paper is organized as follows. In sec. 2, we state our formulation and how to derive the approximate formula. In sec. 3, we compare our approximation result with that of the literally conducted CV on simulated data and on the ISOLET dataset from UCI machine learning repository (Lichman, 2013). The accuracy and the computational time of our approximate formula are reported in comparison with the literal CV. The limitation of a simplified version of the approximation is also examined. The last section is devoted to the conclusion.

2. Formulation

In the maximum likelihood estimation framework, it is natural to employ a predictive likelihood as a criterion for model selection (Bjornstad, 1990; Ando and Tsay, 2010). We require a good estimator of the predictive likelihood, and the CV provides a simple realization of it. Particularly in this paper, we consider an estimator based on the leave-one-out (LOO) CV. The LOO solution is described by

$$\{\hat{\boldsymbol{w}}_a^{\backslash\mu}(\lambda)\}_a = \underset{\{\boldsymbol{w}_a\}_a}{\arg\min} \left\{ \mathcal{H}^{\backslash\mu}\left(\{\boldsymbol{w}_a\}_{a=1}^L \middle| D^M, \lambda \right) \right\}, \tag{6}$$

$$\mathcal{H}^{\backslash\mu}\left(\{\boldsymbol{w}_a\}_{a=1}^L \middle| D^M, \lambda \right) \equiv \mathcal{H}\left(\{\boldsymbol{w}_a\}_{a=1}^L \middle| D^M, \lambda \right) - q_\mu\left(\{\boldsymbol{w}_a\}_{a=1}^L \right). \tag{7}$$

Denoting the overlap of $\boldsymbol{x}_\mu$ with the LOO solution as $\hat{u}_{\mu a}^{\backslash\mu} = \boldsymbol{x}_\mu^\top \hat{\boldsymbol{w}}_a^{\backslash\mu}$, as well as that with the full solution $\hat{u}_{\mu a} = \boldsymbol{x}_\mu^\top \hat{\boldsymbol{w}}_a$, we can define the LOO estimator (LOOE) of the predictive negative log-likelihood as:

$$\epsilon_{\mathrm{LOO}}(\lambda) = \frac{1}{M}\sum_{\mu=1}^M q_\mu\left(\left\{ \hat{\boldsymbol{w}}_a^{\backslash\mu} \right\}_{a=1}^L \right) = -\frac{1}{M}\sum_{\mu=1}^M \ln\phi\left(y_\mu \middle| \{\hat{u}_{\mu a}^{\backslash\mu}\}_{a=1}^L \right). \tag{8}$$

In the following, the predictive negative log-likelihood is simply called prediction error. The minimum of the LOOE determines the optimal value of λ though its evaluation requires us to solve eq. (6) M times, which is computationally demanding.

2.0.1. Notations

Here, we fix the notations for a better flow of the derivation shown below. By summarizing the class index, we introduce a vector notation of the overlap as $\boldsymbol{u}_\mu = (u_{\mu a})_a \in \mathbb{R}^L$ and an extended vector representation of the weight vectors $\{\boldsymbol{w}_a\}_a$ as $\boldsymbol{W} = (\boldsymbol{w}_a)_a \in \mathbb{R}^{LN}$. The mth component of $\boldsymbol{W}$ can thus be decomposed into two parts as $m = (m_c, m_f)$ where $m_c \in \{1, \cdots, L\}$ denotes the class index and $m_f \in \{1, \cdots, N\}$ represents the component index of the feature vector. Namely we write $W_m = w_{m_c m_f}$. Correspondingly, we leverage a matrix $X^\mu \in \mathbb{R}^{L\times LN}$ to define a repetition representation of the feature vector $\boldsymbol{x}_\mu$: Each component is defined as:

$$X_{am}^\mu \equiv \delta_{a m_c} x_{\mu m_f}. \tag{9}$$

This yields simple and convenient relations:

$$\boldsymbol{u}_\mu = X^\mu \boldsymbol{W},\ X^\mu = \left(\frac{\partial \boldsymbol{u}_\mu}{\partial \boldsymbol{W}} \right)^\top. \tag{10}$$

Further, the class-a probability of μth data at the full solution $\hat{\boldsymbol{W}} = (\hat{\boldsymbol{w}}_a)_a$ is denoted by:

$$p_{a|\mu} = \phi(a|\{\hat{u}_{\mu b}\}_b) = \frac{e^{\hat{u}_{\mu a}}}{\sum_{b=1}^L e^{\hat{u}_{\mu b}}} \tag{11}$$

These notations express the gradient and the Hessian of q_μ at the full solution as:

$$\nabla q_\mu(\hat{\boldsymbol{W}}) \equiv \left.\frac{\partial q_\mu}{\partial \boldsymbol{W}}\right|_{\boldsymbol{W}=\hat{\boldsymbol{W}}} = \frac{\partial \boldsymbol{u}_\mu}{\partial \boldsymbol{W}} \left.\frac{\partial}{\partial \boldsymbol{u}_\mu} q_\mu\right|_{\boldsymbol{u}_\mu=\hat{\boldsymbol{u}}_\mu} = (X^\mu)^\top \boldsymbol{b}^\mu, \tag{12}$$

$$\partial^2 q_\mu(\hat{\boldsymbol{W}}) \equiv \left.\frac{\partial^2 q_\mu}{\partial \boldsymbol{W} \partial \boldsymbol{W}'}\right|_{\boldsymbol{W}=\boldsymbol{W}'=\hat{\boldsymbol{W}}}$$
$$= \frac{\partial \boldsymbol{u}_\mu}{\partial \boldsymbol{W}} \left(\left.\frac{\partial^2 q_\mu}{\partial \boldsymbol{u}_\mu \partial \boldsymbol{u}'_\mu}\right|_{\boldsymbol{u}_\mu=\boldsymbol{u}'_\mu=\hat{\boldsymbol{u}}_\mu} \right) \left(\frac{\partial \boldsymbol{u}'_\mu}{\partial \boldsymbol{W}'} \right)^\top = (X^\mu)^\top F^\mu X^\mu, \tag{13}$$

where

$$\boldsymbol{b}^\mu \equiv (p_{1|\mu} - \delta_{1y_\mu}, p_{2|\mu} - \delta_{2y_\mu}, \cdots, p_{L|\mu} - \delta_{Ly_\mu})^\top, \tag{14}$$
$$F^\mu_{ab} \equiv \delta_{ab} p_{a|\mu} - p_{a|\mu} p_{b|\mu}. \tag{15}$$

In addition, we denote the cost function Hessians at the respective solutions as:

$$G \equiv \partial^2 \mathcal{H}(\hat{\boldsymbol{W}}) = \sum_\mu \left(\partial^2 q_\mu(\hat{\boldsymbol{W}}) \right), \tag{16}$$

$$G^{\backslash \mu} \equiv \partial^2 \mathcal{H}^{\backslash \mu}(\hat{\boldsymbol{W}}^{\backslash \mu}) = \sum_{\nu (\neq \mu)} \left(\partial^2 q_\nu(\hat{\boldsymbol{W}}^{\backslash \mu}) \right). \tag{17}$$

Finally, we introduce the symbol $A(\boldsymbol{W}) \equiv \{m | W_m \neq 0\}$ representing the index set of the active components of $\boldsymbol{W}$ and $\hat{A} \equiv A(\hat{\boldsymbol{W}})$. Given $\hat{\boldsymbol{W}}$, we denote the active components of a vector $\boldsymbol{Y} \in \mathbb{R}^{LN}$ by the subscript as $\boldsymbol{Y}_{\hat{A}}$. A similar notation is used for any matrix and the symbol $*$ is assumed to represent all of the components in the corresponding dimension.

2.1. Approximate formula

For a simple derivation, it is important to consider that the $\boldsymbol{w}$-dependence of ϕ appears only in the overlap $u = \boldsymbol{x}^\top \boldsymbol{w}$. Hence, it is sufficient to provide the relation between $\hat{u}_{\mu a}$ and $\hat{u}^{\backslash \mu}_{\mu a}$ in order to derive the approximate formula.

A crucial assumption to derive the formula is that the active set is “common” between the full and LOO solutions, $\hat{\boldsymbol{W}} = (\hat{\boldsymbol{w}}_a)_a$ and $\hat{\boldsymbol{W}}^{\backslash \mu} = (\hat{\boldsymbol{w}}^{\backslash \mu}_a)_a$; namely $\hat{A} = \hat{A}^{\backslash \mu} \equiv A(\hat{\boldsymbol{W}}^{\backslash \mu})$. Although this assumption is literally not true, we numerically confirmed that this approximately holds. In other words, the change of the active set is small enough compared to the size of the active set itself when considering the LOO operation when N and M are large. Moreover, in a related problem of an ℓ_1-regularized linear regression, the so-called LASSO, it has been shown that the contribution of the active set change vanishes in a limit $N, M \to \infty$ keeping $\alpha = M/N = O(1)$ (Obuchi and Kabashima, 2016). It is expected that the same holds in the present problem. Hence, we adopt this assumption in the following definition. Note that this idea of the active set constantness can be found in preceding analyses of support vector machine (Opper and Winther, 2000b; Vapnik and Chapelle, 2000).

Once the active set $\hat{A}$ is assumed to be known and unchanged by the LOO operation, it is easy to determine the active components of the full and LOO solutions $\hat{\boldsymbol{W}}$ and $\hat{\boldsymbol{W}}^{\backslash \mu}$.

The vanishing condition of the gradient of the cost function is the determining equation:

$$(\nabla \mathcal{H})_{\hat{A}} = 0 \Rightarrow \hat{\boldsymbol{W}}_{\hat{A}}, \tag{18}$$

$$\left(\nabla \mathcal{H}^{\backslash \mu}\right)_{\hat{A}} = (\nabla \mathcal{H})_{\hat{A}} - (\nabla q_\mu)_{\hat{A}} = 0 \Rightarrow \hat{\boldsymbol{W}}^{\backslash \mu}_{\hat{A}}. \tag{19}$$

The difference between the gradients is only ∇q_μ, and hence the difference between $\hat{\boldsymbol{W}}$ and $\hat{\boldsymbol{W}}^{\backslash \mu}$ is expected to be small. Denoting the difference as $\boldsymbol{d}^\mu = \hat{\boldsymbol{W}} - \hat{\boldsymbol{W}}^{\backslash \mu}$ and expanding eq. (19) with respect to $\boldsymbol{d}^\mu$ up to the first order, we obtain an equation determining $\boldsymbol{d}^\mu$:

$$\boldsymbol{d}^\mu_{\hat{A}} = -\left(G^{\backslash \mu}_{\hat{A}\hat{A}}\right)^{-1} \left(\nabla q_\mu(\hat{\boldsymbol{W}})\right)_{\hat{A}}. \tag{20}$$

Inserting this and eq. (12) into the definition $\boldsymbol{d}^\mu = \hat{\boldsymbol{W}} - \hat{\boldsymbol{W}}^{\backslash \mu}$ and multiplying X^μ from left, we obtain:

$$\hat{\boldsymbol{u}}^{\backslash \mu}_{\mu} \approx \hat{\boldsymbol{u}}_\mu + C^{\backslash \mu}_{\mu} \boldsymbol{b}^\mu, \tag{21}$$

$$C^{\backslash \mu}_{\mu} \equiv X^\mu_{*\hat{A}} \left(G^{\backslash \mu}_{\hat{A}\hat{A}}\right)^{-1} \left(X^\mu_{*\hat{A}}\right)^\top. \tag{22}$$

This equation implies that the matrix inversion operation is necessary for each μ, which still requires a significant computational cost. To avoid this, we employ an approximation and the Woodbury matrix inversion formula in conjunction with eqs. (13,16,17). The result is:

$$\begin{aligned}\left(G^{\backslash \mu}\right)^{-1} &\equiv \left(\partial^2 \mathcal{H}^{\backslash \mu}(\hat{\boldsymbol{W}}^{\backslash \mu})\right)^{-1} \approx \left(\partial^2 \mathcal{H}^{\backslash \mu}(\hat{\boldsymbol{W}})\right)^{-1} = \left(G - (X^\mu)^\top F^\mu X^\mu\right)^{-1} \\ &= G^{-1} - G^{-1} (X^\mu)^\top \left(-F^\mu + X^\mu G^{-1} (X^\mu)^\top\right)^{-1} X^\mu G^{-1}. \end{aligned} \tag{23}$$

Inserting this into eq. (21) and simplifying several factors, we obtain:

$$\hat{\boldsymbol{u}}^{\backslash \mu}_{\mu} \approx \hat{\boldsymbol{u}}_\mu + C_\mu \left(I_L - F^\mu C_\mu\right)^{-1} \boldsymbol{b}^\mu, \tag{24}$$

where

$$C_\mu = X^\mu_{*\hat{A}} \left(G_{\hat{A}\hat{A}}\right)^{-1} \left(X^\mu_{*\hat{A}}\right)^\top. \tag{25}$$

Now, all of the variables on the righthand side of eq. (24) can be computed from the full solution $\hat{\boldsymbol{W}}$ only, which enables us to estimate the LOOE by leveraging a one-time optimization using all of the data D^M, while avoiding repeated optimizations.

