A lossless online Bayesian classifier.

NGUYEN, T.T.T., NGUYEN, T.T., SHARMA, R. and LIEW, A. W.-C

2019

A Lossless Online Bayesian Classifier

Thi Thu Thuy Nguyen[1], Tien Thanh Nguyen[2], Rabi Sharma[3], Alan Wee-Chung Liew[1]

[1]School of Information and Communication Technology, Griffith University, Australia

[2]School of Computing Science and Digital Media, Robert Gordon University, Aberdeen, UK

[3]CVPR unit, Indian Statistical Institute, Kolkata, India

Abstract: We are living in a world progressively driven by data. Besides mining big data which cannot be altogether stored in the main memory as required by traditional offline methods, the problem of learning rare data that can only be collected over time is also very prevalent. Consequently, there is a need of online methods which can handle arriving data and offer the same accuracy as offline methods. In this paper, we introduce a new lossless online Bayesian-based classifier which uses the arriving data in a 1-by-1 manner and discards each data right after use. The lossless property of our proposed method guarantees that it can reach the same prediction model as its offline counterpart regardless of the incremental training order. Experimental results demonstrate its superior performance over many well-known state-of-the-art methods in the literature.

Keywords: Online learning, lossless methods, online classifiers, Bayesian method, variational inference, multivariate Gaussian

1. Introduction

In machine learning, online learning is a learning mechanism in which data arriving in a sequential manner are processed incrementally, and predictions must be made if required at any time before all the data are seen. This situation is very popular in real-time applications such as

medical data analysis (e.g. [18]), network traffic (e.g. [23]), and financial markets (e.g. [24]), in which data is collected gradually and not all of the data is available at the beginning of the analysis. As opposed to traditional batch algorithms where the prediction is made based on learning the entire training dataset at once, strictly online algorithms do not require the whole dataset to be stored or loaded into memory, but just make use of a single/set of observations and then discard them before the next observations are used (this property is called discard-after-learn or one-pass-thrown-away).

In this paper, we focus on online supervised classification algorithms which usually perform the following three main steps:

- *Predict*: When a new observation $\mathbf{x}_t$ arrives, a prediction $\hat{y}_t$ is made using the current model L_t.
- *Calculate the suffered loss:* After making the prediction, the true label y_t is revealed, and the loss $l(y_t, \hat{y}_t)$ can be estimated to measure the difference between the learner's prediction and the revealed true label y_t.
- *Update:* Based on the result of the loss, the learner can use the observation $(\mathbf{x}_t, y_t)$ to update the classification model $(L_t \rightarrow L_{t+1})$.

Many of the online methods are built based on their offline versions. For example, from offline classifiers like linear methods, Bayesian methods, decision trees, Bagging, Boosting or Random Forest, their online descendants can be found like online linear methods (e.g. [6-11, 14, 33, 38, 39, 40]), online Bayesian classifiers (e.g. [2, 26]) , online decision trees (e.g. [13, 37]), online Bagging [29], online Boosting [29], or online Random Forest [34], respectively. While enjoying computational efficiency, online methods generally suffer degradation in performance compared with their offline counterparts due to the inability of accessing the full training set. Moreover, the sequential nature of the model updating process also introduces order-dependence bias to the final result. In the literature, Incremental Tree Induction (ITI) [37] is a lossless

incremental method which can produce the same tree for a dataset regardless of the incremental training order or whether the tree is induced incrementally or not (batch setting). However, to be lossless, it violates the requirement of a strictly online method by having to retain all the previous training data for revisiting decisions, thus prohibiting its use on large datasets. Besides ITI [37], Online Naïve Bayes (discrete version [2] and continuous version [26]) is also lossless, but its "Naïve" assumptions do not allow the predictive model to achieve a high accuracy on a diversity of datasets. In this paper, we describe a new lossless Online Variational Inference for multivariate Gaussian based method (OVIG), which is demonstrated to outperform many state-of-the-art online methods, namely Adaptive Regularization of Weights (AROW) [10], Soft Confidence Weighted (SCW) [38], and online Bagging with Hoeffding trees as the based learners (OBHT) [13, 29], as well as the widely-used Passive Aggressive learning (PA) [8], and Online Gradient Descent (OGD) [40] algorithms. As opposed to ITI [37], our proposed method does not need to store more than a single observation in the main memory. It also does not require the discretization of attributes before testing and training as in the discrete version of Online Naïve Bayes [2] where relevant counts for the predictive model are maintained in tables. Compared to the continuous version of Online Naïve Bayes [26], the new lossless method is significantly more time-efficient. Moreover, by updating the predictive model only when an arriving observation is classified incorrectly OVIG becomes the first lossless conservative online method. Additionally, OVIG is naturally parallelizable as it learns the predictive models for different classes independently and for each class, the predictive model and its relevant sufficient statistics can be updated independently. As a generative (Bayesian) method, OVIG can give access to the full data distribution and class distributions at any time which is needed for many advanced online learning tasks such as online semi-supervised learning, imbalanced learning, and cost-sensitive learning (see e.g. [27]).

The remainder of this paper is organized as follows. In section 2, we briefly review several online learning methods, especially the ones we used as benchmark algorithms. The

Bayesian-based online learning methods are further discussed in section 3. Section 4 presents our proposed algorithm and the next section is for experimental studies. Finally, the conclusion is given in Section 6.

2. Related work

In literature, there are several online learning methods which are mainly distinguished by the different type of loss function $l(y_t, \hat{y}_t)$ or the way of updating $L_t \rightarrow L_{t+1}$. Many of them only update when they make wrong prediction (using 0-1 loss function) (which are called conservative algorithms). The Perceptron algorithm, a binary linear classifier [33], is perhaps the oldest conservative algorithm. Crammer and Singer [11] extended the binary Perceptron algorithm to a family of multiclass Perceptron algorithms with the label predicted as $\hat{y}_t = \underset{i \in \{1,\cdots,K\}}{\operatorname{argmax}} \mathbf{w}_i \cdot \mathbf{x}_t$, here K is the number of classes and $\mathbf{w}_i$ is the weight vector of class i $(i = 1, \dots, K)$. Perceptron, and some other popular methods like PA (Passive Aggressive learning) [7, 8] and OGD (Online Gradient Descent) [40] are first-order online linear methods. Recently second-order online linear classifiers such as SOP (Second-order Perceptron) [6], CW (Confidence Weighted learning) [14], IELLIP (Improved Ellipsoid Method for Online Learning) [39], SCW (Soft Confidence Weighted) [38], and AROW (Adaptive Regularization of Weights) [9,10] have been explored to better exploit the underlying structures between features. Instead of giving point estimate like first-order learners, most of the second-order learning algorithms typically assume the weight vector follows a Gaussian distribution $\mathbf{w} \sim \mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma})$. They not only find the most likely solution for $\mathbf{w}$ but also the distribution of all possible solutions, hence taking advantage of the training data more efficiently. Although linear methods are efficient in computation and memory, they learn linear prediction models which are not flexible enough for many real-world applications. Moreover, it is often not straightforward to convert from binary linear classifiers to multiclass linear classifiers.

Tree-based models are also a popular family among the supervised online learning models. An attractive feature of prevalent trees like CART [4] (or the related C4.5 [31]) is that the algorithm asks a sequence of hierarchical Boolean questions (e.g., is $x^{(i)} \leq \theta_j$?, where θ_j is a threshold value), which is relatively simple to understand and interpret by humans. When converted into online setting, early incremental trees are either memory intensive as all of the previous training data must be retained (ITI [37])), or they have to discard important information if parent nodes change. In 2000, the state-of-the-art Hoeffding tree [13] was introduced. It is an incremental, anytime decision tree, that is capable of learning from massive data streams, assuming that the underlying distribution that generates the data does not change over time. Among trees, Hoeffding tree is used the most since it has good guarantee of performance not shared by other incremental decision tree learners. To measure the number of observations needed to estimate some sufficient statistics to within a prescribed precision, it exploits the so-called Hoeffding bound, which states that, with probability $1 - \delta$, the true mean of a random variable of range R will not differ from the estimated mean after n independent observations by more than $\epsilon = \sqrt{\frac{R^2 ln(1/\delta)}{2n}}$. With this bound, Hoeffding tree's output is shown to be asymptotically nearly identical to that of a non-incremental learner using infinitely many examples. However, in the implementation of Hoeffding tree, checking for split at a node is often expensive. Therefore, after initializing the predictive model with a number of initial observations, it only checks for split at a node after every new n_{min} (default value = 200) observations have reached that node. As in batch learning, trees often need to be used in an ensemble framework to get high performance [5], one would do the same with the online setting. Hoeffding trees are now mainly used as base learners in online meta-learning methods (see for example [30]).

Besides ensembles of incremental decision trees, online committee classifiers can use many different online methods as base learners to attain better performance. In an ensemble

framework, we might train multiple different models and then make predictions by a combining method on the models' output. Two well-known variants of online committee methods are online Bagging and online Boosting [29]. Online Bagging simulates the bootstrap process of offline Bagging by sending $\mathcal{K}$ copies of each new sample to update each base model, where $\mathcal{K}$ follows the Poisson (λ = 1) distribution. Meanwhile, online Boosting trains a series of base models $\mathrm{h}_1, \ldots, \mathrm{h}_M$, each h_m is learned from each new sample $\mathcal{K}$ times, where $\mathcal{K} \sim$ Poisson(λ), λ is determined based on the classification result of preceding model h_{m-1}. Although ensemble method can inherit the good properties of base learners and give better accuracy, they also experience some disadvantages. For example, deciding on the number of base classifiers is non-trivial and a large number can lead to expensive computation while a small number may not give expected results.

Recently, there are a number of online algorithms using hyper-elliptical capsules as their learning units including a versatile elliptic basis function (VEBF) neural network [21, 22] or a multi-stratum network [36]. In can be seen that hyper-elliptical capsules and multivariate Gaussians share a lot of common properties. First, both of them can be represented by the centre vector (mean) and the covariance matrix. Furthermore, they can be used as building blocks for many flexible algorithms like [21, 22, 36] and the proposed method OVIG. However, our approach not only updates point estimates of the mean and the covariance matrix over time, but also the distribution of their likely values. It is worth pointing out that when we have a distribution, we have the point estimate (as the mean or mode of the distribution) and the confidence of our point estimate (inferred from the covariance) at the same time. That is an advantage of second-order methods.

In the next section, we will discuss the online Bayesian algorithms. These algorithms are very flexible generative algorithms which give us access to the posterior probabilities $p(y = k|\mathbf{x})$ as well as data distribution $p(\mathbf{x})$ and $p(\mathbf{x}, y)$**.** This information is especially valuable in an online setting, where samples are discarded after use and cannot be retrieved later.

