2016 IEEE 16th International Conference on Data Mining

One-pass Logistic Regression for Label-drift and Large-scale Classification on Distributed Systems

Vu Nguyen, Tu Dinh Nguyen, Trung Le, Svetha Venkatesh and Dinh Phung
Center for Pattern Recognition and Data Analytics, Deakin University, Australia
{v.nguyen,tu.nguyen,trung.l,svetha.venkatesh,dinh.phung}@deakin.edu.au

***Abstract*—Logistic regression (LR) for classification is the workhorse in industry, where a set of predefined classes is required. The model, however, fails to work in the case where the class labels are not known in advance, a problem we term *label-drift classification*. Label-drift classification problem naturally occurs in many applications, especially in the context of streaming settings where the incoming data may contain samples categorized with new classes that have not been previously seen. Additionally, in the wave of big data, traditional LR methods may fail due to their expense of running time. In this paper, we introduce a novel variant of LR, namely *one-pass logistic regression* (OLR) to offer a principled treatment for label-drift and large-scale classifications. To handle large-scale classification for big data, we further extend our OLR to a distributed setting for parallelization, termed *sparkling OLR* (Spark-OLR). We demonstrate the scalability of our proposed methods on large-scale datasets with more than one hundred million data points. The experimental results show that the predictive performances of our methods are comparable or better than those of state-of-the-art baselines whilst the execution time is much faster at an order of magnitude. In addition, the OLR and Spark-OLR are invariant to data shuffling and have no hyperparameter to tune that significantly benefits data practitioners and overcomes the curse of big data cross-validation to select optimal hyperparameters.**

***Index Terms*—Logistic regression, large-scale classification, label-drift, Apache Spark, distributed system**

I. Introduction

Logistic regression (LR) [1] is one of the most widely-used machine learning methods for classification. Traditionally, the LR has been solved via maximum likelihood estimation or stochastic gradient descent. In a Bayesian context, it is difficult to compute the posterior since the prior distribution (e.g., Gaussian distribution) is not conjugate to the logistic likelihood function. A common approach is to approximate this posterior distribution using Laplacian approximation [2]. More recently, Polya-Gamma augmentation and Gibbs sampling have been developed and shown to be effective as an alternative approach to infer the posterior of LR [3]. The idea was originally developed for binary classification, but could be straightforwardly extended for multiclass classification.

Multiclass classification is the workhorse in the literature of classification, where a set of predefined classes is required. These models, however, fail to work in the case where the class labels are not known in advance, a problem we term *label-drift classification*, in a similar spirit of the so-called concept drift problem known in the literature [4], [5], [6]. Label-drift classification problem naturally occurs in many applications, especially in the context of streaming and online settings where the incoming data may contain samples categorized with new classes [7], [8]. Despite its significance for the robustness and sustainability of the classification system, this problem has received little research attention.

Under the streaming setting, the amount of data grows super-abundantly poses the so-called *big data* challenge. In such large-scale scenario, it could be prohibitively expensive to use a conventional LR model, hence the logistic regression models should be rethought. Moreover, the ever growing collection of data nowadays is far beyond the capacity that a single machine can handle. Therefore, there is a need for frameworks that support parallel and distributed data processing on multiple machines, both for research communities as well as industry demands.

In this paper, we introduce a novel variant of LR model, entitled *one-pass logistic regression* (OLR) for large-scale and label-drift classifications. We leverage Polya-Gamma augmentation approach [3] by deriving the sufficient statistics update for its maximum a posteriori (MAP) estimation. Manipulating these sufficient statistics allows our proposed method to efficiently perform the label-drift classification under an online setting without retraining the model from scratch. To handle large-scale classification, we further introduce a distributed model, termed *sparkling OLR* (Spark-OLR) that can parallelize to compute its sufficient statistics. Since there is no distributed implementation for the logistic regression with Polya-Gamma augmentation exists, our work provides the first implementation on Apache Spark for massive datasets at real-world and industry scale. Our implementations are also released to the research community and public use.

For data practitioners, our model has an additional advantage where it has no hyperparameter to tune, thus there is no need to perform an expensive cross-validation to choose optimal hyperparameters. We conduct extensive experiments to show that our OLR obtains a significant speedup in the training and testing phases, compared with the state-of-the-art baselines, and our Spark-OLR is substantially faster than the existing counterparts implemented in Spark. At the same time, the classification performances of our models are better than or comparable with those of the baselines. To further the evaluation, we measure the inherent trade-off between speed and accuracy by proposing *quadrant visualization* and *quadrant score*, on which our proposed model outperforms other methods on all datasets.