We should mention the computational cost of this approximation: it is mainly scaled as $O(ML^2|\hat{A}| + ML|\hat{A}|^2 + |\hat{A}|^3)$. The first two terms come from the construction of $G_{\hat{A}\hat{A}}$ and C_μ, and the last one is derived from the inverse of G. If $|\hat{A}|$ is proportional to the feature dimensionality N, this computational cost is of the third order with respect to the system dimensionality N and M. This is admittedly not cheap and the computational cost for the k-fold literal CV with a moderate value of k becomes smaller than that for our approximation in a large dimensionality limit. We, however, stress that there actually exists a wide range of N and M values in which our approximation outperforms the literal

CV in terms of the computational time, as later demonstrated in sec. 3. Moreover, for treating much larger systems, we invent a further simplified approximation based on the above approximate formula. The computational cost of this simplified version is scaled only linearly with respect to the system parameters N and M. Its derivation is in sec. 2.2 and the precision comparison to the original approximation is in sec. 3.

Another sensitive issue is present in computing $\left(G_{\hat{A}\hat{A}}\right)^{-1}$. Occasionally the cost function Hessian G has zero eigenvalues and is not invertible. We handle this problem in the next subsection.

2.1.1. Handling zero modes

In the MLR, there is an intrinsic symmetry such that the model is invariant under the addition of any constant vector to the weight vectors of all classes:

$$\boldsymbol{w}_a \to \boldsymbol{w}_a + \boldsymbol{v} \ (\forall a). \tag{26}$$

In this sense, the weight vectors defining the same model are "degenerated" and our MLR is singular. For finite λ, this is not harmful because the regularization term resolves this singularity and selects an optimal one $\{\hat{\boldsymbol{w}}_a\}_a$ with the smallest value of $||\boldsymbol{w}_a||_1$ among the degenerated vectors. However, this does not mean that the associated Hessian is non-singular. The regularization term does not provide any direct contribution to the Hessian and as a result, the Hessian tends to have some zero modes. This prevents taking the inverse Hessian G^{-1} in eq. (25). How can we overcome this?

One possibility is to fix the weights of one certain class at constant values when solving the optimization problem (4). This is termed "gauge fixing" in physics, and one convenient gauge in the present problem will be the zero gauge in which the weights in a chosen class are fixed at zeros. This is actually found in some earlier implementations (Krishnapuram et al., 2005; Schmidt, 2010) and is preferable for our approximate formula because it removes the harmful zero modes of the Hessian from the beginning. However, some other implementations which are currently well accepted do not employ such gauge fixing (Friedman et al., 2010), and moreover even with gauge fixing very small eigenvalues sometimes accidentally emerge in the Hessian. Hence, for user convenience, we require another way of avoiding this problem.

Another possibility is to remove the zero modes by hand. By construction, the zero modes are associated to the model invariance. This implies that those zero modes are irrelevant and may be removed. In fact, we are only interested in the perturbations which truly change the model, and the modes which maintain the model unchanged are unnecessary. According to this consideration, we replace G^{-1} in eq. (25) with the zero-mode-removed inverse Hessian $\overline{G}^{-1}$. The computation of $\overline{G}^{-1}$ is straightforward: we perform the eigenvalue decomposition of $G_{\hat{A}\hat{A}}$ and obtain the eigenvalues $\{d_i\}_{i=1}^{|\hat{A}|}$ and eigenvectors $\{\boldsymbol{v}_i\}_{i=1}^{|\hat{A}|}$, which allows us to represent

$$G_{\hat{A}\hat{A}} = \sum_i d_i \boldsymbol{v}_i \boldsymbol{v}_i^\top = \sum_{i \in S^+} d_i \boldsymbol{v}_i \boldsymbol{v}_i^\top, \tag{27}$$

where S^{+} denotes the index set of the modes with finite eigenvalues. Then, $\overline{G}^{-1}$ is defined as:

$$\overline{G}^{-1}_{\hat{A}\hat{A}} \equiv \sum_{i \in S^{+}} d_i^{-1} \boldsymbol{v}_i \boldsymbol{v}_i^{\top}. \tag{28}$$

Finally, we replace G^{-1} by $\overline{G}^{-1}$ in eq. (25), and obtain:

$$C_{\mu} = X^{\mu}_{*\hat{A}} \overline{G}^{-1}_{\hat{A}\hat{A}} \left(X^{\mu}_{*\hat{A}} \right)^{\top}. \tag{29}$$

By using this instead of eq. (25), the problem caused by the zero modes can be avoided.

2.1.2. Extension to the mixed regularization case

Let us briefly state how we can generalize the present result to the case of the mixed regularizations of the ℓ_1- and ℓ_2-terms (elastic net, Zou and Hastie, 2005). The problem to be solved can be defined as follows:

$$\{\hat{\boldsymbol{w}}_a(\lambda_1, \lambda_2)\}_a = \underset{\{\boldsymbol{w}_a\}_a}{\arg\min} \left\{ \sum_{\mu=1}^{M} q_{\mu}\left(\{\boldsymbol{w}_a\}_{a=1}^{L} \right) + \lambda_1 \sum_{a=1}^{L} ||\boldsymbol{w}_a||_1 + \frac{\lambda_2}{2} \sum_{a=1}^{L} ||\boldsymbol{w}_a||_2^2 \right\}. \tag{30}$$

where $|| \cdot ||_2$ denotes the ℓ_2 norm. Following the derivation in sec. 2.1, we realize that the derivation is essentially the same, and the difference only appears in the cost function Hessian:

$$G_{\mathrm{mxd}} = \sum_{\mu} \left(\partial^2 q_{\mu}(\hat{\boldsymbol{W}}) \right) + \lambda_2 I_{NL}, \tag{31}$$

where I_K is the identity matrix of size K. As a result, we can compute the LOO solution by leveraging the same equation as eq. (24) by replacing the definition of C_{μ}, eq. (25), with:

$$C_{\mu} = X^{\mu}_{*\hat{A}} \left((G_{\mathrm{mxd}})_{\hat{A}\hat{A}} \right)^{-1} \left(X^{\mu}_{*\hat{A}} \right)^{\top}. \tag{32}$$

Thanks to the ℓ_2 term, the zero mode removal is not needed since the eigenvalues are lifted up by λ_2.

2.1.3. Binomial case

The binomial case $L = 2$ is particularly interesting in several applications and thus we write down the specific formula for this case.

In the binomial case, it is fairly common to express the class y as a binary $y = 0, 1$ and to use the following logit function:

$$\phi_{\mathrm{logit}}(y_{\mu}|u_{\mu}) = \frac{\delta_{y_{\mu}1} + \delta_{y_{\mu}0} e^{-u_{\mu}}}{1 + e^{-u_{\mu}}}, \tag{33}$$

where

$$u_{\mu} = \boldsymbol{x}_{\mu}^{\top} \boldsymbol{w}. \tag{34}$$

If we identify $y = 0$ in this case as $y = 1$ in the two-class MLR case, this is nothing but the two-class MLR with a zero gauge $\boldsymbol{w}_1 = \boldsymbol{0}$. Hence, there is no harmful zero mode in the Hessian and we can straightforwardly apply our approximate formula. The explicit form in this case is:

$$\hat{u}_{\mu}^{\backslash \mu} \approx \hat{u}_{\mu} + \frac{c^{\mu}}{1 - \frac{\partial^2 q_{\mu}}{\partial u_{\mu}^2} c^{\mu}} \frac{\partial q_{\mu}}{\partial u_{\mu}}, \tag{35}$$

where $q_{\mu} = -\ln \phi_{\mathrm{logit}}(y_{\mu}|u_{\mu})$ and

$$\frac{\partial q_{\mu}}{\partial u_{\mu}} = \delta_{y_{\mu}0} - \frac{e^{-u_{\mu}}}{1 + e^{-u_{\mu}}}, \tag{36}$$

$$\frac{\partial^2 q_{\mu}}{\partial u_{\mu}^2} = \frac{e^{-u_{\mu}}}{(1 + e^{-u_{\mu}})^2}, \tag{37}$$

$$G_{\hat{A}\hat{A}} = \sum_{\mu=1}^{M} \frac{\partial^2 q_{\mu}}{\partial u_{\mu}^2} \left(\boldsymbol{x}_{\mu} \boldsymbol{x}_{\mu}^{\top} \right)_{\hat{A}\hat{A}}, \tag{38}$$

$$c^{\mu} = \left(\boldsymbol{x}_{\mu}^{\top} \right)_{\hat{A}} \left(G_{\hat{A}\hat{A}} \right)^{-1} \left(\boldsymbol{x}_{\mu} \right)_{\hat{A}}, \tag{39}$$

and $\hat{A} = \{i | \hat{w}_i \neq 0\}$ is the active set of the full solution, as before.

Note that this approximation can be easily generalized to arbitrary differentiable output functions by replacing the logit function ϕ_{logit}. Readers are thus encouraged to implement approximate CVs in a variety of different problems.

2.2. Further simplified approximation

As mentioned above, the computational cost of our approximation is $O(ML^2|\hat{A}| + ML|\hat{A}|^2 + |\hat{A}|^3)$ and should be reduced for treating larger systems. For this, we derive a further simplified approximation based on the invented approximate formula above. We call this a self-averaging (SA) approximation according to physics terminology.

The basic idea for simplifying our approximate formula is to assume that correlations between W_m and W_n are sufficiently weak. The meaning of "correlation" is not evident here, but as seen in sec. A the Hessian G can be connected to a (rescaled) covariance χ between W_m and W_n in a statistical mechanical formulation introducing a probability distribution of $\boldsymbol{W}$. Our weak correlation assumption requires that the correlation between different feature components is negligibly small; $\chi_{mn} (\equiv (1/\beta) \mathrm{cov}(W_m, W_n)) = \chi_{(m_f, m_c),(n_f, n_c)} = \delta_{m_f n_f} (\chi_{m_f})_{m_c n_c}$, where $m_c, n_c (= 1, \cdots, L)$ are the class indices and $m_f, n_f (= 1, \cdots, N)$ are the feature component indices defined thus far, and β is the rescaling factor. In this way, the Hessian is assumed to be expressed in a rather restricted form:

$$\left(G^{\backslash \mu} \right)^{-1}_{mn} \approx \begin{cases} \left(\chi_{m_f} \right)_{m_c n_c} \delta_{m_f n_f}, & (m, n \in \hat{A}) \\ 0, & (\text{otherwise}) \end{cases}, \tag{40}$$

Namely, the SA Hessian is allowed to take finite values if and only if the two indices share the same feature vector component. The dependence on the data index μ is also assumed to be negligible, implying that strong heterogeneity among feature vectors is assumed to be absent.