3. Bayesian classifiers

Given an observation to be classified represented by a vector $\mathbf{x} = \left(x^{(1)}, \cdots, x^{(2)}\right) \in \mathbb{R}^D$. A Bayesian classifier based on Bayes' theorem predicts the label y of $\mathbf{x}$ from the label set $\{1,2,\cdots,K\}$ as

$$y = \underset{k\in\{1,2,\cdots,K\}}{\operatorname{argmax}}\ p(y=k|\mathbf{x}) \sim \underset{k\in\{1,2,\cdots,K\}}{\operatorname{argmax}}\ p(y=k)p(\mathbf{x}|y=k) \tag{1}$$

where $p(y=k|\mathbf{x})$ is the posterior probability that $\mathbf{x}$ belongs to the class k, $p(y=k)$ is the prior probability of class k, and $p(\mathbf{x}|y)$ is the class conditional probability density function, respectively.

The class prior $p(y=k)$ can often be estimated simply from the fractions of training data in each of the classes. For online setting, at time step t

$$p(y=k) = \frac{n_t^k}{n_t} \tag{2}$$

where n_t^k denotes the number of observations of class k arriving before $\mathbf{x}_\mathbf{t}$, and n_t denotes the number of all observations arriving before $\mathbf{x}_\mathbf{t}$.

Based on the way of approximating $p(\mathbf{x}|y=k)$, $k \in \{1,\ldots,K\}$, we have different Bayesian algorithms. Naïve Bayes is the simplest Bayesian classifier which is called 'naive' because it assumes independence of the attributes given the label, i.e.

$$\forall k \in \{1,\cdots,K\},\ p(\mathbf{x}|y=k) = p\left(x^{(1)}\middle|y=k\right)\ldots p\left(x^{(D)}\middle|y=k\right) \tag{3}$$

A version of Naïve Bayes classifiers with continuous attributes is introduced in [26], where every attribute $x^{(i)}$ of $\mathbf{x}$ is assumed to follow a univariate Gaussian distribution given the label $y=k$: $p\left(x^{(i)}\middle|y=k\right) = \mathcal{N}\left(x^{(i)}|\mu^{(i)},\sigma^{(i)2}\right),\ i \in \{1,\cdots,D\}$. The offline maximum likelihood

estimates of parameters $\mu^{(i)}, \sigma^{(i)2}$ based on the training set of n observations $\mathbb{X} = \{\mathbf{x}_1, \cdots, \mathbf{x}_n\}$ from class k can be found in standard statistics textbooks as:

$$\mu_n^{(i)} = \frac{1}{n}\sum_{j=1}^{n} x_j^{(i)} \tag{4}$$

$$\sigma_n^{(i)2} = \frac{1}{n}\sum_{j=1}^{n} \left(x_j^{(i)} - \mu^{(i)}\right)^2 \tag{5}$$

Consequently, the incremental formulae for online Naïve Bayes for Gaussian method (ONBG) [26] are as follows.

$$\mu_t^{(i)} = \frac{t-1}{t}\mu_{t-1}^{(i)} + \frac{1}{t}x_t^{(i)} \tag{6}$$

$$\sigma_t^{(i)2} = \frac{t-1}{t}\sigma_{t-1}^{(\mathrm{i})2} + \frac{1}{t-1}\left(x_t^{(i)} - \mu_t^{(i)}\right)^2 \tag{7}$$

On the other hand, Naïve Bayes (NB) method in MOA [2] supposes that each $x^{(i)}$ can take n_i different discrete values. Upon receiving an unlabelled observation $\mathbf{x} = \left(x^{(1)} = v^{(1)}, \ldots, x^{(D)} = v^{(D)}\right)$, Naïve Bayes classifiers estimate the probability of $\mathbf{x}$ belonging to class $y = k$ as:

$$p(y = k|\mathbf{x}) \sim p(y = k)p(\mathbf{x}|y = k) \;=\; p(y = k)\prod_{i=1}^{D} p\left(x^{(i)} = v^{(i)}|y = k\right)$$

$$= p(y = k)\prod_{i=1}^{D}\frac{p\left(x^{(i)}=v^{(i)},y=k\right)}{p(y=k)} \tag{8}$$

The values $p\left(x^{(i)} = v^{(i)}, y = k\right)$ and $p(y = k)$ are estimated from the training data which is summarized in a 3-dimensional table that stores for each triple $\left(x^{(i)}, v_j^{(i)}, k\right)$ a count $N_{i,j,k}$ of training observations with $x^{(i)} = v_j^{(i)}$ and $y = k$, together with a 1-dimensional table for the counts of $y = k$. Upon receiving a new observation, the relevant counts are increased. It can be seen that if ONBG [26] and NB [2] always update their model, after learning on the training set, their online and offline versions will have identical performance, i.e. they are lossless online methods.

4. A novel lossless Bayesian method

Recently, Nguyen et al. [28] proposed the Variational Inference (VI) for multivariate Gaussian distribution (VIG) algorithm to approximate $p(\mathbf{x}|y=k)$ for each class k. The VIG algorithm has been demonstrated to offer superior performance for batch learning under an ensemble framework. In this study, we propose a lossless online version of VIG named OVIG, which not only theoretically converges to its offline counterpart but also achieves the same predictive model regardless of the incremental training order. Similar to the two aforementioned lossless methods, we inspect data strictly in a 1-by-1 fashion, each used observation is discarded before the next one is obtained. However, our refined update mechanism makes OVIG much more stable. In particular, as opposed to the constantly updating frameworks of ONBG to maintain the lossless property, OVIG is a conservative method, that means it only updates its predictive model when it makes a wrong prediction. To the best of our knowledge, OVIG is the first online method which is simultaneously discard-after-learn, conservative, and lossless. OVIG explores the deep underlying structure of the data, making its model effective for even very small datasets where many other online methods have difficulty to learn (see section 5).

4.1 Offline Variational Inference for multivariate Gaussian (VIG)

Before describing OVIG, we briefly summarize the VIG method [28]. VIG applies Variational Inference technique to approximate the multivariate Gaussian model $\mathcal{N}(\mathbf{x}|\boldsymbol{\mu},\boldsymbol{\Sigma})$ for $p(\mathbf{x}|y=k)$ of each class $k \in \{1,\dots,K\}$. In contrast to maximum-likelihood learning, VI treats the parameters $(\boldsymbol{\mu},\boldsymbol{\Sigma})$ as random variables and a prior is placed over the parameters to obtain the posterior distribution $p(\boldsymbol{\mu},\boldsymbol{\Sigma}|\mathbb{X})$, where $\mathbb{X}=\{\mathbf{x}_j|j=1,\dots,n\}$ is the training set, which is assumed to be drawn independently from the multivariate Gaussian distribution $\mathcal{N}(\mathbf{x}|\boldsymbol{\mu},\boldsymbol{\Sigma})$. The idea behind VI method is to approximate the posterior distribution $p(\boldsymbol{\mu},\boldsymbol{\Sigma}|\mathbb{X})$ of hidden variables $\boldsymbol{\mu},\boldsymbol{\Sigma}$ given observed data $\mathbb{X}$ by a more easily accessible distribution $q(\boldsymbol{\mu},\boldsymbol{\Sigma})$ which minimizes the Kullback-Leibler divergence $\mathrm{KL}(q||p)$ between $p(\boldsymbol{\mu},\boldsymbol{\Sigma}|\mathbb{X})$ and $q(\boldsymbol{\mu},\boldsymbol{\Sigma})$. To

minimize $\mathrm{KL}(q||p)$, we maximize $\mathcal{L}(q) = \ln p(\mathbb{X}) - \mathrm{KL}(q||p)$. As $\mathrm{KL}(q||p) \geq 0$, $\mathcal{L}(q)$ is a lower bound on the log marginal probability $\ln p(\mathbb{X})$.

The conjugate prior of a multivariate Gaussian distribution, $p(\boldsymbol{\mu}, \boldsymbol{\Lambda})$, where $\boldsymbol{\Lambda} = \boldsymbol{\Sigma}^{-1}$ is the precision matrix, with unknown $\boldsymbol{\mu}$ and $\boldsymbol{\Lambda}$, is given by the Gaussian-Wishart distribution: $p(\boldsymbol{\mu}, \boldsymbol{\Lambda}) = p(\boldsymbol{\mu}|\boldsymbol{\Lambda})p(\boldsymbol{\Lambda})$, where $p(\boldsymbol{\mu}|\boldsymbol{\Lambda})$ is a Gaussian distribution:

$$p(\boldsymbol{\mu}|\boldsymbol{\Lambda}) = \mathcal{N}(\boldsymbol{\mu}|\mathbf{m}_0, (\beta_0\boldsymbol{\Lambda})^{-1}) = (2\pi)^{-\frac{D}{2}}|\beta_0\boldsymbol{\Lambda}|^{\frac{1}{2}} \exp\left\{-\frac{1}{2}(\boldsymbol{\mu} - \mathbf{m}_0)^{\mathrm{T}}\beta_0\boldsymbol{\Lambda}(\boldsymbol{\mu} - \mathbf{m}_0)\right\} \tag{9}$$

and $p(\boldsymbol{\Lambda})$ is a Wishart distribution:

$$p(\boldsymbol{\Lambda}) = \mathcal{W}(\boldsymbol{\Lambda}|\mathbf{W}_0, v_0) = \mathrm{B}(\mathbf{W}_0, v_0)|\boldsymbol{\Lambda}|^{\frac{(v_0 - D - 1)}{2}} \exp\left\{-\frac{1}{2}\mathrm{Tr}(\mathbf{W}_0^{-1}\boldsymbol{\Lambda})\right\}, \tag{10}$$

$$\mathrm{B}(\mathbf{W}_0, v_0) = |\mathbf{W}_0|^{\frac{-v_0}{2}}\left(2^{\frac{v_0 D}{2}}\pi^{\frac{D(D-1)}{4}}\prod_{i=1}^{D}\Gamma\left(\frac{v_0 + 1 - i}{2}\right)\right)^{-1} \tag{11}$$

where $\mathbf{m}_0$ and β_0 are the D-dimension mean vector and the scale of the precision matrix $\boldsymbol{\Lambda}$ of the Gaussian distribution $p(\boldsymbol{\mu}|\boldsymbol{\Lambda})$, $\mathbf{W}_0$ and v_0 are the $D \times D$-dimension scale matrix and the number of degrees of freedom of the Wishart distribution $p(\boldsymbol{\Lambda})$, $\mathrm{Tr}(.)$denotes the trace operator of a matrix, and $\Gamma(.)$ denotes the Gamma function defined by $\Gamma(.) = \int_0^\infty x^{t-1}e^{-x}\mathrm{d}x$.

From [28], the variational solution for parameters $\boldsymbol{\mu}$ and $\boldsymbol{\Lambda} = \boldsymbol{\Sigma}^{-1}$ are given as follows.