To proceed with the computation, we require a closed equation to determine the $L \times L$ matrix χ_i for $i = 1, \cdots, N$. Its derivation is rather technical and is deferred to sec. A. The result is:

$$(\chi_i)_{\hat{A}_i\hat{A}_i} = \left(\lambda_2 I_{|\hat{A}_i|} + \sigma_x^2 \sum_{\nu=1}^{M} \left(\left(I_L + F^\nu C_{\mathrm{SA}} \right)^{-1} F^\nu \right)_{\hat{A}_i\hat{A}_i} \right)^{-1}, \tag{41}$$

where $\sigma_x^2 = \sum_\mu \sum_i x_{\mu i}^2/(NM)$ and $\hat{A}_i = \{a | \hat{w}_{ai} \neq 0\}$ is the set of active class variables at the feature component i; the other components of χ_i related to inactive variables are zeros. The SA approximation of $C_\mu^{\backslash\mu}$, $C_{\mathrm{SA}} \in \mathbb{R}^{L\times L}$, is defined by:

$$C_{\mathrm{SA}} = \sigma_x^2 \sum_{i=1}^{N} \chi_i. \tag{42}$$

Using the solution of eqs. (41,42), the approximate formula is now simply expressed as:

$$\hat{\boldsymbol{u}}_\mu^{\backslash\mu} \approx \hat{\boldsymbol{u}}_\mu + C_{\mathrm{SA}} \boldsymbol{b}^\mu. \tag{43}$$

Note that there is no factor like $(I_L - F^\mu C_\mu)^{-1}$ in contrast to eq. (24), because we directly approximate $C_\mu^{\backslash\mu}$ in eq. (21).

When solving eqs. (41,42), the inverse at the right-hand side of eq. (41) becomes occasionally ill-defined again due to the presence of zero modes. In such cases, we should remove the zero modes as eq. (28). Putting $R = \lambda_2 I_L + \sigma_x^2 \sum_{\mu=1}^{M} \left(\left(I_L + F^\mu C_{\mathrm{SA}} \right)^{-1} F^\mu \right)$ and performing the eigenvalue decomposition, we define its zero-mode-removed inverse $\overline{R}^{-1}$ as:

$$R_{\hat{A}_i\hat{A}_i} = \sum_j d_j \boldsymbol{v}_j \boldsymbol{v}_j^\top = \sum_{j\in S^+} d_j \boldsymbol{v}_j \boldsymbol{v}_j^\top \Rightarrow \overline{R}^{-1}_{\hat{A}_i\hat{A}_i} = \sum_{j\in S^+} d_j^{-1} \boldsymbol{v}_j \boldsymbol{v}_j^\top, \tag{44}$$

where S^+ is the index set of the modes with finite eigenvalues. This requires a $O(L^3)$ computational cost at a maximum. Leveraging this approach, a naive way to solve eqs. (41,42) is a recursive substitution. If this converges in a constant time, irrespectively of the system parameters N, M and L, then the computational cost of the SA approximation is scaled as $O(NL^3 + ML^3)$. This is linear in the feature dimensionality N and the data size M and hence, its advantage is significant.

2.3. Summary of procedures

Here, we summarize the two versions of the approximation derived thus far as algorithmic procedures. We call the first version, based on eq. (24), the approximate CV or ACV, and call the second one, using eq. (43), the self-averaging approximate CV or SAACV. The procedures of ACV and SAACV are given in Alg. 1 and Alg. 2, respectively; they are written for the case of the mixed regularization (30). Comments are added for

Algorithm 1 Approximate CV of the MLR

```
1: procedure ACV($\hat{\boldsymbol{W}}(\lambda_1, \lambda_2), D^M, \lambda_2$)
2:     Compute the active set $\hat{A}$ from $\hat{\boldsymbol{W}}$
3:     Compute $\{\hat{\boldsymbol{u}}_\mu, X^\mu, \boldsymbol{b}^\mu, F^\mu\}_\mu$ by eqs. (1,9), (14) and (15)
4:     $G_{\hat{A}\hat{A}} \leftarrow \sum_{\mu=1}^{M} (X^\mu)^\top F^\mu X^\mu + \lambda_2 I_{|\hat{A}|}$        ▷ $O(ML|\hat{A}|^2 + ML^2|\hat{A}|)$
5:     if $\lambda_2$ is large enough then        ▷ $O(|\hat{A}|^3)$
6:         $\overline{G}^{-1}_{\hat{A}\hat{A}} = (G_{\hat{A}\hat{A}})^{-1}$
7:     else
8:         Compute $\overline{G}^{-1}_{\hat{A}\hat{A}}$ by eq. (28)
9:     end if
10:    for $\mu = 1, \cdots, M$ do        ▷ $O(ML|\hat{A}|^2 + ML^2|\hat{A}| + ML^3)$
11:        $C_\mu \leftarrow X^\mu_{*\hat{A}} \overline{G}^{-1}_{\hat{A}\hat{A}} \left(X^\mu_{*\hat{A}}\right)^\top$
12:        $\hat{\boldsymbol{u}}^{\backslash\mu}_\mu \leftarrow \hat{\boldsymbol{u}}_\mu + C_\mu (I_L - F^\mu C_\mu)^{-1} \boldsymbol{b}^\mu$
13:    end for
14:    Compute $\epsilon_{\mathrm{LOO}}$ from $\{\boldsymbol{u}^{\backslash\mu}_\mu\}_\mu$ by eq. (8)
15:    return $\epsilon_{\mathrm{LOO}}$
16: end procedure
```

specifying the time consuming parts in the entire procedures. In Alg. 2, we describe an actual implementation for solving C_{SA} by recursion, which is not fully specified in sec. 2.2. The symbol $||\cdot||_{\mathrm{F}}$ denotes the Frobenius norm and we set the threshold θ judging the convergence as $\theta = 10^{-6}$ in typical situations. We also set as 10^{-6} the threshold judging if λ_2 is large or not.

3. Numerical experiments

In this section, we examine the precision and actual computational time of ACV and SAACV in numerical experiments. Both simulated and actual datasets (from UCI machine learning repository, Lichman, 2013) are used.

For examination, we compute the errors also by literally conducting k-fold CV with some ks, and compare it to the result of our approximate formula. In principle, we should compare our approximate result with that of the LOO CV ($k = M$) because our formula approximates it. However for large M, the literal LOO CV requires huge computational burdens despite that the result is empirically not much different from that of the k-hold CV with moderate ks. Hence in some of the following experiments with large M, we use the 10-hold CV instead of the LOO CV. Further, to directly check the approximation accuracy, we also compute the normalized error difference defined as

$$\frac{\epsilon_{\mathrm{LOO}}^{\mathrm{approximate}} - \epsilon_{\mathrm{CV}}^{\mathrm{literal}}}{\epsilon_{\mathrm{CV}}^{\mathrm{literal}}}, \tag{45}$$

where $\epsilon_{\mathrm{CV}}^{\mathrm{literal}}$ denotes the literal CV estimator of the prediction error while $\epsilon_{\mathrm{LOO}}^{\mathrm{approximate}}$ is the approximated LOOE. Moreover, as a reference, we compute the negative log-likelihood of

Algorithm 2 Self-averaging approximate CV of the MLR

```
1: procedure SAACV($\hat{\boldsymbol{W}}(\lambda_1, \lambda_2), D^M, \lambda_2$)
2:     Compute the active sets $\{\hat{A}_i\}_{i=1}^N$ from $\hat{\boldsymbol{W}}$
3:     Compute $\{\boldsymbol{u}_\mu, X^\mu, \boldsymbol{b}_\mu, F^\mu\}_\mu$ by eqs. (1,9), (14) and (15)
4:     $t \leftarrow 0$                                         ▷ Start initialization
5:     for $i = 1, \cdots, N$ do
6:         $\left(\chi_i^{\backslash\mu}\right)^{(t)} \leftarrow 0,$
7:         $\left(\chi_i^{\backslash\mu}\right)^{(t)}_{\hat{A}_i\hat{A}_i} \leftarrow \sigma_x^{-2},$
8:     end for
9:     $\Delta \leftarrow 100$                                  ▷ End initialization
10:    while $\Delta > \theta$ do                                ▷ Compute $C_{\mathrm{SA}}$ by recursion
11:        $C_{\mathrm{SA}}^{(t+1)} \leftarrow \sigma_x^2 \sum_{i=1}^N \left(\chi_i^{\backslash\mu}\right)^{(t)}$
12:        $R \leftarrow \sigma_x^2 \sum_{\mu=1}^M \left(I_L + F^\mu C_{\mathrm{SA}}^{(t+1)}\right)^{-1} F^\mu + \lambda_2 I_L$       ▷ $O(ML^3)$
13:        $\Delta \leftarrow 0$
14:        for $i = 1, \cdots, N$ do                             ▷ $O(NL^3)$
15:            if $\lambda_2$ is large enough
16:                $\overline{R}^{-1}_{\hat{A}_i\hat{A}_i} = \left(R_{\hat{A}_i\hat{A}_i}\right)^{-1}$
17:            else
18:                Compute $\overline{R}^{-1}_{\hat{A}_i\hat{A}_i}$ by eq. (44) from $R$
19:            end if then
20:            $\left(\chi_i^{\backslash\mu}\right)^{(t+1)}_{\hat{A}_i\hat{A}_i} \leftarrow \overline{R}^{-1}_{\hat{A}_i\hat{A}_i}$
21:            $\Delta \leftarrow \Delta + \left|\left|\left(\chi_i^{\backslash\mu}\right)^{(t+1)}_{\hat{A}_i\hat{A}_i} - \left(\chi_i^{\backslash\mu}\right)^{(t)}_{\hat{A}_i\hat{A}_i}\right|\right|_{\mathrm{F}}$
22:        end for
23:        $\Delta \leftarrow \Delta / N$
24:        $t \leftarrow t + 1$
25:    end while
26:    for $\mu = 1, \cdots, M$ do
27:        $\boldsymbol{u}_\mu^{\backslash\mu} \leftarrow \boldsymbol{u}_\mu + C_{\mathrm{SA}}^{(t)} \boldsymbol{b}^\mu$
28:    end for
29:    Compute $\epsilon_{\mathrm{LOO}}$ from $\{\boldsymbol{u}_\mu^{\backslash\mu}\}_\mu$ by eq. (8)
30:    return $\epsilon_{\mathrm{LOO}}$
31: end procedure
```

the full solution $\{\hat{\boldsymbol{w}}_a\}_{a=1}^L$ as:

$$\epsilon = \frac{1}{M}\sum_{\mu=1}^{M} q_\mu\left(\{\hat{\boldsymbol{w}}_a\}_{a=1}^L\right), \tag{46}$$

and call it the training error, hereafter. The training error is expected to be a monotonic increasing function with respect to λ, while the prediction one is supposed to be non-monotonic.

In all of the experiments, we used a single CPU of Intel(R) Xeon(R) E5-2630 v3 2.4GHz. To solve the optimization problems in eqs. (4,6), we employed *Glmnet* (Friedman et al., 2010) which is implemented as a *MEX* subroutine in MATLAB®. The two approximations were implemented as raw codes in MATLAB. This is not the most optimized approach, because as seen in Algs. 1,2 our approximate formula uses a number of *for* and *while* loops which are slow in MATLAB, and hence the comparison is not necessarily fair. However, even in this comparison there is a significant difference in the computational time between the literal CV and our approximations, as shown below.

In Glmnet, the corresponding optimization problem is parameterized as follows:

$$\{\hat{\boldsymbol{w}}_a(\tilde{\lambda},\eta)\}_a = \underset{\{\boldsymbol{w}_a\}_a}{\arg\min}\left\{\frac{1}{M}\sum_{\mu=1}^{M} q_\mu\left(\{\boldsymbol{w}_a\}_{a=1}^L\right) + \tilde{\lambda}\left(\eta\sum_{a=1}^{L}||\boldsymbol{w}_a||_1 + \frac{(1-\eta)}{2}\sum_{a=1}^{L}||\boldsymbol{w}_a||_2^2\right)\right\}. \tag{47}$$

In the following experiments, we present the results based on this parameterization. We basically prefer $\eta = 1$ in which the ℓ_2 term is absent, because the main contribution of the present paper is to overcome technical difficulties stemming from the ℓ_1 term. However, Glmnet or its employing algorithm occasionally loses its stability in some uncontrolled manner without the ℓ_2 term. Hence, in the following experiments we adaptively choose the value of η.[1]

A sensitive point which should be noted is the convergence problem of the algorithm for solving the present optimization problem. In Glmnet, a specialized version of coordinate descent methods is employed, and it requires a threshold δ to judge the algorithm convergence. Unless explicitly mentioned, we set this as $\delta = 10^{-8}$ being tighter than the default value. This is necessary since we treat problems of rather large sizes. A looser choice for δ rather strongly affects the literal CV result, while it does not change the full solution or the training error as much. As a result, our approximations employing only the full solution are rather robust against the choice of δ compared to the literal CV. This is also demonstrated below.

3.1. On simulated dataset

Let us start by testing with the simulated data. Suppose each "true" feature vector $\boldsymbol{w}_{0a}$ is independently identically drawn (i.i.d.) from the following Bernoulli-Gaussian prior:

$$\boldsymbol{w}_{0a} \sim \prod_{i=1}^{N}\left\{(1-\rho_0)\delta(w_{0ai}) + \rho_0\mathcal{N}(0,1/\rho_0)\right\}, \tag{48}$$

1. When employing our distributed codes implementing the approximate formula (Obuchi, 2017; Takahashi and Obuchi, 2017) in conjunction with Glmnet, the parameters λ_1 and λ_2 are read as $\lambda_1 = M\tilde{\lambda}\eta$ and $\lambda_2 = M\tilde{\lambda}(1-\eta)$.

where $\mathcal{N}(\mu, \sigma^2)$ denotes a Gaussian distribution whose mean and variance are μ and σ^2, respectively. The resultant feature vector $\boldsymbol{v}_a$ becomes $N\rho_0(\equiv K_0)$-sparse and its norm becomes $\sqrt{N}$ on average. Then, we choose a class y_μ from $\{1, \cdots, L\}$ uniformly and randomly, and generate an observed feature vector $\boldsymbol{x}_\mu$ by leveraging the following linear process:

$$\boldsymbol{x}_\mu = \frac{\boldsymbol{w}_{0y_\mu}}{\sqrt{N}} + \boldsymbol{\xi}, \tag{49}$$

where $\boldsymbol{\xi}$ is an observation noise each component of which is i.i.d. from a Gaussian $\mathcal{N}(0, \sigma_N^2)$.

For convenience, we introduce the ratio of the data size to the feature dimensionality, $\alpha = M/N$, and now obtain five parameters $\{N, L, \alpha, \rho_0, \sigma_\xi^2\}$ characterizing the experimental setup. It is rather heavy to obtain the dependence of all parameters and below, and hence we mainly focus on the dependence on L, σ_ξ^2, and N. Other parameters are set to be $\alpha = 2$ and $\rho_0 = 0.5$.

3.1.1. Result

Let us summarize the result on simulated data.

Fig. 1 shows the plots of the prediction and training errors against $\tilde{\lambda}$ for $L = 4, 8, 16$ at $N = 200$ and $\sigma_\xi^2 = 0.01$. This demonstrates that both approximations provide consistent results with the literal LOO CV, except at small $\tilde{\lambda}$s. This inconsistency at small $\tilde{\lambda}$s is considered to be due to a numerical instability occurring in the literal CV. Actually, for small $\tilde{\lambda}$s, we have observed that certain small changes in the data induce large differences in the literal CV result. This example demonstrates that our approximations provide robust curves even in such situations. Note that as L grows the number of estimated parameters $\{\boldsymbol{w}_a\}_{a=1}^L$ increases while the data size $M = \alpha N = 400$ is fixed, meaning that the problem becomes more and more underdetermined with the growth of L. Hence, Fig. 1 demonstrates that the developed approximations work irrespectively of how much the problem is underdetermined.

Fig. 2 exhibits the σ_ξ^2-dependence of the errors and the approximation results for $L = 8$ and $N = 200$. For the very weak noise case ($\sigma_\xi^2 = 0.001$, left), the difference between the predictive and training errors is negligible and hence all four curves are not discriminable. For the moderate ($\sigma_\xi^2 = 0.1$, middle) and large ($\sigma_\xi^2 = 1$, right) noise cases, the training curve is very different from the predictive ones. The approximation curves are again consistent with the literal LOO one.

Fig. 3 demonstrates how the approximation accuracy changes as the system size N grows. For small sizes $N = 50, 100$, a discriminable difference exists between the results of the approximations and the literal LOO CV, as well as the difference between the results of the two approximations. This is expected, because our derivation relies on the largeness of N and M. For large systems $N = 400, 800$, the difference among the two approximations and the literal CV is much smaller. Considering this example in conjunction with the middle panel of Fig. 1, we can recognize that our approximate formula becomes fairly precise for $N \geq 200$ in this parameter set. The normalized error difference corresponding to Fig. 3 is shown in Fig. 4. We can observe that the difference tends to be smaller as the system size increases, which is expected because the perturbation employed in our approximate formula is justified in the large N, M limit.

Finally, let us consider the actual computational time to evaluate $\{\hat{\boldsymbol{w}}_a\}_a$ and the approximate LOOEs, and observe its system size dependence. The left panel of Fig. 5 provides

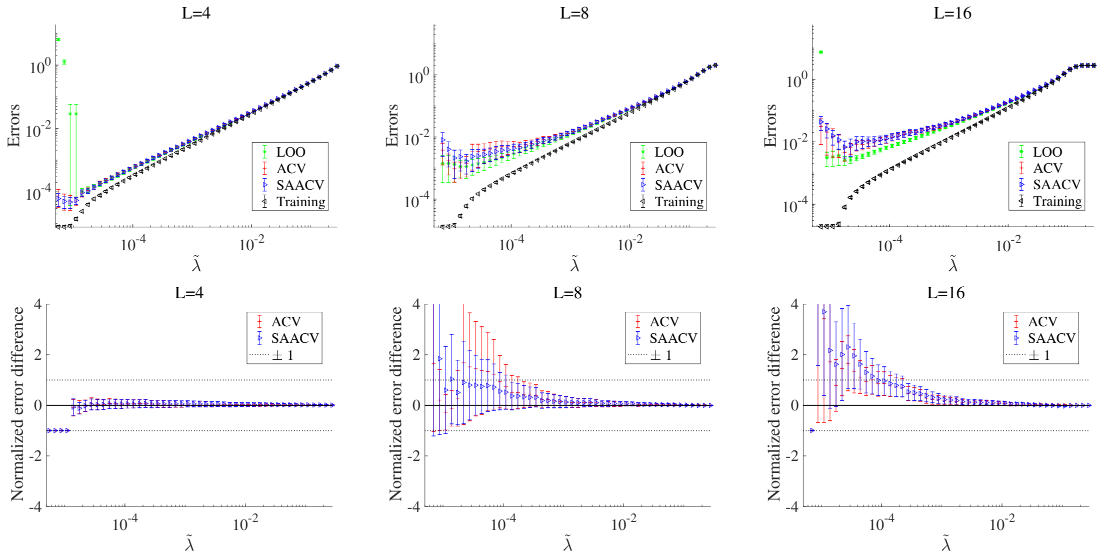

Figure 1: (Upper) Log-log plots of the errors against $\tilde{\lambda}$ for several values of the class number L. Other parameters are fixed at $N = 200$, $\sigma_\xi^2 = 0.01$, $\alpha = 2$ and $\rho_0 = 0.5$. The approximation results are consistent with the literal LOO CV results, except at small λs, which is presumably due to a numerical instability occurring in the literal CV at small λs. Here, $\eta = 0.9$. (Lower) The normalized error difference (45) plotted against $\tilde{\lambda}$. The parameters of each panel are them of the corresponding upper one. The horizontal dotted lines denote ± 1 and drawn for comparison. For small $\tilde{\lambda}$s the difference is not negligible, but the literal CV itself is not stable in that region and hence the error difference is not reliable.

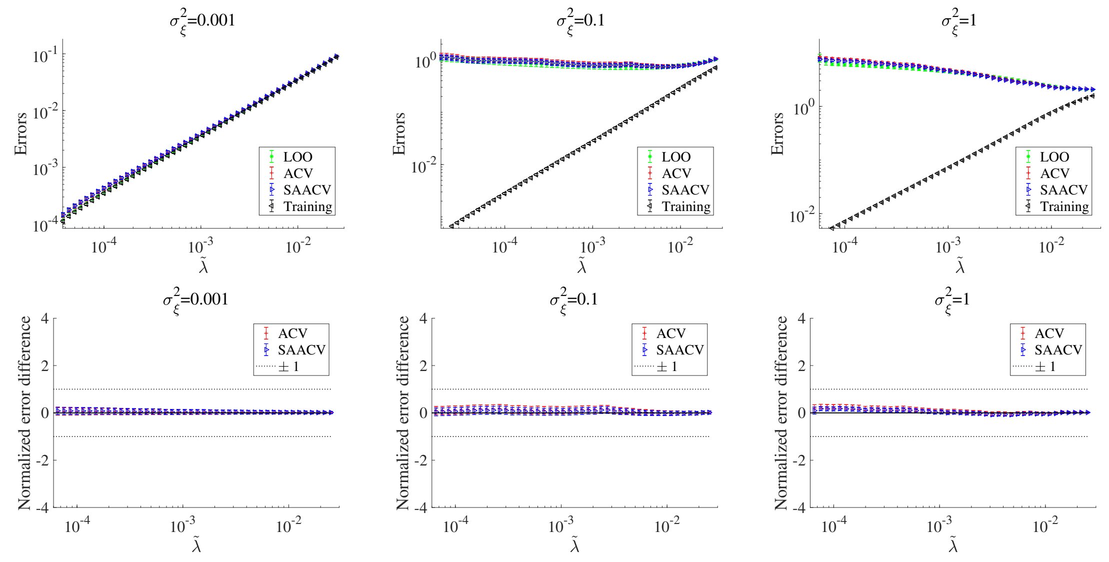

Figure 2: (Upper) Log-log plots of the errors against $\tilde{\lambda}$ for several noise strengths. Other parameters are fixed at $N = 200$, $L = 8$, $\alpha = 2$ and $\rho_0 = 0.5$. The approximation results are consistent with the literal LOO CV, irrespectively of the noise strength. The convergence threshold δ is set to be $\delta = 10^{-9}$ for the case $\sigma_\xi^2 = 1$. Here, $\eta = 1$. (Lower) The normalized error difference (45) plotted against $\tilde{\lambda}$. The parameters of each panel are them of the corresponding upper one. In the whole region the difference is negligibly small.