$\boldsymbol{\mu} \sim \mathrm{q}(\boldsymbol{\mu}) = \mathcal{N}(\boldsymbol{\mu}|\mathbf{m}, \mathbf{H}^{-1})$, is a Gaussian with mean $\mathbf{m}$ and precision $\mathbf{H}$ given by (12) and (13):

$$\mathbf{m} = \frac{\beta_0\mathbf{m}_0 + n\bar{\mathbf{x}}}{\beta_0 + n} \tag{12}$$

$$\mathbf{H} = (\beta_0 + n)\mathbb{E}[\boldsymbol{\Lambda}]. \tag{13}$$

$\boldsymbol{\Lambda} \sim q(\boldsymbol{\Lambda}) = \mathcal{W}(\boldsymbol{\Lambda}|\mathbf{W}, v)$, is a Wishart with the number of degrees of freedom v and the scale matrix $\mathbf{W}$ given by (14) and (15):

$$v = v_0 + n + 1 \tag{14}$$

$$\mathbf{W}^{-1} = \mathbf{W}_0^{-1} + (\beta_0 + n)\mathbf{H}^{-1} + \mathbf{S} + \frac{\beta_0 n}{\beta_0 + n}\mathbf{J} \tag{15}$$

where

$$\bar{\mathbf{x}} = \frac{1}{n}\sum_{j=1}^{n} \mathbf{x}_j \tag{16}$$

$$\mathbf{S} = \sum_{j=1}^{n}\left(\mathbf{x}_j - \bar{\mathbf{x}}\right)\left(\mathbf{x}_j - \bar{\mathbf{x}}\right)^{\mathrm{T}} \tag{17}$$

$$\mathbf{J} = (\bar{\mathbf{x}} - \mathbf{m}_0)(\bar{\mathbf{x}} - \mathbf{m}_0)^{\mathrm{T}} \tag{18}$$

Thus, we have expressions for the optimal distributions of $\boldsymbol{\mu}$ and $\boldsymbol{\Lambda}$, each of which depends on values evaluated with respect to the other distributions such as the expectation $\mathbb{E}[\boldsymbol{\Lambda}]$ of $\boldsymbol{\Lambda}$ and the precision $\mathbf{H}$ of $\boldsymbol{\mu}$. To start the iterative re-estimation procedure of VI method, we can make an initial guess for the moment $\mathbb{E}[\boldsymbol{\Lambda}]$, say, $\mathbb{E}[\boldsymbol{\Lambda}] = v_0\mathbf{W}_0$, and use this to re-compute the distribution $q(\boldsymbol{\mu})$. Given this revised distribution we can then use the precision $\mathbf{H}$ to recompute the distribution $q(\boldsymbol{\Lambda})$, and so on.

The lower bound $\mathcal{L}(q)$ of the Variational Inference for the multivariate Gaussian distribution is given by

$$\mathcal{L}(q) = \ln B(\mathbf{W}_0, v_0) - \ln B(\mathbf{W}, v) - \frac{1}{2}[nD\ln(2\pi) - D\ln(\beta_0) - vD + \ln|\mathbf{H}| + v\mathrm{Tr}(\mathbf{SW}) +$$

$$v\mathrm{Tr}(\mathbf{W}_0^{-1}\mathbf{W}) + \frac{\beta_0 n v}{\beta_0 + n}\mathrm{Tr}(\mathbf{JW})]. \tag{19}$$

We have the following algorithm for multivariate Gaussian distribution estimation [28].

Algorithm 1. VIG

Input: Dataset $\mathbb{X}$, threshold ε, $\mathbf{m}_0$, β_0, v_0, $\mathbf{W}_0$, $\mathbb{E}[\boldsymbol{\Lambda}] = v_0\mathbf{W}_0$
Output: $\mathbf{m}$, $\mathbf{H}$ of $q(\boldsymbol{\mu}) = \mathcal{N}(\boldsymbol{\mu}|\mathbf{m}, \mathbf{H}^{-1})$ and $\mathbf{W}$, v of $q(\boldsymbol{\Lambda}) = \mathcal{W}(\boldsymbol{\Lambda}|\mathbf{W}, v)$
$i \coloneqq 1$
for each i
  Update $\mathbf{m}$, $\mathbf{H}$ using (12), (13)
  Update v, $\mathbf{W}$ using (14), (15)
  if $i > 1$ and $\mathcal{L}_i(q) - \mathcal{L}_{i-1}(q) < \varepsilon$

break;
end if
$i \coloneqq i + 1$
end for

In the algorithm above, the four variables of $q(\boldsymbol{\mu})$ and $q(\boldsymbol{\Lambda})$ are updated step by step from their initial values. The updating process will stop when the change in lower bound value $\mathcal{L}(q)$ is smaller than a specified threshold ε. Only 3 or 4 iterations are typically needed to achieve convergence with a threshold set as $\varepsilon = 1e - 10$ [28].

Estimates for the variables can then be derived in the standard Bayesian ways, e.g. calculating the mean of the distribution to get a single point estimate or deriving a credible interval, highest density region, etc. In practice, ones often choose $\mathbb{E}[\boldsymbol{\mu}] = \mathbf{m}$ as the value of $\boldsymbol{\mu}$, and $\mathbb{E}[\boldsymbol{\Lambda}] = v\mathbf{W}$ as the value of $\boldsymbol{\Lambda}$ when they need to evaluate $\mathcal{N}(\mathbf{x}|\boldsymbol{\mu}, \boldsymbol{\Lambda}^{-1})$.

4.2 The lossless OVIG

In this section, we introduce the lossless online version of VIG. We assume that the class conditional probability density functions $p(\mathbf{x}|y = k)$ are multivariate Gaussians $N(\boldsymbol{\mu}_k, \boldsymbol{\Lambda}_k^{-1})$ for $k = 1, \ldots, K$, and using Variational Inference technique to update the distributions of $\boldsymbol{\mu}_k$ and $\boldsymbol{\Lambda}_k$.

To make the algorithm operate in an online mode, *two main questions* need to be answered:

- *How to update the model on-the-fly?*
- *When to update the model?*

To answer the *first question*, we notice that the updating equations (12-15) of the hyperparameters $\mathbf{m}, \mathbf{H}, v, \mathbf{W}$ essentially depend on the sufficient statistics $\bar{\mathbf{x}}$, $\mathbf{S}$, $\mathbf{J}$, where $\mathbf{J}$ can calculated from $\bar{\mathbf{x}}$. Therefore, when a new sample arrives, instead of update $\mathbf{m}, \mathbf{H}, v, \mathbf{W}$ directly, we can update two sufficient statistics $\bar{\mathbf{x}}$, $\mathbf{S}$.

Let us consider the results (16-17) for the sufficient statistics $\bar{\mathbf{x}}$, $\mathbf{S}$, which are denoted as $\bar{\mathbf{x}}_n, \mathbf{S}_n$, when they are based on n observations.

$$\bar{\mathbf{x}}_t = \frac{1}{t}\sum_{j=1}^{t}\mathbf{x}_j = \frac{1}{t}\sum_{j=1}^{t-1}\mathbf{x}_j + \frac{1}{t}\mathbf{x}_t = \frac{t-1}{t}\bar{\mathbf{x}}_{t-1} + \frac{1}{t}\mathbf{x}_t$$

$$\mathbf{S}_t = \sum_{j=1}^{t}(\mathbf{x}_j - \bar{\mathbf{x}}_t)(\mathbf{x}_j - \bar{\mathbf{x}}_t)^T$$

$$= (\mathbf{x}_t - \bar{\mathbf{x}}_t)(\mathbf{x}_t - \bar{\mathbf{x}}_t)^T + \sum_{j=1}^{t-1}(\mathbf{x}_j - \bar{\mathbf{x}}_{t-1} + \bar{\mathbf{x}}_{t-1} - \bar{\mathbf{x}}_t)(\mathbf{x}_j - \bar{\mathbf{x}}_{t-1} + \bar{\mathbf{x}}_{t-1} - \bar{\mathbf{x}}_t)^T$$

$$= (\mathbf{x}_t - \bar{\mathbf{x}}_t)(\mathbf{x}_t - \bar{\mathbf{x}}_t)^T + \sum_{j=1}^{t-1}(\mathbf{x}_j - \bar{\mathbf{x}}_{t-1})(\mathbf{x}_j - \bar{\mathbf{x}}_{t-1})^T + \sum_{j=1}^{t-1}(\bar{\mathbf{x}}_{t-1} - \bar{\mathbf{x}}_t)(\bar{\mathbf{x}}_{t-1} - \bar{\mathbf{x}}_t)^T.$$

Here we have just used $\sum_{j=1}^{t-1}(\mathbf{x}_j - \bar{\mathbf{x}}_{t-1})(\bar{\mathbf{x}}_{t-1} - \bar{\mathbf{x}}_t)^T = \{\sum_{j=1}^{t-1}\mathbf{x}_j - (t-1)\bar{\mathbf{x}}_{t-1}\}(\bar{\mathbf{x}}_{t-1} - \bar{\mathbf{x}}_t)^T = 0$, and similarly, $\sum_{j=1}^{t-1}(\bar{\mathbf{x}}_{t-1} - \bar{\mathbf{x}}_t)(\mathbf{x}_j - \bar{\mathbf{x}}_{t-1})^T = 0$.

As $\sum_{j=1}^{t-1}(\mathbf{x}_j - \bar{\mathbf{x}}_{t-1})(\mathbf{x}_j - \bar{\mathbf{x}}_{t-1})^T = \mathbf{S}_{t-1}$, and $\bar{\mathbf{x}}_{t-1} = (t\bar{\mathbf{x}}_t - \mathbf{x}_t)/(t-1) \Leftrightarrow \bar{\mathbf{x}}_{t-1} - \bar{\mathbf{x}}_t = (\bar{\mathbf{x}}_t - \mathbf{x}_t)/(t-1)$, we have:

$$\mathbf{S}_t = \mathbf{S}_{t-1} + (\mathbf{x}_t - \bar{\mathbf{x}}_t)(\mathbf{x}_t - \bar{\mathbf{x}}_t)^T + \frac{1}{(t-1)^2}\sum_{j=1}^{t-1}(\mathbf{x}_t - \bar{\mathbf{x}}_t)(\mathbf{x}_t - \bar{\mathbf{x}}_t)^T$$

$$= \mathbf{S}_{t-1} + \frac{t}{t-1}(\mathbf{x}_t - \bar{\mathbf{x}}_t)(\mathbf{x}_t - \bar{\mathbf{x}}_t)^{\boldsymbol{T}}.$$

Therefore, sufficient statistics for current training example $(\mathbf{x}_t, \mathrm{y}_t)$ can be updated in a sequential manner:

$$\bar{\mathbf{x}}_t = \frac{t-1}{t}\bar{\mathbf{x}}_{t-1} + \frac{1}{t}\mathbf{x}_t \tag{20}$$

$$\mathbf{S}_t = \mathbf{S}_{t-1} + \frac{t}{t-1}(\mathbf{x}_t - \bar{\mathbf{x}}_t)(\mathbf{x}_t - \bar{\mathbf{x}}_t)^{\boldsymbol{T}} \tag{21}$$

From here, if OVIG always updates its sufficient statistics, after learning on the training set, online and offline VI-based model for multivariate Gaussian will obtain the same sufficient statistics which are needed to update their hyperparameters $\mathbf{m}, \mathbf{H}, v, \mathbf{W}$. After each arrival observation is used to update the sufficient statistics, they will be discarded permanently.

Moving to the *second question*, we only update the predictive model when it makes a wrong prediction, i.e. when the 0-1 loss function is non-negative. To be more precise, the sufficient statistics are always updated, but the hyperparameters and consequently the parameters of the predictive model itself are only updated when the model has an improper performance. Below, the pseudo-code of OVIG is given.

Algorithm 2. OVIG

Input: Dataset $\mathbb{X}$, threshold ε, $\mathbf{m}_0$, β_0, v_0, $\mathbf{W}_0$, $\mathbb{E}[\mathbf{\Lambda}] = v_0\mathbf{W}_0$
for $t = 1,2, \ldots$
  Receive $\mathbf{x}_t$
  Predict $\hat{y}_t = \underset{k\in\{1,2,\cdots,K\}}{\operatorname{argmax}} p(y = k)p(\mathbf{x}|y = k)$
  Reveal the true class label y_t
  Update sufficient statistics $\bar{\mathbf{x}}$, $\mathbf{S}$ for class y_t by (20), (21)
  Discard $\mathbf{x}_t$
  Calculate the suffered loss: $l_t = \mathbb{I}(\hat{y}_t \neq y_t)$
  if $l_t > 0$
    Update sufficient statistics $\mathbf{J}$ for class y_t by (18)
    Use VIG to update $\mathbf{m}, \mathbf{H}, v, \mathbf{W}$ for y_t
  end if
end for

In Algorithm 2, the D-dimension vector $\mathbf{m}_0$, $\beta_0 > 0$, $v_0 > D - 1$, and $D\times D$ dimension symmetric positive definite matrix $\mathbf{W}_0$ are the initial parameters representing prior information, which are simply set to default values in our experiments, i.e. $\mathbf{m}_0$ is a D-dimension vector of zero elements $(0, \ldots, 0)^T$, $\beta_0 = 1$, $v_0 = D$, $\mathbf{W}_0$ is a $D \times D$ dimension identity matrix, $(0, \ldots, 0)^T$ is a D-dimension vector of zero elements, $\mathbf{I}$ is a $D \times D$ dimension identity matrix, D is the dimension of data.

It can be seen that as ONBG [26] and NB [2], OVIG is lossless as the sufficient statistics can be updated sequentially. And after learning on a training set, offline VIG and online OVIG will have the same sufficient statistics. However, unlike ONBG [26], OVIG only updates its predictive model ($N\left(\boldsymbol{\mu}_k, \boldsymbol{\Lambda}_k^{-1}\right), k = 1, \ldots, K$) when needed, that makes it more stable. OVIG is also more flexible then NB [2] as it does not need to discretize continuous attributes before testing and training. Furthermore, OVIG learns models for different classes independently, and for each class it updates the sufficient statistics and predictive model independently, which make it naturally parallelizable. In online learning, parallelization can help decrease the execution time significantly (see for example [35]).

5. Experimental Studies

5.1 Datasets

To evaluate the performance of the proposed lossless online method, we performed experiments on 30 UCI real-world datasets [25] (see Table 1). The data samples are presented to the algorithms sequentially to simulate data streams.

5.2 Experimental settings

We compared OVIG with the widely used first-order linear methods: PA (Passive Aggressive learning) [8], OGD (Online Gradient Descent) [40]; state-of-the-art second-order linear methods: AROW (Adaptive Regularization of Weights) [10], SCW (Soft Confidence Weighted) [38]; well-known OBHT (online Bagging using famous Hoeffding trees as based learners) [13, 29], and lossless ONBG [26]. All algorithms mentioned in the experiments discard instances after using, i.e. they are one-pass-thrown-away online classifiers.

TABLE 1. INFORMATION OF DATASETS IN EVALUATION

Dataset	#Attributes	#Classes	#Observations
Abalone	8	3	4174
Appendicitis	7	2	106

Balance	4	3	625
Banana	2	2	5300
Biodeg	41	2	1055
Blood	4	2	748
Chess-Krvk	6	18	28056
Contraceptive	9	3	1473
Led7digit	7	10	500
Letter	16	26	20000
Libras	90	15	360
Marketing	13	9	6876
Nursery	8	5	12960
Optdigits	64	10	5620
Penbased	16	10	10992
Phoneme	5	2	5404
Poker	10	10	1025009
Ring	20	2	7400
Satimage	36	6	6435
Segment	19	7	2310
Skin-NonSkin	3	2	245057
Sonar	60	2	208
Spambase	57	2	4601
Texture	40	10	5500
Tic-Tac-Toe	9	2	958
Titanic	3	2	2201
Wine-Red	11	6	1599
Wine-White	11	7	4898
Yeast	8	10	1484
Zoo	16	7	101

These four linear benchmark algorithms (PA, OGD, AROW, SCW) have separate codes for binary and multiclass cases which are available from the LIBOL library [19]. For benchmark algorithm that has more than one version with almost similar results, we used the first version in the comparison. Implementation of online Bagging using the default 10 Hoeffding trees as base learners (OBHT) can be found in MOA which is the most popular open source framework for data stream mining [2]. The lossless ONBG is always updated as each observation arrives and in this work, it was used as a single classifier (not in an ensemble framework as in [26]). Default parameters for each benchmark algorithms were used if available. For OVIG, we also used default parameters as discussed in the earlier sections.

We use the prequential evaluation i.e. each arriving instance is first used to test the classification performance of an online algorithm, then it is used to update the model of the

algorithm. For each dataset and each method, we draw 10 random permutations of the whole dataset as the input data to run the test 10 times and take the average error rate and the standard deviation (STD). When comparing our lossless methods with any benchmark algorithms, Wilcoxon signed-rank test [12] with a level of significance of 0.05 is used. In this test, the null hypothesis is that the performances of two methods are not different, and the alternative hypothesis is that they are different. Besides the traditional Wilcoxon signed-rank test for comparisons of 2 classifiers, the Friedman test [12] is also employed to test the difference between the classification results of multiple methods on multiple datasets. If the null hypothesis that "all methods perform equally" is rejected, we can proceed with a post-hoc test to further analyze the relative performance of the comparing algorithms. Experiments were implemented on a machine with Core i7 3.4 GHz CPU and 16 GB RAM.

5.3 Results and discussion

5.3.1 Classification error rate comparison

The mean and standard deviation (STD) of the classification error rates for all mentioned algorithms are reported in Tables 2-3. From the average results over 30 datasets (see the last rows of Tables 2-3), we have an overall view that OVIG gets the lowest average error rate as well as the smallest variance (see Fig. 1). Between linear methods, undoubtedly first-order methods are outperformed by second-order methods. Between lossless Bayesian methods, the same situation happens where second-order method OVIG achieves higher accuracy than the first-order ONBG. This confirms the greater capability of second-order methods in exploring the underlying structure of training data.

The lower classification error rates together with the smaller variance of OVIG compared with linear methods are due to the lossless property of OVIG as each sample is exploited to update the sufficient statistics. For linear methods like PA, OGD, SCW, many correctly classified samples are discarded without learning from them. Furthermore, the linear

models seem to be not flexible enough for learning real datasets coming from real-world applications.

TABLE 2. THE MEAN AND STANDARD DEVIATION OF ERROR RATES OF THE TWO LOSSLESS ALGORITHMS AND OBHT

Dataset	OVIG	ONBG	OBHT
Abalone	0.4646 ± 4.00E-3	0.4797↑ ± 6.76E-3	0.4797↑ ± 4.38E-3
Appendicitis	0.1868 ± 9.24E-3	**0.1500↓ ± 1.43E-2**	0.1509↓ ± 1.46E-2
Balance	**0.1130 ± 3.22E-3**	0.1328↑ ± 9.78E-3	0.1507↑ ± 1.51E-2
Banana	0.3887 ± 7.62E-3	0.3930↑ ± 3.78E-3	**0.3348↓ ± 1.18E-2**
Biodeg	0.1923 ± 9.21E-3	0.2474↑ ± 9.44E-3	0.1812↓ ± 8.47E-3
Blood	0.2362 ± 7.23E-3	0.2541↑ ± 8.73E-3	0.2428↑ ± 6.79E-3
Chess-Krvk	**0.6152 ± 1.83E-3**	0.7080↑ ± 2.09E-3	0.6934↑ ± 5.97E-3
Contraceptive	**0.4994 ± 7.82E-3**	0.5329↑ ± 8.87E-3	0.5215↑ ± 8.29E-3
Led7digit	0.3300 ± 1.10E-2	0.3404↑ ± 1.08E-2	0.3464↑ ± 1.49E-2
Letter	**0.1371 ± 1.14E-3**	0.3674↑ ± 2.45E-3	0.3611↑ ± 3.65E-3
Libras	**0.3419 ± 1.51E-2**	0.7081↑ ± 7.50E-3	0.5069↑ ± 2.12E-2
Marketing	**0.6915 ± 1.82E-3**	0.6978↑ ± 2.32E-3	0.7049↑ ± 4.75E-3
Nursery	**0.0666 ± 1.42E-3**	0.0953↑ ± 1.59E-3	0.0864↑ ± 1.98E-3
Optdigits	0.1089 ± 2.03E-3	**0.1065↓ ± 2.03E-3**	0.1133↑ ± 2.12E-3
Penbased	**0.0360 ± 8.25E-4**	0.1474↑ ± 1.61E-3	0.1277↑ ± 6.40E-3
Phoneme	0.2159 ± 2.46E-3	0.2397↑ ± 5.02E-3	**0.2126↓ ± 5.20E-3**
Poker	**0.4517 ± 6.78E-5**	0.4989↑ ± 3.17E-5	0.4585↑ ± 9.06E-3
Ring	0.0313 ± 9.25E-4	**0.0215↓ ± 6.03E-4**	0.1168↑ ± 2.63E-3
Satimage	**0.1784 ± 1.79E-3**	0.2063↑ ± 1.92E-3	0.2152↑ ± 3.99E-3
Segment	**0.1174 ± 6.69E-3**	0.2124↑ ± 4.99E-3	0.2208↑ ± 1.06E-2
Skin-NonSkin	0.0164 ± 2.07E-4	0.0760↑ ± 8.23E-5	**0.0065↓ ± 6.85E-4**
Sonar	**0.2490 ± 1.11E-2**	0.3572↑ ± 2.41E-2	0.3231↑ ± 2.45E-2
Spambase	0.1659 ± 4.91E-3	0.1791↑ ± 5.86E-3	0.1568↓ ± 6.12E-3
Texture	0.0321 ± 1.52E-3	0.2487↑ ± 4.11E-3	0.2260↑ ± 6.08E-3
Tic-Tac-Toe	**0.2689 ± 7.52E-3**	0.3105↑ ± 9.74E-3	0.3220↑ ± 1.41E-2
Titanic	0.2284 ± 1.08E-3	0.2299↑ ± 1.92E-3	**0.2278↓ ± 2.68E-3**
Wine-Red	**0.4446 ± 4.97E-3**	0.4460↑ ± 8.86E-3	0.4755↑ ± 1.45E-2
Wine-White	**0.4812 ± 3.09E-3**	0.5057↑ ± 2.01E-3	0.5443↑ ± 6.94E-3
Yeast	0.5427 ± 4.53E-3	0.5590↑ ± 9.43E-3	0.4724↓ ± 1.37E-2
Zoo	**0.1436 ± 1.42E-2**	0.1545↑ ± 2.13E-2	0.2525↑ ± 2.51E-2
Average	**0.2659 ± 6.46E-3**	0.3202↑ ± 8.59E-3	0.3078↑ ± 1.12E-2