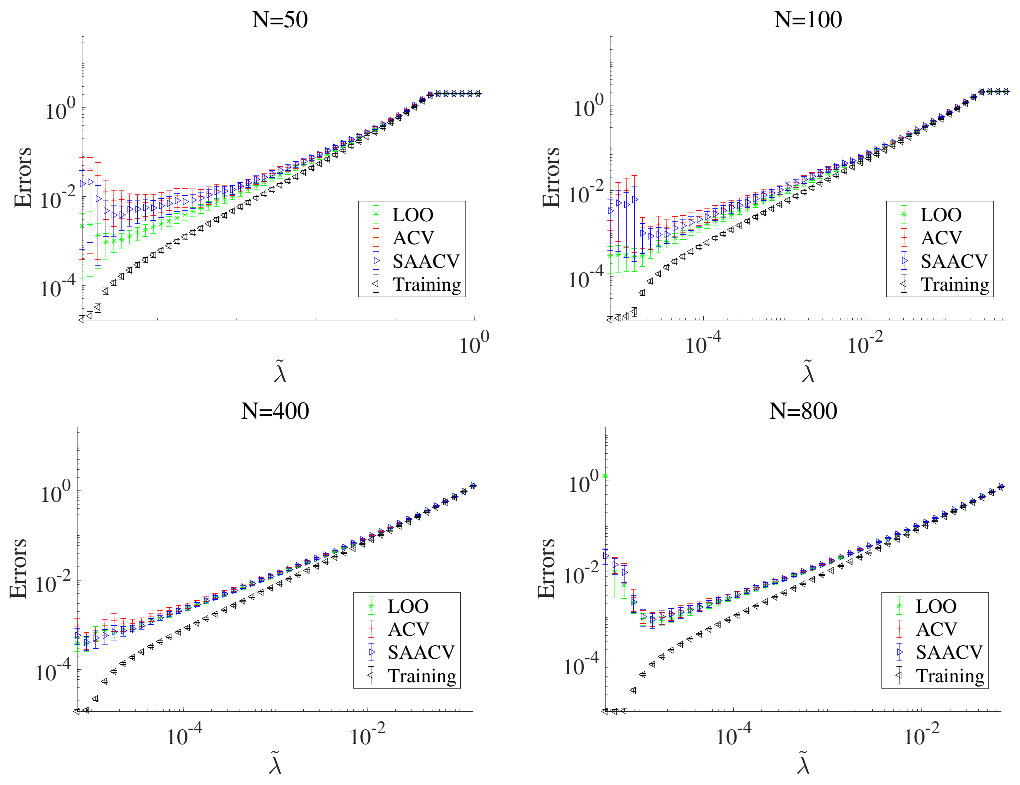

Figure 3: Log-log plots of the errors against $\tilde{\lambda}$ for several values of feature dimensionality N. Other parameters are fixed at $L = 8$, $\sigma_\xi^2 = 0.01$, $\alpha = 2$ and $\rho_0 = 0.5$. Here, $\eta = 0.9$.

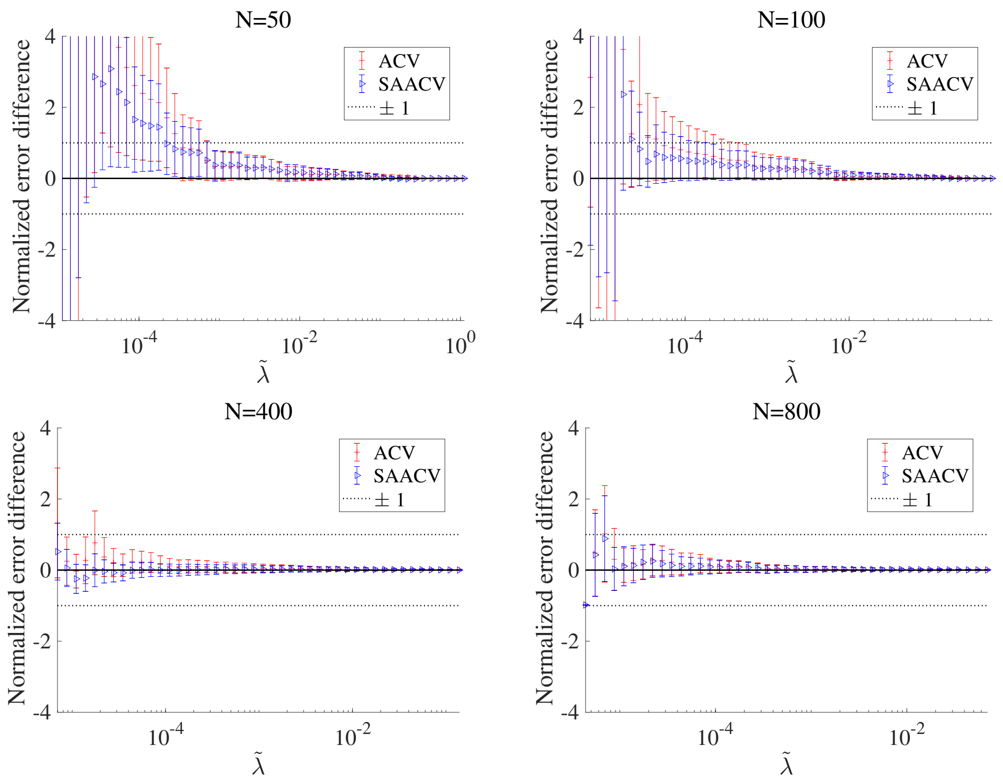

Figure 4: The plot of the normalized error difference corresponding to Fig. 3. The difference tends to be smaller as the system size increases.

the plot of the actual computational time against the system size. Here, the number of examined points of $\tilde{\lambda}$ to obtain a solution path is different from size to size, and hence the plotted time is given as the whole computational time to obtain the solution path divided by the number of $\tilde{\lambda}$s points. The left panel of Fig. 5 clearly displays the advantage and

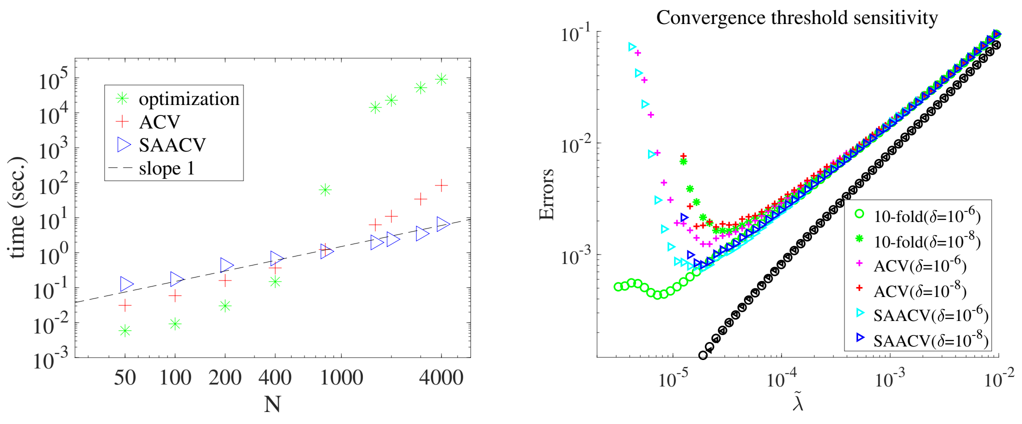

Figure 5: (Left) Actual computational time spent to find the solution of eq. (4) and that for ACV and SAACV, plotted against the feature dimensionality N in a double logarithmic scale. Note that the computational time for the k-fold CV is about k times larger than that for finding the solution of eq. (4), represented by the green asterisks. Parameters are fixed at $L = 8$, $\sigma_\xi^2 = 0.01$, $\alpha = 2$ and $\rho_0 = 0.5$. Here, $\eta = 1$. (Right) The errors are obtained for the two convergence thresholds $\delta = 10^{-6}$ and $\delta = 10^{-8}$. Error bars are omitted for visibility. For the tighter case $\delta = 10^{-8}$, the minimum value of $\tilde{\lambda}$ in the examined range is larger than that of the case $\delta = 10^{-6}$, though the systematic difference with the results of the literal LOO CV is already clear. The training errors of these two different δ, represented by black circles and left-pointing triangles, are completely overlapping. The system parameters are $N = 400$, $L = 8$, $\sigma_\xi^2 = 0.01$, $\alpha = 2$ and $\rho_0 = 0.5$. Here, $\eta = 1$.

disadvantage of the developed approximations. For small sizes, the computational time for optimization to obtain $\{\hat{\boldsymbol{w}}_a\}_a$ is shorter than the time to compute the approximate LOOEs, and hence the literal CV is better. However, for larger systems, the optimization cost increases rapidly and for $N \gtrsim 400$ the approximate CV is better. For $N \gtrsim 800$, the ACV cost exceeds that of SAACV. The SAACV cost behaves linearly as a function of N (see the black dashed line), and hence for larger systems of $N \gtrsim 800$ SAACV can be a very powerful tool. As a related issue, we mention the convergence problem of the algorithm. In the right panel of Fig. 5, we compare the errors at two different values of the convergence threshold δ. An important observation is that a significant difference exists in the literal CV results while other curves do not show a strong change. This implies that our approximate formula is rather robust and can be used with a rather loose convergence threshold or conversely, we can use the systematic deviation between the literal CV and our approximations as an indicator to verify the tightness of the convergence threshold. This is beneficial, especially when treating large models, for which the convergence check is a common annoying task.

3.2. On real-world dataset

Next, we test the approximate formula on a real-world dataset. As shown above, our approximations become more precise if the model dimensionality and data size are large. Hence, we chose the ISOLET dataset which is a relatively large problem among classification tasks collected in the UCI machine learning repository (Lichman, 2013). The feature dimensionality, the data size, and the class number are $N = 617$, $M = 6238$, and $L = 26$, respectively. Here we apply the 10-fold CV, instead of the LOO CV because of the computational reason, and our approximations to this dataset. The result is given in Fig. 6. The

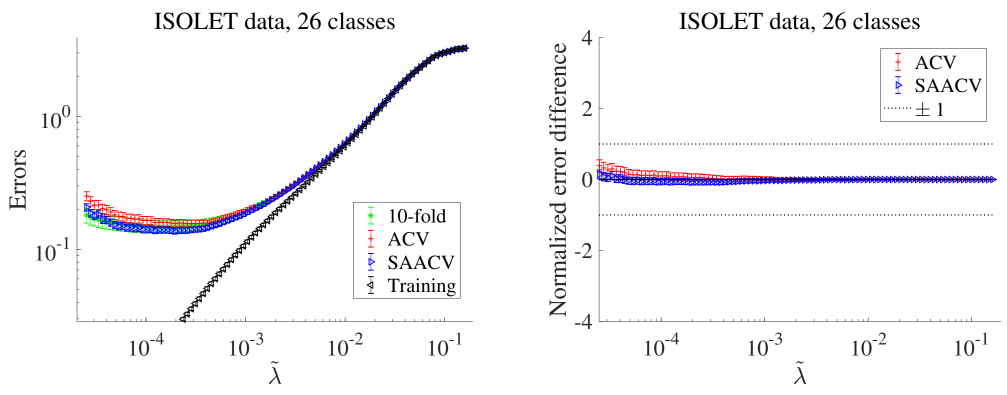

Figure 6: Approximate CV performance on the ISOLET data of $L = 26$ classes. The errors are shown in the left panel and the normalized error differences between the approximations and the 10-fold CV are in the right panel. At the estimated minimums of the prediction error, the accuracy rate for correctly classifying the test data is about 0.86 while the probability of recovering the training data is about 0.98, commonly among the literal CV and the two approximations. At the minimum value of $\tilde{\lambda}$, the leftmost point in the figure, the accuracy rates are different among the three different methods, and are $0.83, 0.78$, and 0.81 for the literal CV, ACV and SAACV, respectively. Here, $\eta = 1$.

results of the approximations and of the 10-fold CV demonstrate a fairly good agreement, proving the actual effectiveness of the developed approximations. In an experiment, the actual computational time to obtain the result of the full simulation, of the 10-fold CV, of ACV, and of SAACV were 785, 7825, 5173, and 689 seconds, respectively. The system parameters $\{N, M, L\}$ are rather large in this problem and thus the advantage of ACV is not large, while the efficiency of SAACV stands out in such situations.