↓ or ↑ mean that OVIG is worse or better than the benchmark algorithm, respectively
For each dataset, the best over all results shown in Table 2 and 3 is in bold

TABLE 3. THE MEAN AND STANDARD DEVIATION OF THE CLASSIFICATION ERROR RATES OF THE LINEAR METHODS

Dataset	AROW	SCW	OGD	PA
Abalone	**0.4631↓ ± 5.75E-3**	0.4624↓ ± 7.47E-3	0.5878↑ ± 6.02E-3	0.6019↑ ± 5.97E-3
Appendicitis	0.2066↑ ± 5.08E-3	0.2094↑ ± 1.57E-2	0.2075↑ ± 0.00E0	0.3104↑ ± 3.25E-2
Balance	0.1310↑ ± 7.38E-3	0.1211↑ ± 6.79E-3	0.1746↑ ± 1.60E-2	0.2134↑ ± 1.07E-2
Banana	0.4438↑ ± 1.33E-2	0.4415↑ ± 3.62E-2	0.4565↑ ± 7.54E-3	0.4838↑ ± 7.15E-3
Biodeg	**0.1570↓ ± 3.45E-3**	0.1835↓ ± 6.30E-3	0.2937↑ ± 1.02E-2	0.3570↑ ± 1.11E-2
Blood	**0.2295↓ ± 4.14E-3**	0.2418↑ ± 5.56E-3	0.3929↑ ± 2.00E-2	0.3449↑ ± 8.66E-3
Chess-Krvk	0.8110↑ ± 9.03E-3	0.8271↑ ± 1.37E-2	0.8298↑ ± 3.21E-3	0.8524↑ ± 1.85E-3
Contraceptive	0.5119↑ ± 7.09E-3	0.5136↑ ± 5.82E-3	0.6000↑ ± 1.19E-2	0.6356↑ ± 1.18E-2

Led7digit	**0.3278↓ ± 1.20E-2**	0.3356↑ ± 7.89E-3	0.4128↑ ± 1.28E-2	0.5302↑ ± 2.15E-2
Letter	0.4687↑ ± 2.22E-2	0.4859↑ ± 2.82E-2	0.4372↑ ± 3.16E-3	0.5309↑ ± 1.28E-3
Libras	0.4764↑ ± 3.05E-2	0.4797↑ ± 2.27E-2	0.7753↑ ± 1.75E-2	0.8378↑ ± 1.38E-2
Marketing	0.7148↑ ± 1.12E-2	0.7324↑ ± 1.07E-2	0.7566↑ ± 3.68E-3	0.7840↑ ± 5.55E-3
Nursery	0.2464↑ ± 2.98E-3	0.2647↑ ± 9.15E-3	0.2950↑ ± 1.91E-3	0.3829↑ ± 4.12E-3
Optdigits	0.1248↑ ± 1.43E-2	**0.0632↓ ± 2.29E-3**	0.0981↓ ± 1.89E-3	0.0940↓ ± 2.45E-3
Penbased	0.1851↑ ± 3.58E-2	0.1405↑ ± 2.04E-2	0.1452↑ ± 2.95E-3	0.1773↑ ± 2.55E-3
Phoneme	0.2416↑ ± 2.90E-3	0.2364↑ ± 2.44E-3	0.2359↑ ± 1.67E-3	0.3182↑ ± 3.68E-3
Poker	0.5075↑ ± 8.43E-3	0.5213↑ ± 9.57E-3	0.5404↑ ± 4.70E-4	0.5695↑ ± 4.31E-4
Ring	0.2647↑ ± 2.52E-3	0.2834↑ ± 3.04E-3	0.3526↑ ± 5.57E-3	0.3148↑ ± 3.26E-3
Satimage	0.3379↑ ± 1.32E-2	0.2713↑ ± 1.97E-2	0.3991↑ ± 8.22E-3	0.4652↑ ± 3.65E-3
Segment	0.2326↑ ± 4.44E-2	0.1564↑ ± 1.80E-2	0.3795↑ ± 1.08E-2	0.4513↑ ± 6.62E-3
Skin-NonSkin	0.0917↑ ± 4.48E-4	0.0679↑ ± 1.77E-4	0.1636↑ ± 4.14E-4	0.1553↑ ± 6.72E-4
Sonar	0.2688↑ ± 2.33E-2	0.2899↑ ± 2.52E-2	0.4139↑ ± 2.15E-2	0.4308↑ ± 1.71E-2
Spambase	**0.0960↓ ± 4.82E-3**	0.1120↓ ± 1.92E-3	0.4471↑ ± 8.31E-3	0.3317↑ ± 3.63E-3
Texture	0.0297↓ ± 4.47E-3	**0.0276↓ ± 1.86E-2**	0.2025↑ ± 4.89E-3	0.2587↑ ± 6.70E-3
Tic-Tac-Toe	0.3399↑ ± 6.32E-3	0.3574↑ ± 1.15E-2	0.4023↑ ± 1.73E-2	0.4410↑ ± 1.59E-2
Titanic	0.2359↑ ± 9.62E-3	0.2313↑ ± 2.06E-3	0.2309↑ ± 3.55E-3	0.3507↑ ± 7.56E-3
Wine-Red	0.4584↑ ± 3.79E-2	0.4603↑ ± 2.54E-2	0.6008↑ ± 1.02E-2	0.6278↑ ± 9.90E-3
Wine-White	0.4976↑ ± 8.45E-3	0.5617↑ ± 3.07E-2	0.6529↑ ± 4.26E-3	0.6722↑ ± 6.05E-3
Yeast	0.4569↓ ± 9.09E-3	**0.4478↓ ± 7.48E-3**	0.5763↑ ± 8.54E-3	0.6528↑ ± 1.21E-2
Zoo	0.1564↑ ± 9.70E-3	0.1931↑ ± 9.13E-3	0.2861↑ ± 1.50E-2	0.2931↑ ± 1.99E-2
Average	0.3238↑ ± 1.66E-2	0.3240↑ ± 1.59E-2	0.4116↑ ± 1.00E-2	0.4490↑ ± 1.12E-2

↓ or ↑ mean that OVIG is worse or better than the benchmark algorithm, respectively
For each dataset, the best over all results shown in Table 2 and 3 is in bold

Compared with OBHT, for small datasets e.g. Libras, Sonar, Zoo, clearly OVIG wins. This is because of the sound guarantee by Hoeffding bounds of Hoeffding trees, the base learners of OBHT is only effective for medium and big datasets.

The above observations are further verified statistically by Wilcoxon Sign Rank test between lossless methods and the other methods. The resulting p-values for OVIG and each of the benchmark algorithms are shown in Table A1 in the Appendix, where those ≤ 0.05 indicates a significant difference between compared methods. For each dataset, OVIG wins a benchmark algorithm if p-value ≤ 0.05, and in 10 runs it gets lower error rate more times than the benchmark algorithm. We depict results comparing our proposed method with the benchmark algorithms based on Wilcoxon signed-rank test in Fig. 2. It can be seen that OVIG significantly wins more than any benchmark algorithms. Between the two lossless Bayesian methods (OVIG and ONBG), the second-order Variational Inference based algorithm OVIG has 22 wins, 5 draws and 3 losses compared with the first-order maximum likelihood-based method ONBG.

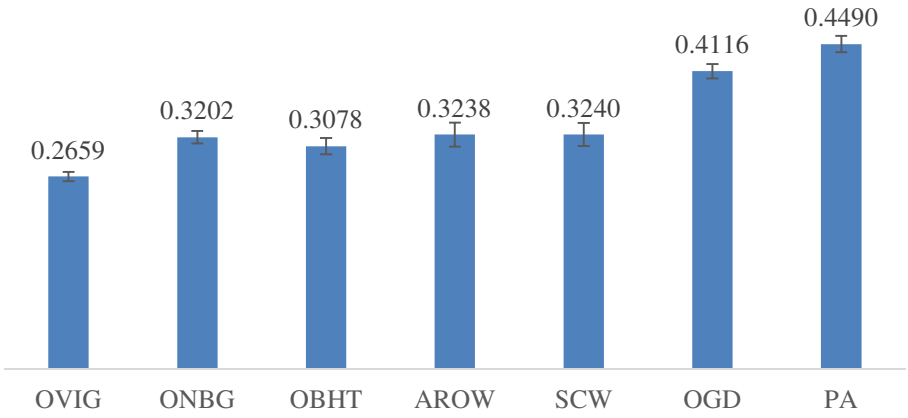

Figure 1. Average mean and standard deviation of the classification error rate of all methods over 30 datasets

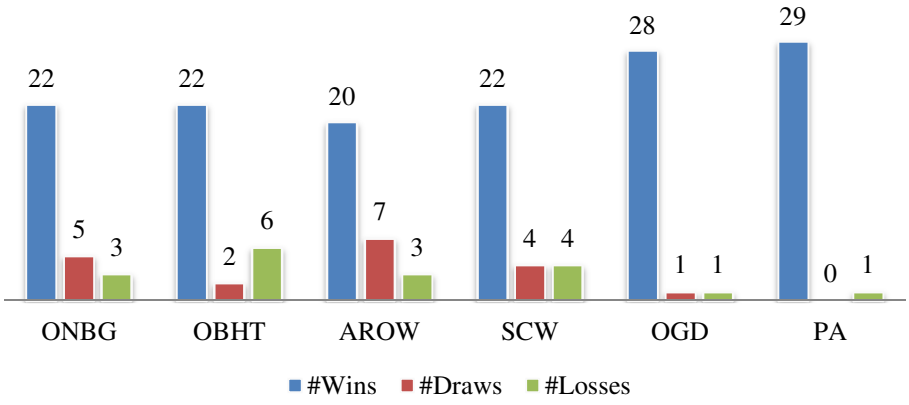

Figure 2. Statistical test results comparing OVIG to the benchmark methods based on Wilcoxon signed-rank test

Although Wilcoxon signed-rank test is very popular in comparison of two classifiers, we also conduct Friedman test [12] with the significance level α set as 0.05 to test the difference between the classification results of multiple methods on multiple datasets. The *p*-value computed by Friedman Test is 9.0528E-11, which is much smaller than 0.05, so the null hypothesis that "all methods perform equally" is clearly rejected. The rankings obtained from Friedman test are depicted in Table 4. On average, OVIG ranked the first with ranks 1.8667 which is far higher than the second best rank 3.1833 of OBHT. AROW and SCW ranked the

third with the same rank 3.5333. The followers were ONBG (rank = 3.5833), OGD (rank = 5.6333) and PA (rank = 6.6667).