3.3. When does SAACV fail?

Two major factors neglected in SAACV are the correlations among feature components and the heterogeneity among feature vectors. If these factors are strong, the approximation accuracy of SAACV is expected to be degraded. In this section, we examine this point.

First, to test the impact of correlations in feature components, we add further constraints to the simulated data treated in sec. 3.1 and examine the approximation performance on the situation. Two cases are treated: the first is the case where the true feature vectors $\{\boldsymbol{w}_{0a}\}$ have common components among all the classes. The result of this case is shown in the left panels in Fig. 7. Here, the fraction of the common components to the non-zero components is $r_{\rm common} = 0.9$ and thus the overlap between feature vectors of different classes is rather large. The other is the case where the noise vector has strong correlations among the components. The result of this case is presented in the right panels in Fig. 7, in which the noise strength is $\sigma_\xi^2 = 1$ and the correlation coefficient of any pair of noise components is $\mathrm{Corr}(\xi_i, \xi_j) = 0.9$; hence the noise and the correlation are rather large. For both the cases, the performance of the approximate formula is fairly good, implying that SAACV is likely to perform well even when components of the feature vectors are correlated. Similar findings were actually obtained in the case of linear models (Obuchi and Kabashima, 2016). This is a preferable observation because it implies that the applicable limit of SAACV can be extended to a wider class of feature vectors than that is assumed in our present derivation in which the weakness of the correlations is assumed, as seen in sec. A. These also imply that there possibly exists another approximation formula taking into account the correlations but being similar to SAACV. A promising framework to derive such a formula might be the adaptive TAP method (Opper and Winther, 2001a,b, 2005). The adaptive TAP method itself requires a larger computational cost than that of SAACV but it is possible to reduce the computational cost up to the linear scaling with respect to N and M by employing an additional simplifying approximation (Kabashima and Vehkaperä, 2014; Çakmak and Opper, 2018). This is, however, rather technical and we leave it as a future work.

Second, to examine the effect of the heterogeneity among feature vectors, we introduce an amplifying factor Ω to control the norm of feature vectors. In particular, we multiply the factor Ω to the feature vectors of some chosen classes, as $\boldsymbol{x}_{\boldsymbol{\mu}} \to \Omega \boldsymbol{x}_{\boldsymbol{\mu}}$. Here, we use the simulated data identical to that for the center panels in Fig. 3 of the parameters $(N, L, \alpha, \rho_0, \sigma_\xi^2, \eta) = (200, 8, 2, 0.5, 0.1, 1)$, except that the amplifying factor $\Omega = 100$ is applied to the latter four classes $y = 5, 6, 7, 8$. The approximation performance on this dataset is shown in Fig. 8. We also examine the impact of the same heterogeneity on a real-world dataset in Fig. 9. Here, we treat the well-known MNIST data of handwritten digits (LeCun et al., 1998). For simplicity, we only use the data of two digits: 0 and 1. As a preprocessing, feature components with small variances are removed and only the $N = 350$ components of the largest variances are retained; the original size of the feature vector is $784 = 28 \times 28$ and thus almost the half of the components are discarded. Then, the usual standardization procedure is conducted. Further, we apply the amplifying factor $\Omega = 10$ to the class of 1 (right panels), while the case without the amplification (or $\Omega = 1$, left panels) is also examined for comparison. These two examples clearly show that ACV shows a consistency with the LOO CV behavior while SAACV does not, demonstrating that SAACV gives an inaccurate estimate of the CV error for datasets with strong heterogeneity. This kind of heterogeneity

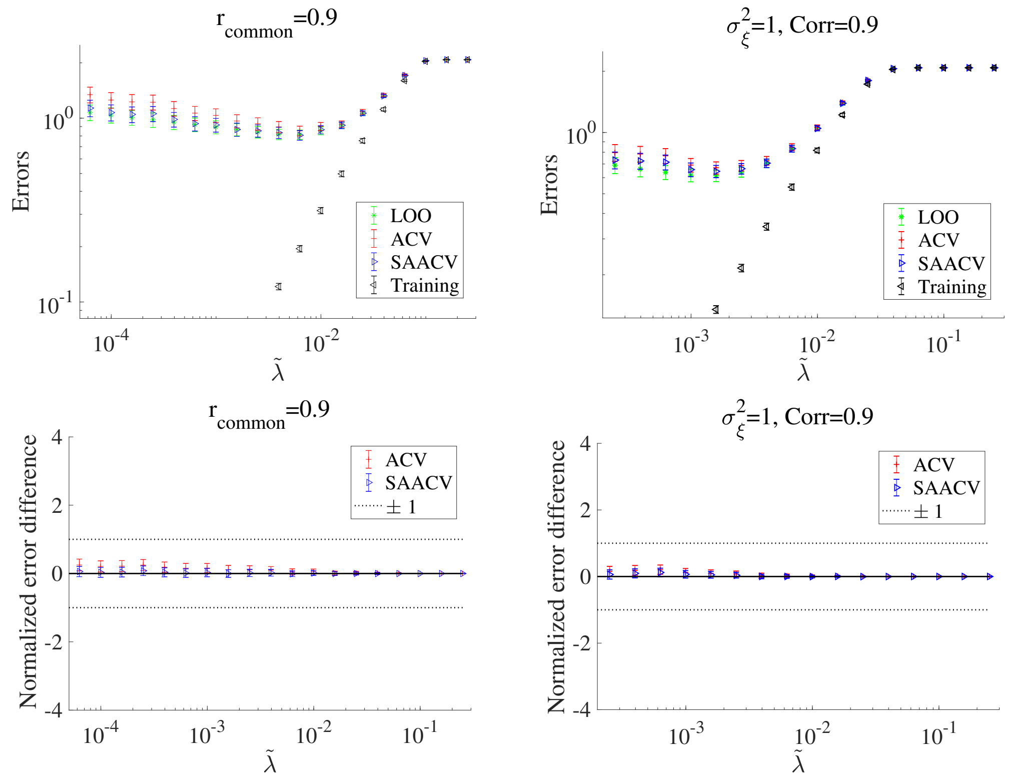

Figure 7: (Upper) Log-log plots of the errors against $\tilde{\lambda}$ for correlated feature vectors. The left panel is for the case with common components in true feature vectors while the right one is of the correlated noise case. Parameters $(N, L, \alpha, \rho_0, \eta) = (200, 8, 2, 0.5, 1)$ are common in both the cases, while the noise strengths and convergence thresholds are different: $(\sigma_\xi^2, \delta) = (0.1, 10^{-8})$ (left) and $(\sigma_\xi^2, \delta) = (1, 10^{-9})$ (right). (Lower) The normalized error difference (45) plotted against $\tilde{\lambda}$. The parameters of each panel are them of the corresponding upper one.

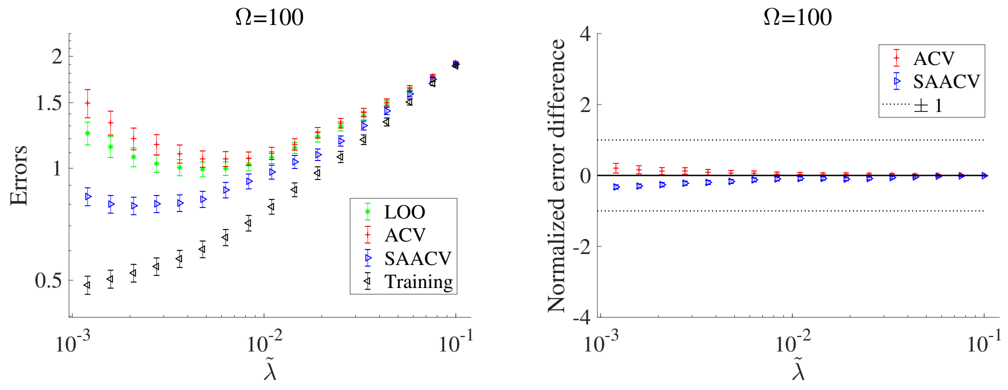

Figure 8: (Left) Log-log plots of the errors against $\tilde{\lambda}$ with strong heterogeneity in feature vectors. The same dataset as that of the center panels in Fig. 3 is used but the feature vectors for the classes $y_\mu = 5, 6, 7, 8$ are amplified as $\boldsymbol{x}_{\boldsymbol{\mu}} \to \Omega \boldsymbol{x}_{\boldsymbol{\mu}}$ by the factor $\Omega = 100$. The ACV result is consistent with the LOO CV one while that of SAACV is not. (Right) The normalized error difference corresponding to the left panel.

can naturally emerge in some applications: for example if we consider problems in medical statistics, a number of biological markers can give distinguishably large values for affected patients compared to unaffected ones, yielding larger values in norm for feature vectors of affected patients. This consideration suspects the efficiency of SAACV. We, however, stress that this kind of heterogeneity attributed to the belonging class can be absorbed by rescaling the weights as $\{\boldsymbol{w}_a\}_a \to \{\Omega_a^{-1}\boldsymbol{w}_a\}_a$, where Ω_a is chosen to homogenize the feature vector norm in different classes as $||\boldsymbol{x}_{y_\mu}\Omega_a||_2 \approx$ const. For the ℓ_1 regularization case, this resultantly leads to the regularization coefficients which take different values adaptively to the belonging class as

$$\lambda \sum_a ||\boldsymbol{w}_a||_1 \to \sum_a \lambda\Omega_a ||\boldsymbol{w}_a||_1 = \sum_a \lambda_a ||\boldsymbol{w}_a||_1. \tag{50}$$

For this problem with adaptive regularization coefficients, our approximation formula can be applied in the completely same manner, which can be convinced by seeing Algs. 1,2 where the value of the regularization coefficient is not required as the argument. The ℓ_2-norm can also be handled, though the codes should be extended to take into account the groupwise coefficients as arguments. We argue that this rescaling is a natural prescription to treat strong heterogeneity among different classes, and once employing this prescription the weak point of SAACV is naturally cured.

As a noteworthy remark, we point out that the basic idea of SAACV is closely related to Wahba's generalized cross-validation (GCV) for linear regression (Golub et al., 1979). In GCV, the heterogeneity in coefficient corresponding to $C_\mu^{\setminus \mu}$ in SAACV is also neglected, and hence it shares the same weak point as SAACV, when it is regarded as an approximation of the CV estimator to generalization errors. We stress that this kind of approximation

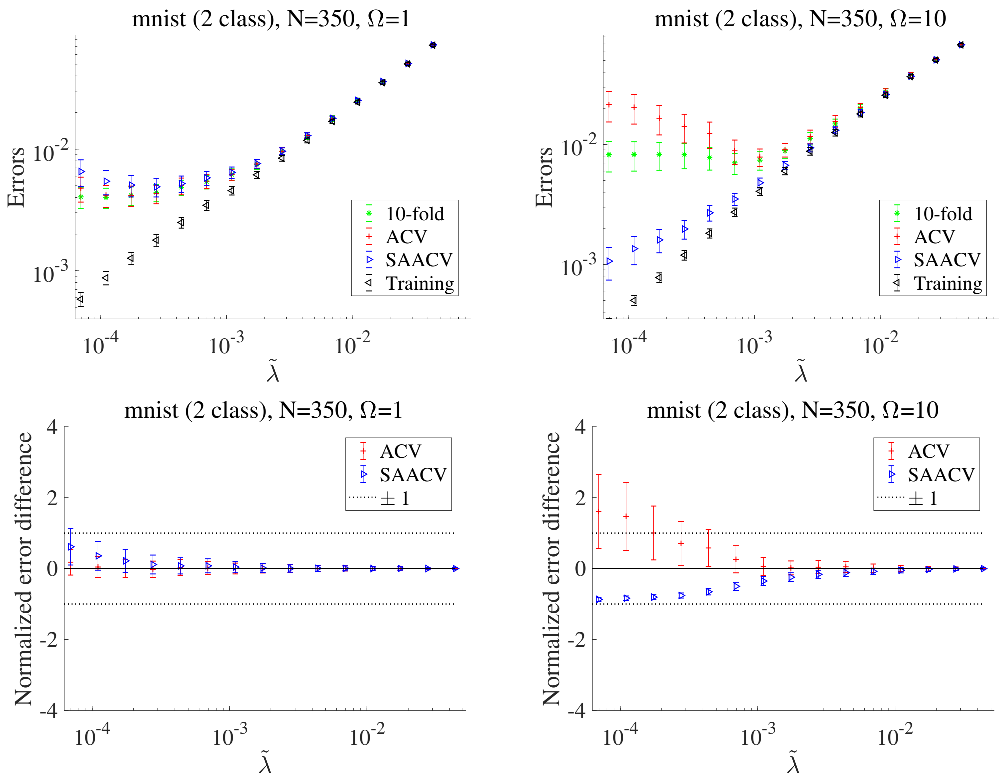

Figure 9: (Upper) Log-log plots of the errors against $\tilde{\lambda}$ of mnist handwritten data with two digits 0 and 1. The amplifying factor is not applied (or $\Omega = 1$) in the left panels while is applied ($\Omega = 10$) in the right ones. The strong heterogeneity among the classes affect the performance of SAACV. (Lower) The normalized error difference corresponding to the upper panels.

reducing the computational cost is again needed because the data size and the model dimensionality are increasing rapidly in recent years.

4. Conclusion

In this paper, we have developed an approximate formula for the CV estimator of the predictive likelihood of the multinomial logistic regression regularized by the ℓ_1-norm. An extension to the elastic net regularization has been also stated. We have demonstrated their advantages and disadvantages in numerical experiments using simulated and real-world datasets. Two versions of the approximation have been defined based on the developed formula. The first version, abbreviated as ACV, has a better performance, in terms of computational time, for middle size problems. It will eventually become worse than the literal k-fold CV with moderate ks as the problem size grows, because its computational time is scaled as a third-order polynomial of the feature dimensionality and data size, N and M, though such a tendency has not been observed in the investigated range of N. We have also defined the second version based on ACV, the computational time of which is just scaled linearly with respect to N and M. This second approximation is called SAACV, and it has been demonstrated that SAACV is slow for small size problems but has a great advantage for large size problems. Hence, we suggest leveraging the literal CV for small, ACV for middle, and SAACV for large size problems.

Our derivation is based on the perturbation which assumes that there is a small difference between the full and leave-one-out solutions. This assumption will not be satisfied for some specific cases. Even with this restriction, we expect the range of application of our formula is wide enough and we would like to encourage readers to leverage it in their own work. We have implemented MATLAB and python codes and they are available in (Obuchi, 2017; Takahashi and Obuchi, 2017).

The perturbative approach employed here is fairly general and can be applied to a wide class of generalized linear models with convex regularizations. The development of practical formulas for these cases will be of great assistance, given that we are living in the Big Data era.

Acknowledgments

This work was supported by JSPS KAKENHI Nos. 18K11463 (TO), 25120013 and 17H00764 (YK). TO is also supported by a Grant for Basic Science Research Projects from the Sumitomo Foundation. The authors are grateful to Takashi Takahashi for implementation of the approximation formula in python.

Appendix A. The SA approximation

Let us derive eq. (41) in the SA approximation. We work on a framework called a cavity method in statistical physics or belief propagation (BP) in computer science. We start from defining the so-called Boltzmann distribution:

$$P\left(\{\boldsymbol{w}_a\}_{a=1}^L \middle| D^M, \lambda\right) = \frac{1}{Z\left(D^M, \lambda\right)} e^{-\beta \mathcal{H}\left(\{\boldsymbol{w}_a\}_{a=1}^L \middle| D^M, \lambda\right)}$$
$$= \frac{e^{-\beta \sum_a \left(\lambda_1 ||\boldsymbol{w}_a||_1 + \frac{\lambda_2}{2} ||\boldsymbol{w}_a||_2^2\right)}}{Z\left(D^M, \lambda\right)} \prod_{\mu=1}^{M} \phi^\beta \left(y_\mu | \{u_{\mu a}\}_a\right). \tag{51}$$

In the $\beta \to \infty$ limit, this distribution converges to a point-wise measure of the solution of eq. (4) and hence, it is useful for analyzing eq. (51). We note that the BP is usually applied to graphical models having sparse tree-like structures, but is also applicable to ones with densely connected structures. In such applications, the BP can be regarded as a systematic implementation of the Thouless-Anderson-Palmer (TAP) approach (Thouless et al., 1977) in statistical physics, which can yield a set of self-consistent equations of the first and second moments of variables in statistical models when the models are of densely connected types. This approach has been applied to many different models in machine learning, which continuously yields evidences of its effectiveness (Opper and Winther, 1996, 1997, 2000a,b). When applied to densely connected models, certain correlations between variables have to be neglected to make the computation tractable; for that reason this approach, or the associated algorithm derived from it, is recently called approximate message passing (AMP) (Kabashima, 2003; Donoho et al., 2009). Basically, the AMP assumes that "interactions" between the variables are weak: in the present problem this implies $(1/M)\sum_{\mu=1}^M x_{\mu_i} x_{\mu_j} - \left((1/M)\sum_{\mu=1}^M x_{\mu_i}\right)\left((1/M)\sum_{\mu=1}^M x_{\mu_j}\right) \approx 0$ $(i \neq j)$. This treatment can be justified if each feature vector $\boldsymbol{x}_\mu$ is i.i.d.. Rigorous proofs of this fact are available for linear models and their some variants (Bayati and Montanari, 2011; Barbier et al., 2017). We implicitly assume this in the following derivation.

In this appendix, we introduce a new vector representation summarizing class variables: $\boldsymbol{w}_i = (w_{ai})_a$. Note that this is different from the notation used in the main body of this paper, $\boldsymbol{w}_a = (w_{ai})_i$ in which the feature components are summarized.

By regarding $\boldsymbol{w}_i$ as a single variable node, the BP decomposes eq. (51) into two types of messages as follows:

$$\tilde{M}_{\mu \to i}(\boldsymbol{w}_i) = \int \prod_{j(\neq i)} d\boldsymbol{w}_j \; \phi^\beta(\boldsymbol{u}_\mu) \prod_{j(\neq i)} M_{j \to \mu}(\boldsymbol{w}_j), \tag{52}$$

$$M_{i \to \mu}(\boldsymbol{w}_i) = e^{-\beta\left(\lambda_1 ||\boldsymbol{w}_i||_1 + \frac{\lambda_2}{2}||\boldsymbol{w}_i||_2^2\right)} \prod_{\nu(\neq \mu)} \tilde{M}_{\nu \to i}(\boldsymbol{w}_i), \tag{53}$$

where $\boldsymbol{u}_\mu = (u_{\mu a})_a$. A crucial observation to assess eqs. (52,53) is that the argument of the potential function $\phi(\boldsymbol{u}_\mu)$ has a sum of an extensive number of random variables; the central limit theorem thus justifies treating it as a Gaussian variable with the appropriate mean and variance. Hence, according to eq. (52) where $\boldsymbol{w}_i$ is special, we can divide the extensive sum as follows:

$$u_{\mu a} = \sum_j x_{\mu j} w_{aj} = x_{\mu i} w_{ai} + \sum_{j(\neq i)} x_{\mu j} w_{aj} \approx x_{\mu i} w_{ai} + \sum_{j(\neq i)} x_{\mu j} \left\langle w_{aj} \right\rangle^{\backslash \mu} + t_a, \tag{54}$$

where the second term on the right-hand side represents the mean of $\sum_{j(\neq i)} x_{\mu j} w_{aj}$, the symbol $\langle\cdot\rangle^{\setminus\mu}$ denotes the average over the Boltzmann distribution without the μth potential function, and $\boldsymbol{t}^{\mu} = (t_a^{\mu})_a$ denotes the zero-mean Gaussian variables whose covariance is set to be that of $\left(\sum_{j(\neq i)} x_{\mu j} w_{aj}\right)_a$. This expression allows us to replace the integration $\int \prod_{j(\neq i)} d\boldsymbol{w}_j$ by that over $\boldsymbol{t}^{\mu}$ in eq. (52). This significantly simplifies the computation and yields:

$$\tilde{M}_{\mu\to i}(\boldsymbol{w}_i) \approx \int d\boldsymbol{t}\; e^{\beta\left(-\frac{1}{2}\boldsymbol{t}^{\top}(C_{\mu}^{\setminus\mu})^{-1}\boldsymbol{t} - q_{\mu}(\boldsymbol{w}_i,\boldsymbol{t})\right)} \equiv \int d\boldsymbol{t}\; e^{\beta f^{\mu}(\boldsymbol{w}_i,\boldsymbol{t})} \tag{55}$$

where $q_{\mu}(\boldsymbol{w}_i,\boldsymbol{t})$ is the negative log-likelihood whose argument $u_{\mu a}$ is approximated by eq. (54) and $C_{\mu}^{\setminus\mu}$ is the rescaled covariance of $\sum_{j(\neq i)} x_{\mu j} w_{aj}$ defined as

$$\begin{aligned} \chi_{(ai)(bj)}^{\setminus\mu} &\equiv \beta\left(\langle w_{ai} w_{bj}\rangle^{\setminus\mu} - \langle w_{ai}\rangle^{\setminus\mu}\langle w_{bj}\rangle^{\setminus\mu}\right), \\ \left(C_{\mu}^{\setminus\mu}\right)_{ab} &\equiv \sum_{i,j} x_{\mu i} x_{\mu j} \chi_{(ai)(bj)}^{\setminus\mu}. \end{aligned} \tag{56}$$

In the second equation we added the contribution from i for simplicity. It does not affect the following result because the ith term contribution is small enough. Let us focus on the limit $\beta\to\infty$. This limit allows us to use the saddle-point method, or Laplace's method, with respect to $\boldsymbol{t}^{\mu}$. The associated saddle-point equation is:

$$\hat{\boldsymbol{t}}^{\mu} = -C_{\mu}^{\setminus\mu}\boldsymbol{b}^{\mu}(\boldsymbol{w}_i,\hat{\boldsymbol{t}}^{\mu}), \tag{57}$$

where $\boldsymbol{b}^{\mu}(\boldsymbol{w}_i,\hat{\boldsymbol{t}}^{\mu})$ is the gradient of q_{μ} defined at eq. (14) but the argument $u_{\mu a}$ is approximated by eq. (54). Now, let us expand the exponent $f^{\mu}(\boldsymbol{w}_i,\boldsymbol{t})$ in eq. (55) with respect to the dynamical variables $\boldsymbol{w}_i$ up to the second order. Putting $z_a = \sum_i x_{\mu i} w_{ai}$, we can define the derivatives as:

$$\frac{\partial \hat{\boldsymbol{t}}^{\mu}}{\partial z_a} = -(I_L + C_{\mu}^{\setminus\mu}F^{\mu})^{-1} C_{\mu}^{\setminus\mu} F_{*a}^{\mu}, \tag{58}$$

$$\frac{\partial f^{\mu}(\boldsymbol{w}_i,\hat{\boldsymbol{t}}^{\mu})}{\partial z_a} = -b_a^{\mu}(\boldsymbol{w}_i,\hat{\boldsymbol{t}}^{\mu}), \tag{59}$$

$$\frac{\partial^2 f^{\mu}(\boldsymbol{w}_i,\hat{\boldsymbol{t}}^{\mu})}{\partial z_a \partial z_b} = -F_{ab}^{\mu} - \left(\frac{\partial \hat{\boldsymbol{t}}^{\mu}}{\partial z_a}\right)^{\top} F_{*b}^{\mu} = -\left(\left(I_L + F^{\mu}C_{\mu}^{\setminus\mu}\right)^{-1} F^{\mu}\right)_{ab}. \tag{60}$$

Hence,

$$\tilde{M}_{\mu\to i}(\boldsymbol{w}_i) \propto e^{\beta\left(\left(\boldsymbol{h}_i^{\mu}\right)^{\top}\boldsymbol{w}_i - \frac{1}{2}\boldsymbol{w}_i^{\top}\Gamma_i^{\mu}\boldsymbol{w}_i\right)} \tag{61}$$

where

$$\begin{aligned} \boldsymbol{h}_i^{\mu} &= -x_{\mu i}\boldsymbol{b}^{\mu}, \\ \Gamma_i^{\mu} &= x_{\mu i}^2 (I_L + F^{\mu}C_{\mu}^{\setminus\mu})^{-1} F^{\mu}. \end{aligned} \tag{62}$$

Note that this second order expansion is justified in the limit $\beta\to\infty$.