TABLE 4. AVERAGE RANKINGS OF ALGORITHMS BASED ON FRIEDMAN TEST

Algorithm	Ranking
OVIG	**1.8667**
ONBG	3.5833
OBHT	3.1833
AROW	3.5333
SCW	3.5333
OGD	5.6333
PA	6.6667

As the null hypothesis of "equal performance among the algorithms" is rejected, we now proceed with a post-hoc test to further analyze their relative performance. As suggested by [16, 17], Hommel test [20] is a suitable post-hoc test when we want to treat OVIG as the control algorithm, and for the case we care about the overall comparison of all 7 algorithms mentioned in the experiments, the Bergmann-Hommel test [1] is one of the best performing pairwise test. Since the post hoc tests differ in the way they adjust the value of the level of significance α to compensate for multiple comparisons, we report adjusted *p*-values in Table 5-6, which take into account that multiple tests are conducted [16, 17]. Adjusted *p*-values can be compared directly with any chosen significance level α and provide more information in a statistical analysis. In our post-hoc tests, the significance level α is set to 0.05. The difference in the performance of two methods is treated as statistically significant if the adjusted value of the *p*-value computed from the respective post-hoc procedure is smaller than 0.05.

For Hommel test, all adjusted *p*-values (see Table 5) are < 0.05, so the Hommel's procedure rejects all null hypotheses of "equal performance", that means the performance of the control algorithm OVIG, given that it has the highest rank from Friedman test, is significantly better than all benchmark algorithms based on Hommel post-hoc test.

TABLE 5. ADJUSTED P-VALUES OF HOMMEL POST-HOC TEST

Algorithm	Adjusted *p*-value
PA	4.55E-17
OGD	7.24E-11
ONBG	4.21E-03
AROW	5.61E-03
SCW	5.61E-03
OBHT	1.82E-02

TABLE 6. ADJUSTED P-VALUES OF BERGMANN-HOMMEL POST-HOC TEST

Hypothesis	Adjusted *p*-value
OVIG vs. PA	1.59E-16
OVIG vs. OGD	2.17E-10
OBHT vs. PA	6.35E-09
SCW vs. PA	2.13E-07
AROW vs. PA	2.13E-07
ONBG vs. PA	2.92E-07
OBHT vs. OGD	1.12E-04
SCW vs. OGD	1.17E-03
AROW vs. OGD	1.17E-03
ONBG vs. OGD	1.66E-03
OVIG vs. ONBG	2.29E-02
OVIG vs. AROW	2.29E-02
OVIG vs. SCW	2.29E-02
OVIG vs. OBHT	9.12E-02
OGD vs. PA	4.48E-01
ONBG vs. OBHT	2.84E+00
OBHT vs. AROW	2.84E+00
OBHT vs. SCW	2.84E+00
ONBG vs. SCW	2.84E+00
ONBG vs. AROW	2.84E+00
AROW vs. SCW	2.84E+00

Regarding pairwise Bergmann-Hommel test, it only rejects the null hypotheses of "equal performance" for the pair of algorithms with the adjusted p-value ≤ 0.05, which are (see Table 6): OVIG vs. ONBG, OVIG vs. AROW, OVIG vs. SCW, OVIG vs. OGD, OVIG vs. PA, ONBG vs. OGD, ONBG vs. PA, OBHT vs. OGD, OBHT vs. PA, AROW vs. OGD, AROW vs. PA, SCW vs. OGD, SCW vs. PA. Therefore, based on Bergmann-Hommel test, given the

rankings from Friedman test (see Table 4), OVIG significantly outperforms AROW, SCW, ONBG, OGD and PA, but its performance is not significantly different from that of OBHT. This confirms the statement of Demsar [12] that the power of the post-hoc test is much greater when all classifiers are compared only to a control classifier and not between themselves.

5.3.2 Time complexity comparison

To compare the running time of OVIG and the benchmark algorithms, we report the average time costs in Fig. 3 (the full-time cost result is presented in Table A2 in the Appendix). Clearly, ONBG is the most time-consuming algorithms. OVIG and OBHT have the same order of time cost magnitude (~E+0), meanwhile, linear methods are the quickest ones (the order ~E-1).

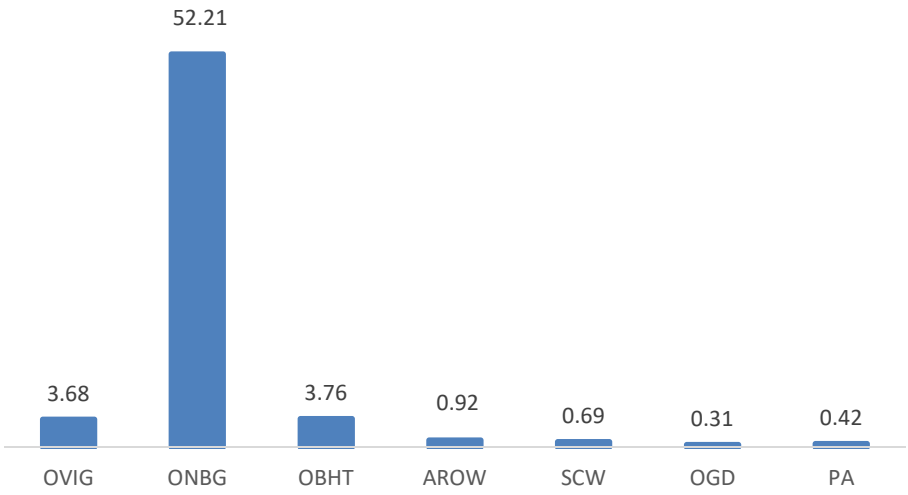

Figure 3. Average runtime of OVIG and the benchmark algorithms

5.3.3 Comparison in noisy setting

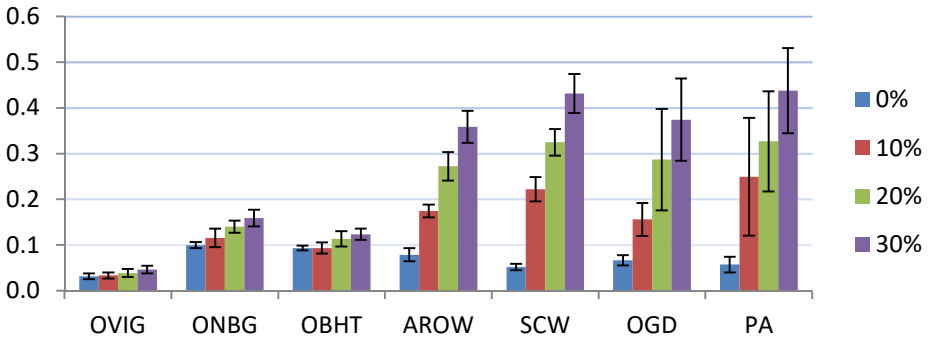

Figure 4. The mean and standard deviation of the classification error rate of all methods in noisy setting for Optdigits dataset

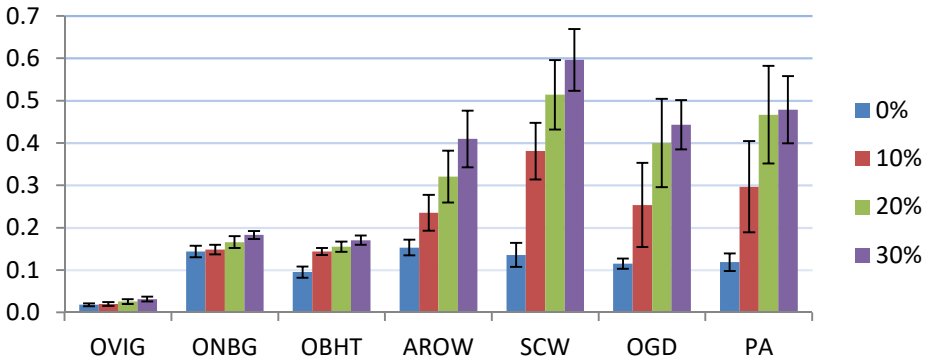

Figure 5. The mean and standard deviation of the classification error rate of all methods in noisy setting for Penbased dataset

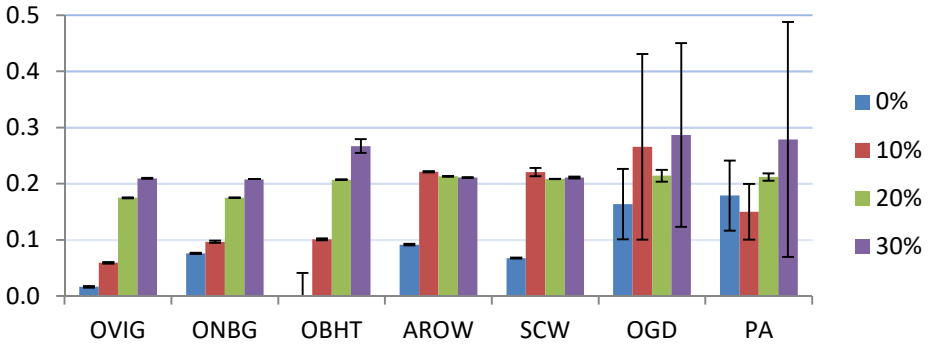

Figure 6. The mean and standard deviation of the classification error rate of all methods in noisy setting for Skin-NonSkin dataset

To evaluate the performance degradation in data with noisy label, we used 10-fold Cross Validation and randomly selected another label to replace the correct label on 0%, 10%, 20% and 30% of the training samples. The learning process is online (in a sequential manner) on each noisy training set while the evaluation is performed on the respective separate clean testing set. The classification error rates are averaged over 10 runs. Due to space limitation, we only exhibit the performance degradation of online methods for three datasets, i.e. binary Skin-NonSkin, multiclass Optdigits, and Penbased, as they are the files that all algorithms perform well on (mean of error rates for every method < 0.2 (see Tables 2-3)). If a method is not good with clean data, it will not be worth analyzing further for noisy data.

It can be seen from Fig. 4-6 that OVIG is more stable when dealing with noise compared to ONBG, the linear methods, and OBHT, as it has the smallest mean and STD of classification error on all 3 mentioned datasets and all noisy setting (10%, 20%, 30%).

For the Skin-NonSkin dataset (see Fig. 6), at 0% label noise (clean data), OBHT ranks the first. However, its performance degrades quicker with noise than that of OVIG and ONBG, leading to a poorer performance than these lossless algorithms at all noise level of 10%, 20%, and 30%.