Collecting all the messages except for μ, we can construct the LOO marginal distribution of $\boldsymbol{w}_i$ as:

$$P^{\backslash\mu}(\boldsymbol{w}_i) \propto e^{-\beta\left(\lambda_1||\boldsymbol{w}_i||_1+\frac{\lambda_2}{2}||\boldsymbol{w}_i||_2^2\right)} \prod_{\nu(\neq\mu)} \tilde{M}_{\nu\to i}(\boldsymbol{w}_i)$$
$$\propto e^{\beta\left((\sum_{\nu(\neq\mu)}\boldsymbol{h}_i^{\mu})^\top \boldsymbol{w}_i - \frac{1}{2}\boldsymbol{w}_i^\top\left(\lambda_2 I_L+\sum_{\nu(\neq\mu)}\Gamma_i^{\mu}\right)\boldsymbol{w}_i-\lambda_1||\boldsymbol{w}_i||_1\right)}. \tag{63}$$

Now, we can close the equation for the rescaled variance $\left(\chi_i^{\backslash\mu}\right)_{ab} \equiv \chi_{(ai),(bi)}^{\backslash\mu}$, because we can compute the variance of $\boldsymbol{w}_i$ from eq. (63). By considering the scaling, we can recognize that the variances vanish in the speed of $O(\beta^{-2})$ if one of the two components or both are inactive. The active-active components of the variance are scaled by $O(\beta^{-1})$ and remain in the rescaled variance. Focusing on the limit $\beta\to\infty$, we thus obtain:

$$\left(\chi_i^{\backslash\mu}\right)_{\hat{A}_i\hat{A}_i} = \left(\lambda_2 I_{|\hat{A}_i|} + \left(\sum_{\nu(\neq\mu)}\Gamma_i^{\nu}\right)_{\hat{A}_i\hat{A}_i}\right)^{-1} \approx \left(\lambda_2 I_{|\hat{A}_i|} + \left(\sum_{\nu}\Gamma_i^{\nu}\right)_{\hat{A}_i\hat{A}_i}\right)^{-1}. \tag{64}$$

At the last step, the μth term is added since its contribution is expected to be small enough in the summation. This manifests that the μ-dependence of $\chi^{\backslash\mu}$ can be neglected and we rewrite it as $\chi^{\backslash\mu}=\chi$ hereafter. By considering the meaning of the Hessian, it is easy to understand that $G^{\backslash\mu}$ is identified with $\left(\lambda_2 I_L + \sum_{\nu(\neq\mu)}\Gamma_i^{\nu}\right)$. This yields eq. (40).

By assuming the vanishing correlation between $\boldsymbol{w}_i$ and $\boldsymbol{w}_j$ for $i\neq j$, we can write

$$\chi_{(ai)(bj)} \approx \delta_{ij}\left\{\begin{array}{cc} (\chi_i)_{ab} & (a,b\in\hat{A}_i) \\ 0 & (\text{otherwise}) \end{array}\right. . \tag{65}$$

These leads to:

$$\left(C_\mu^{\backslash\mu}\right)_{ab} = \sum_{ij} x_{\mu i}x_{\mu j}\chi_{(ai)(bj)}^{\backslash\mu} \approx \sum_i x_{\mu i}^2 (\chi_i)_{ab} \approx \sigma_x^2 \sum_i (\chi_i)_{ab} \equiv (C_{\mathrm{SA}})_{ab} \tag{66}$$

The μ-dependence through $x_{\mu i}^2$ is neglected at the last step, because the sum $\sum_i$ would mask such a weak μ-dependence as long as strong heterogeneity in $\{x_{\mu i}\}_\mu$ is absent. Similarly, we may write the sum inside the parentheses of the righthand side of eq. (64) as:

$$\sum_\nu \Gamma_i^\nu \approx \sigma_x^2 \sum_\nu (I_L + F^\nu C_{\mathrm{SA}})^{-1}F^\nu. \tag{67}$$

Inserting eqs. (65-67) into eq. (64), we obtain eq. (41).

Careful readers may be concerned about the neglected μ-dependence of $\chi^{\backslash\mu}$, as well as that of $G^{\backslash\mu}$. If this can be neglected, may we replace $G^{\backslash\mu}$ with G from the beginning at eq. (21)? The answer is of course no. The reason is that the difference between $G^{\backslash\mu}$ and G is not negligible if they are "projected" onto X^μ as in eq. (21). If they are projected onto other directions perpendicular to X^μ, the difference is actually tiny and can be neglected, but for computing the factor $C_\mu^{\backslash\mu}$ we need to take into account this difference appropriately. This results in the additional factor $(I-F_\mu C_\mu)^{-1}$ in eq. (24). In the SA approximation, the factor C is computed based on neglecting the difference between G and $G^{\backslash \mu}$. As a result we cannot discriminate the two factors C_μ and $C_\mu^{\backslash \mu}$. This consideration implies that our SA estimation of C, C_{SA}, should be applied to $C_\mu^{\backslash \mu}$ in eq. (21) and should NOT be applied to C_μ in eq. (24), because the latter formula formally takes into account the difference in advance.

References

Tomohiro Ando and Ruey Tsay. Predictive likelihood for bayesian model selection and averaging. *International Journal of Forecasting*, 26(4):744–763, 2010.

Jean Barbier, Nicolas Macris, Mohamad Dia, and Florent Krzakala. Mutual information and optimality of approximate message-passing in random linear estimation. *arXiv preprint arXiv:1701.05823*, 2017.

Mohsen Bayati and Andrea Montanari. The dynamics of message passing on dense graphs, with applications to compressed sensing. *IEEE Transactions on Information Theory*, 57 (2):764–785, 2011.

Jan F Bjornstad. Predictive likelihood: a review. *Statistical Science*, pages 242–254, 1990.

Burak Çakmak and Manfred Opper. Expectation propagation for approximate inference: Free probability framework. *CoRR*, abs/1801.05411, 2018. URL `http://arxiv.org/abs/1801.05411`.

David L Donoho, Arian Maleki, and Andrea Montanari. Message-passing algorithms for compressed sensing. *Proceedings of the National Academy of Sciences*, 106(45):18914–18919, 2009.

Jerome Friedman, Trevor Hastie, and Rob Tibshirani. Regularization paths for generalized linear models via coordinate descent. *Journal of statistical software*, 33(1):1, 2010.

Gene H Golub, Michael Heath, and Grace Wahba. Generalized cross-validation as a method for choosing a good ridge parameter. *Technometrics*, 21(2):215–223, 1979.

Yoshiyuki Kabashima. A CDMA multiuser detection algorithm on the basis of belief propagation. *Journal of Physics A: Mathematical and General*, 36(43):11111, 2003.

Yoshiyuki Kabashima and Mikko Vehkaperä. Signal recovery using expectation consistent approximation for linear observations. In *Information Theory (ISIT), 2014 IEEE International Symposium on*, pages 226–230. IEEE, 2014.

Balaji Krishnapuram, Lawrence Carin, Mario AT Figueiredo, and Alexander J Hartemink. Sparse multinomial logistic regression: Fast algorithms and generalization bounds. *IEEE transactions on pattern analysis and machine intelligence*, 27(6):957–968, 2005.

Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

M. Lichman. UCI machine learning repository, 2013. URL `http://archive.ics.uci.edu/ml`.

Tomoyuki Obuchi. Matlab package of ACV on MLR. `https://github.com/T-Obuchi/AcceleratedCVonMLR_matlab`, 2017.

Tomoyuki Obuchi and Yoshiyuki Kabashima. Cross validation in lasso and its acceleration. *Journal of Statistical Mechanics: Theory and Experiment*, 2016(5):53304–53339, 2016.

Tomoyuki Obuchi, Shiro Ikeda, Kazunori Akiyama, and Yoshiyuki Kabashima. Accelerating cross-validation with total variation and its application to super-resolution imaging. *PloS one*, 12(12):e0188012, 2017.

Manfred Opper and Ole Winther. Mean field approach to bayes learning in feed-forward neural networks. *Physical review letters*, 76(11):1964, 1996.

Manfred Opper and Ole Winther. A mean field algorithm for bayes learning in large feed-forward neural networks. In *Advances in Neural Information Processing Systems*, pages 225–231, 1997.

Manfred Opper and Ole Winther. *Gaussian processes and SVM: Mean field results and leave-one-out*, pages 43–65. MIT, 10 2000a. ISBN 0262194481. Massachusetts Institute of Technology Press (MIT Press) Available on Google Books.

Manfred Opper and Ole Winther. Gaussian processes for classification: Mean-field algorithms. *Neural computation*, 12(11):2655–2684, 2000b.

Manfred Opper and Ole Winther. Adaptive and self-averaging thouless-anderson-palmer mean-field theory for probabilistic modeling. *Physical Review E*, 64(5):056131, 2001a.

Manfred Opper and Ole Winther. Tractable approximations for probabilistic models: The adaptive thouless-anderson-palmer mean field approach. *Physical Review Letters*, 86(17): 3695, 2001b.

Manfred Opper and Ole Winther. Expectation consistent approximate inference. *Journal of Machine Learning Research*, 6(Dec):2177–2204, 2005.

Kamiar Rahnama Rad and Arian Maleki. A scalable estimate of the extra-sample prediction error via approximate leave-one-out. *arXiv preprint arXiv:1801.10243*, 2018.

Mark Schmidt. Graphical model structure learning with l1-regularization. *University of British Columbia*, 2010.

Takashi Takahashi and Tomoyuki Obuchi. Python package of ACV on MLR. `https://github.com/T-Obuchi/AcceleratedCVonMLR_python`, 2017.

David J Thouless, Philip W Anderson, and Robert G Palmer. Solution of'solvable model of a spin glass'. *Philosophical Magazine*, 35(3):593–601, 1977.

Hastie Trevor, Tibshirani Robert, and Friedman Jerome. *The Elements of Statistical Learning; Data Mining, Inference, and Prediction.* Springer-Verlag New York, 2009. doi: 10.1007/978-0-387-84858-7.

Vladimir Vapnik and Olivier Chapelle. Bounds on error expectation for support vector machines. *Neural computation*, 12(9):2013–2036, 2000.

Shuaiwen Wang, Wenda Zhou, Haihao Lu, Arian Maleki, and Vahab Mirrokni. Approximate leave-one-out for fast parameter tuning in high dimensions. *arXiv preprint arXiv:1807.02694*, 2018.

Hui Zou and Trevor Hastie. Regularization and variable selection via the elastic net. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 67(2):301–320, 2005.