5.4 Application to movie genre classification

Here, we apply the OVIG and benchmark algorithms to movie genre classification. A collection of movie plot text summaries was downloaded from the Internet Movie Database (*www.imdb.com/interfaces/#plain*) with their genres as the class labels. There are 28 class labels such as Sci-Fi, Crime, Romance, Animation, Comedy and so on. For the multiclass classification purpose, all the data having more than 1 class labels are removed. To extract features from raw texts, we use a string-to-word-vector filter as in MEKA project [32], where string attributes are converted into a set of numeric attributes representing word occurrence information from the text contained in the strings. The dictionary with 1001 words is determined from a validation set of data. This results in a dataset of n = 47845 data, $D = 1001$

features and K= 28 classes. It is very popular that in text classification, datasets are high-dimensional (say > 1000). The curse of dimensionality can make generative methods in common, and OVIG in particular less effective. Therefore, it is helpful to reduce the dimensionality before applying OVIG. There are a number of common dimensionality reduction approaches like principal component analysis (PCA) [3], linear discriminant analysis (LDA) [3], random projections (RP) [15]. For simplicity, here we will use random projections as they are data-independent. In a random projection, a projection $\mathbf{z}_t \in \mathbb{R}^Q$ (called down-space) of vector $\mathbf{x}_t \in \mathbb{R}^D$ (called up-space) can be obtained as follows

$$\mathbf{z}_t = \frac{1}{\sqrt{Q}}\mathbf{x}_t R \tag{22}$$

where $Q = 2\log_2 D$ is the dimension of the down-space, $R = \{r_{ij}\}$ is $D \times Q$ random matrix. We follow [26] to use an ensemble framework of L base classifiers, where l^{th} base classifier is an OVIG model (denoted as $\text{OVIG}^{(l)}$) trained on a projection data stream $\left\{\mathbf{z}_t^{(l)}\right\}_{t=1,2,\dots}$ by a random matrix $R^{(l)}$, $l = 1,2, \dots, L$. At step t, the prediction is made by the sum rule

$$\hat{y}_t = \underset{k\in\{1,2,\cdots,K\}}{\text{argmax}} \frac{1}{L}\sum_{l=1}^{L} p^{(l)}(y = k|\mathbf{x}_t) \tag{23}$$

where $p^{(l)}(y = k|\mathbf{x}_t)$ is the output of $\text{OVIG}^{(l)}$. In our experiment, we use Gaussian random projections, i.e. $r_{ij} \sim \mathcal{N}(0; 1)$, and choose the ensemble size $L = 50$. Generally, the bigger L is, the more accurate the result, but the higher the time complexity (see [26] for more information about random projection based ensembles).

We use prequential evaluation and perform 10 runs on each algorithm with different random permutations of the training data. In each run, the model is trained in a single pass through the data, each arriving datum is tested first then used to update the model. We show the top 1-5 error rates, noting that for a prediction task, a top i error is made if among i class labels having the highest confidences there is no the true class label. The average top 1-5 error rate (top1-5-

ER) with the standard deviation (for top1-ER) over all runs and the average running time (in seconds) are shown in Table 7, where the abbreviation RPVI stands for OVIG under a Random Projection based ensemble. We also plot top 1-5 error rates of all mentioned algorithms in Figure 7.

TABLE 7. THE ERROR RATE AND RUNTIME FOR THE MOVIE GENRE CLASSIFICATION

	Top1-ER	Top2-ER	Top3-ER	Top4-ER	Top5-ER	Time
RPVI	0.7252 ± 8.63E-04	0.5261	0.3892	0.2871	0.2361	701.97
ONBG	0.8593 ± 2.38E-03	0.7421	0.6369	0.5460	0.4731	2445.57
OBHT	0.7254 ± 7.04E-05	0.5431	0.3869	0.2875	0.2436	2142.25
AROW	0.8311 ± 2.55E-03	0.7256	0.6518	0.5943	0.5480	1691.71
SCW	0.8226 ± 2.19E-03	0.7105	0.6330	0.5737	0.5254	1527.38
OGD	0.7683 ± 1.45E-03	0.6443	0.5685	0.5160	0.4755	2.12
PA	0.9720 ± 1.31E-02	0.9490	0.9214	0.8754	0.8359	2.06

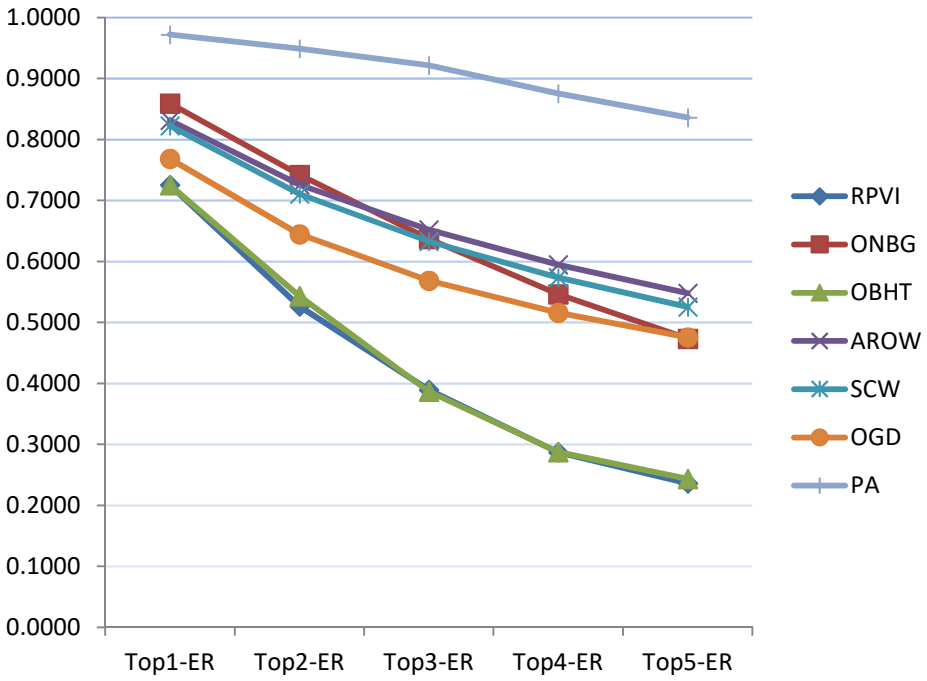

Figure 7. The mean of the top 1-5 classification error rate of all methods for the movie genre classification

It can be seen from Table 7 and Fig. 7 that RPVI and OBHT have the lowest average top 1-5 error rates as well as the lowest standard deviation. While RPVI and OBHT have top 5 error rate around 0.2, that of the other 4 methods including ONBG, AROW, SCW, OGD is ≈ 0.5, and the value for PA is still very high (over 0.8). Regarding the time cost, OGD and PA seem not to be affected by the high dimensionality running impressively fast, RPVI has the moderate run time, meanwhile, OBHT, ONBG, AROW and SCW take a long time on this dataset. This experiment shows a powerful way to allow OVIG to deal with the curse of high dimensionality (say $D >$ 1000). Here the number of feature is $D = 1001,$ and the dimension of the down-space is $Q = 2\log_2(1000) \approx 20$. In the case of $D = 1000000$, $Q = 2\log_2(1000000) \approx 40$, which is still very moderate.

6. Conclusion

We have presented a new lossless online Bayesian method (OVIG) which is guaranteed to converge to the same model as its offline counterpart (VIG). OVIG is a second-order generative model, where training data is exploited effectively to estimate the full distribution of parameters for the predictive model. Besides the superior performance over many well-known methods, OVIG is very stable when dealing with noisy datasets. Compared with first-order maximum-likelihood based ONBG, Variational Inference based OVIG has a more flexible way of updating its predictive model, where all information about the arrived data is encoded in the sufficient statistics, and the predictive model is only updated when a wrong prediction is made. The updating mechanism and the ability to explore the underlying second-order structure of data make OVIG the first online method which simultaneously has the properties of discard-after-learn, conservative, and lossless.

Acknowledgments

Thi Thu Thuy Nguyen is supported by the Australian Government Research Training Program Scholarship to undertake this research.

References

[1] G. Bergmann and G. Hommel, Improvements of general multiple test procedures for redundant systems of hypotheses, In P. Bauer, G. Hommel, and E. Sonnemann, editors, Multiple Hypotheses Testing, pp. 100–115. Springer, Berlin, 1988.

[2] A. Bifet, G. Holmes, B. Pfahringer, P. Kranen, H. Kremer, T. Jansen, and T. Seidl, MOA: Massive Online Analysis, a Framework for Stream Classification and Clustering, In Journal of Machine Learning Research Workshop and Conference Proceedings, Vol.11: Workshop on Applications of Pattern Analysis, 2010.

[3] C.M. Bishop, Pattern recognition and machine learning, Springer, 2006.

[4] L. Breiman, J. Friedman, R. Olshen, and C.J. Stone, Classification and Regression Trees, Wadsworth and Brooks, Monterey, CA, 1984.

[5] R. Caruana, and A. Niculescu-Mizil, An empirical comparison of supervised learning algorithms, In ICML, 2006, pp. 161–168.

[6] N. Cesa-Bianchi, A. Conconi, C. Gentile, A Second-order Perceptron Algorithm, SIAM Journal on Computing 34 (2005), pp. 640–668.

[7] K. Crammer, O. Dekel, S. Shalev-Shwartz, and Y. Singer, Online Passive Aggressive Algorithms, In NIPS, 2003.

[8] K. Crammer, O. Dekel, J. Keshet, S. Shalev-Shwartz, and Y. Singer, Online Passive-Aggressive Algorithms, Journal of Machine Learning Research 7 (2006), pp. 551–585.

[9] K. Crammer, A. Kulesza, and M. Dredze, Adaptive Regularization of Weight Vectors, In NIPS, 2009, pp. 414–422.

[10] K. Crammer, A. Kulesza, and M. Dredze, Adaptive Regularization of Weight Vectors, Machine Learning 91 (2013), pp. 155-187.

[11] K. Crammer, and Y. Singer, Ultraconservative Online Algorithms for Multiclass Problems, Journal of Machine Learning Research 3 (2003), pp. 951–991.

[12] J. Demsar, Statistical comparisons of classifiers over multiple datasets, Journal of Machine Learning Research 7 (2006), pp. 1–30.

[13] P. Domingos, and G. Hulten, Mining High-speed Data Streams, In KDD, 2000, pp. 71–80.

[14] M. Dredze, K. Crammer, and F. Pereira, Confidence-weighted linear classification, In ICML, 2008, pp. 264-271.

[15] X.Z. Fern and C.E. Brodley, Random projection for high dimensional data clustering: A cluster ensemble approach, In ICML, 2003, pp. 186–193.

[16] S. García, A. Fernández, J. Luengo, and F. Herrera, Advanced nonparametric tests for multiple comparisons in the design of experiments in computational intelligence and data mining: Experimental analysis of power, Information Sciences 180(10), 2010, pp. 2044–2064.

[17] S. García and F. Herrera. An extension on "Statistical comparisons of classifiers over multiple data sets" for all pairwise comparisons, Journal of Machine Learning Research 9 (2008), pp. 2677–2694.

[18] F.A. Gonzalez, and E. Romero, Biomedical Image Analysis and Machine Learning Technologies: Applications and Techniques, Information Science Reference, Hershey, PA, 2009.

[19] S.C.H. Hoi, J. Wang, and P. Zhao, LIBOL: A Library for Online Learning Algorithms, Journal of Machine Learning Research 15 (2014), pp. 495-499.

[20] G. Hommel, A stagewise rejective multiple test procedure based on a modified Bonferroni test, Biometrika 75 (1988), pp. 383–386.

[21] S. Jaiyen, C. Lursinsap, and S. Phimoltares, A Very Fast Neural Learning for Classification Using Only New Incoming Datum, IEEE Transactions on Neural Networks 21(3), 2010, pp. 381-392.

[22] P. Junsawang, S. Phimoltares, and C. Lursinsap, A fast learning method for streaming and randomly ordered multi-class data chunks by using one-pass-throw-away class-wise learning concept, Expert Systems with Applications 63(2016), pp. 249-266.

[23] W. Li, and A.W. Moore, A Machine Learning Approach for Efficient Traffic Classification, In MASCOTS, 2007.

[24] B. Li, P. Zhao, S.C.H. Hoi, and V. Gopalkrishnan, Passive Aggressive Mean Reversion Strategy for Portfolio Selection, Machine Learning 87(2), 2012, pp. 221–258.

[25] M. Lichman, UCI Machine Learning Repository [http://archive.ics.uci.edu/ml], Irvine, CA: University of California, School of Information and Computer Science, 2013.

[26] T.T. Nguyen, T.T.T. Nguyen, X.C. Pham, A.W.C. Liew, and J.C. Bezdek, An Ensemble-based Online Learning Algorithm for Streaming Data, http://arxiv.org, 2017.

[27] T.T.T. Nguyen, A.W.C. Liew, T.T. Nguyen, and S. Wang, A Novel Bayesian Framework for Online Imbalanced Learning, In DICTA, 2017.

[28] T.T. Nguyen, T.T.T. Nguyen, X.C. Pham, and A.W.-C. Liew, A novel combining classifier method based on Variational Inference, Pattern Recognition 49 (2016), pp. 198-212.

[29] N. Oza, and S. Russell, Online Bagging and Boosting, In Proceedings of Artificial Intelligence and Statistics, 2001, pp. 105–112.

[30] X.C. Pham, M.T. Dang, S.V. Dinh, S. Hoang, T.T. Nguyen, A.W.-C. Liew, An online machine learning method based on Random Projection and Hoeffding Tree classifier, In DICTA, 2017.

[31] J.R. Quinlan, C4.5: programs for machine learning. Morgan Kaufmann, San Francisco, 1993.

[32] J. Read, P. Reutemann, B. Pfahringer, G. Holmes, MEKA: A Multi-label/Multi-target Extension to WEKA, Journal of Machine Learning Research 17(21), 2016, pp. 1−5.

[33] F. Rosenblatt, The Perceptron: A Probabilistic Model for Information Storage and Organization in the Brain, Psychological Review 65(6), 1958, pp. 386–408.

[34] A. Saffari, C. Leistner, J. Santner, M. Godec, and H. Bischof, On-line Random Forests, In Computer Vision Workshops (ICCV), 2009, pp. 1393–1400.

[35] M. Tennant, F. Stahl, O. Rana and J.B. Gomes, Scalable Real-time Classification of Data Streams with Concept Drift, Future Generation Computer Systems 75 (2017), pp. 187-199.

[36] M. Thakong, S. Phimoltares, S. Jaiyen, and C. Lursinsap, Fast Learning and Testing for Imbalanced Multi-Class Changes in Streaming Data by Dynamic Multi-Stratum Network, IEEE Access 5(2017), pp. 10633-10648.

[37] P.E. Utgoff, N.C. Berkman, and J.A. Clouse, Decision Tree Induction Based on Efficient Tree Restructuring, Machine Learning 29 (1997), pp. 5-44.

[38] J. Wang, P. Zhao, and S.C.H. Hoi, Exact Soft Confidence-weighted Learning, In ICML, 2012, pp. 107–114.

[39] L. Yang, R. Jin, and J. Ye, Online Learning by Ellipsoid Method, In ICML, 2009, pp. 1153–1160.

[40] J. Zinkevich, Online Convex Programming and Generalized Infinitesimal Gradient Ascent, In ICML, 2003, pp. 928–936.

Appendix

TABLE A1. P-VALUE OF WILCOXON SIGN RANK TEST BETWEEN OVIG AND THE BENCHMARK ALGORITHMS

	ONBG		OBHT		AROW		SCW		OGD		PA	
Dataset	p-value	Reject?	p-value	Reject?	p-value	Reject?	p-value	Reject?	p-value	Reject?	p-value	Reject?
Abalone	0.002	Y	0.002	Y	0.186	N	0.496	N	0.002	Y	0.002	Y
Appendicitis	0.002	Y	0.002	Y	0.004	Y	0.012	Y	0.002	Y	0.002	Y
Balance	0.004	Y	0.002	Y	0.002	Y	0.008	Y	0.002	Y	0.002	Y
Banana	0.131	N	0.002	Y	0.002	Y	0.002	Y	0.002	Y	0.002	Y
Biodeg	0.002	Y	0.002	Y	0.002	Y	0.037	Y	0.002	Y	0.002	Y
Blood	0.002	Y	0.027	Y	0.051	N	0.082	N	0.002	Y	0.002	Y
Chess-Krvk	0.002	Y	0.002	Y	0.002	Y	0.002	Y	0.002	Y	0.002	Y
Contraceptive	0.002	Y	0.002	Y	0.002	Y	0.002	Y	0.002	Y	0.002	Y
Led7digit	0.098	N	0.012	Y	0.904	N	0.357	N	0.002	Y	0.002	Y
Letter	0.002	Y	0.002	Y	0.002	Y	0.002	Y	0.002	Y	0.002	Y
Libras	0.002	Y	0.002	Y	0.002	Y	0.002	Y	0.002	Y	0.002	Y
Marketing	0.002	Y	0.002	Y	0.002	Y	0.002	Y	0.002	Y	0.002	Y
Nursery	0.002	Y	0.002	Y	0.002	Y	0.002	Y	0.002	Y	0.002	Y
Optdigits	0.049	Y	0.002	Y	0.004	Y	0.002	Y	0.002	Y	0.002	Y
Penbased	0.002	Y	0.002	Y	0.002	Y	0.002	Y	0.002	Y	0.002	Y
Phoneme	0.002	Y	0.064	N	0.002	Y	0.002	Y	0.002	Y	0.002	Y
Poker	0.002	Y	0.049	Y	0.002	Y	0.002	Y	0.002	Y	0.002	Y
Ring	0.002	Y	0.002	Y	0.002	Y	0.002	Y	0.002	Y	0.002	Y
Satimage	0.002	Y	0.002	Y	0.002	Y	0.002	Y	0.002	Y	0.002	Y
Segment	0.002	Y	0.002	Y	0.002	Y	0.002	Y	0.002	Y	0.002	Y
Skin-NonSkin	0.002	Y	0.002	Y	0.002	Y	0.002	Y	0.002	Y	0.002	Y
Sonar	0.002	Y	0.002	Y	0.051	N	0.008	Y	0.002	Y	0.002	Y
Spambase	0.002	Y	0.002	Y	0.002	Y	0.002	Y	0.002	Y	0.002	Y
Texture	0.002	Y	0.002	Y	0.084	N	0.084	N	0.002	Y	0.002	Y
Tic-Tac-Toe	0.002	Y	0.002	Y	0.002	Y	0.002	Y	0.002	Y	0.002	Y
Titanic	0.055	N	0.678	N	0.041	Y	0.006	Y	0.063	N	0.002	Y
Wine-Red	0.566	N	0.002	Y	0.492	N	0.049	Y	0.002	Y	0.002	Y
Wine-White	0.002	Y	0.002	Y	0.002	Y	0.002	Y	0.002	Y	0.002	Y
Yeast	0.002	Y	0.002	Y	0.002	Y	0.002	Y	0.002	Y	0.002	Y
Zoo	0.109	N	0.002	Y	0.078	N	0.002	Y	0.002	Y	0.002	Y
		Y:25		Y: 28		Y: 23		Y: 26		Y: 29		Y: 30
		N: 5		N: 2		N: 7		N: 4		N: 1		N: 0

**'Reject?' column means whether the null hypothesis is rejected (Y) or not (N)*

TABLE A2. THE RUN TIME IN SECONDS OF OVIG AND THE BENCHMARK ALGORITHMS

Dataset	OVIG	ONBG	OBHT	AROW	SCW	OGD	PA
Abalone	0.28	1.39	0.43	0.08	0.13	0.03	0.05
Appendicitis	0.00	0.02	0.03	0.00	0.00	0.01	0.00
Balance	0.02	0.12	0.02	0.01	0.01	0.00	0.00
Banana	0.22	0.48	0.13	0.07	0.07	0.03	0.03
Biodeg	0.14	1.07	0.17	0.01	0.01	0.01	0.00
Blood	0.03	0.10	0.02	0.01	0.01	0.00	0.00
Chess-Krvk	3.23	38.89	3.41	0.52	0.44	0.22	0.33
Contraceptive	0.11	0.54	0.25	0.04	0.02	0.01	0.02
Led7digit	0.04	0.42	0.06	0.01	0.01	0.00	0.00
Letter	1.83	107.67	8.28	0.44	0.28	0.15	0.21
Libras	0.28	5.76	0.52	0.05	0.02	0.00	0.00
Marketing	1.01	10.10	1.17	0.14	0.10	0.05	0.07
Nursery	0.52	6.55	0.65	0.21	0.19	0.08	0.14
Optdigits	1.08	45.05	3.14	0.28	0.13	0.03	0.05
Penbased	0.44	23.38	1.73	0.18	0.12	0.07	0.09
Phoneme	0.19	1.09	0.39	0.08	0.05	0.04	0.02
Poker	93.25	1220.00	81.93	21.60	16.65	7.18	10.49
Ring	0.18	3.99	0.47	0.11	0.08	0.04	0.03
Satimage	0.77	17.64	1.48	0.26	0.18	0.05	0.07
Segment	0.16	4.69	0.62	0.03	0.03	0.02	0.02
Skin-NonSkin	4.26	28.20	3.01	3.12	1.84	1.18	0.84
Sonar	0.07	0.47	0.09	0.00	0.00	0.00	0.00
Spambase	0.90	6.47	0.72	0.08	0.04	0.04	0.02
Texture	0.40	32.85	2.90	0.18	0.08	0.04	0.05
Tic-Tac-Toe	0.05	0.23	0.03	0.01	0.01	0.01	0.00
Titanic	0.08	0.25	0.02	0.03	0.02	0.01	0.01
Wine-Red	0.16	1.63	0.36	0.03	0.02	0.01	0.02
Wine-White	0.51	5.68	0.58	0.11	0.07	0.04	0.05
Yeast	0.15	1.37	0.08	0.03	0.04	0.01	0.01
Zoo	0.01	0.16	0.03	0.00	0.00	0.00	0.00
Average	3.68	52.21	3.76	0.92	0.69	0.31	0.42