2374-8486/16 $31.00 © 2016 IEEE
DOI 10.1109/ICDM.2016.69

II. Bayesian Logistic Regression using Polya-Gamma Approach

Logistic regression is a popular choice for classification using a form $p(y=1 \mid \boldsymbol{x}) = \frac{1}{1+\exp(-\mathbf{w}^T\boldsymbol{x})}$ where $\boldsymbol{x} \in \mathcal{R}^D$ is the feature vector and $y \in \{0,1\}$ is the label, parameterized by the coefficient vector $\mathbf{w} \in \mathcal{R}^D$. Under the frequentist view, we can use maximum likelihood estimation (MLE) by conjugate gradient descent to train a LR model in batch setting and stochastic gradient descent (SGD) for online setting.

Under the Bayesian view, there are two noticeable Bayesian logistic regression approaches using Laplacian approximation and Polya-Gamma data augmentation. The Bayesian logistic regression (BLR) approaches utilize Bayesian statistics, the uncertainty about the unknown parameters is quantified using probability so that the unknown parameters are regarded as random variables. Specifically, we define a prior distribution $p(\mathbf{w} \mid \boldsymbol{\mu}_0, \boldsymbol{\Sigma}_0) = \mathcal{N}(\boldsymbol{\mu}_0, \boldsymbol{\Sigma}_0)$. Then, the posterior of $\mathbf{w}$ is

$$\log p(\mathbf{w}) \propto -\frac{1}{2}(\mathbf{w}-\boldsymbol{\mu}_0)^T \boldsymbol{\Sigma}_0^{-1}(\mathbf{w}-\boldsymbol{\mu}_0) + \sum_{i=1}^{N}\left[y_i \log \sigma_i + (1-y_i)(1-\log \sigma_i)\right] \quad (1)$$

Since the posterior inference for BLR is not in closed-form update, Polson *et al* [3] utilize a new class of Polya-Gamma distribution to propose the augmentation strategy for closed-form Bayesian inference. We utilize the interesting property of sigmoid function [3]: $\frac{\exp(\mathbf{w}^T\boldsymbol{x})^y}{1+\exp(\mathbf{w}^T\boldsymbol{x})} =$

$$\frac{1}{2}\exp\left(\frac{1}{\sqrt{\kappa}}\right)\int_0^\infty \exp\left(-\frac{\lambda}{2}\left[\mathbf{w}^T\boldsymbol{x}\right]^2\right)p(\lambda)\,d\lambda \quad (2)$$

where $\kappa = y - \frac{1}{2}$ and $\lambda \sim PG(b,c)$ follows the Polya-Gamma distribution. We obtain the posterior of $\mathbf{w}$ by plugging Eq. 2 into Eq. 1, $\log p(\mathbf{w}) \propto$

$$-\frac{1}{2}(\mathbf{w}-\boldsymbol{\mu}_0)^T\boldsymbol{\Sigma}_0^{-1}(\mathbf{w}-\boldsymbol{\mu}_0) + \sum_{i=1}^{N}\left(\kappa_i \mathbf{w}^T\boldsymbol{x}_i - \int \frac{\lambda_i}{2}\left[\mathbf{w}^T\boldsymbol{x}_i\right]^2 \log p(\lambda_i)\,d\lambda_i\right).$$

Let $\boldsymbol{\lambda} = \{\lambda_1, ..., \lambda_N\}$, the posterior $\log p(\mathbf{w})$ in Eq. 1 can be viewed as the marginal from a joint posterior with the auxiliary variables as $\log p(\mathbf{w}, \boldsymbol{\lambda}) \propto$

$$\sum_{i=1}^{N}\left(\kappa_i \mathbf{w}^T\boldsymbol{x}_i - \frac{\lambda_i}{2}\left[\mathbf{w}^T\boldsymbol{x}_i\right]^2\right) - \frac{1}{2}(\mathbf{w}-\boldsymbol{\mu}_0)^T\boldsymbol{\Sigma}_0^{-1}(\mathbf{w}-\boldsymbol{\mu}_0).$$

Gibbs sampling can be performed on this joint posterior by sequentially sampling $\mathbf{w}$ given λ_i and vice versa. Completing the square of the above equation, we get:

$$p(\lambda_i \mid \boldsymbol{x}_i, \mathbf{w}) \sim PG\left(1, \boldsymbol{x}_i^T\mathbf{w}\right) \quad ,\forall i = 1...N \quad (3)$$

$$p(\mathbf{w} \mid \boldsymbol{y}, \boldsymbol{\lambda}) \sim \mathcal{N}(\boldsymbol{\mu}_N, \boldsymbol{\Sigma}_N) \quad (4)$$

where $\boldsymbol{X} = [\boldsymbol{x}_1, ..., \boldsymbol{x}_N]^T$, $\boldsymbol{\Sigma}_N = \left(\boldsymbol{X}^T \text{diag}(\boldsymbol{\lambda})\boldsymbol{X} + \boldsymbol{\Sigma}_0^{-1}\right)^{-1}$, $\boldsymbol{\mu}_N = \boldsymbol{\Sigma}_N\left(\boldsymbol{X}^T\kappa + \boldsymbol{\Sigma}_0^{-1}\boldsymbol{\mu}_0\right)$ and $\kappa = [y_1 - 1/2, ..., y_N - 1/2]^T$.

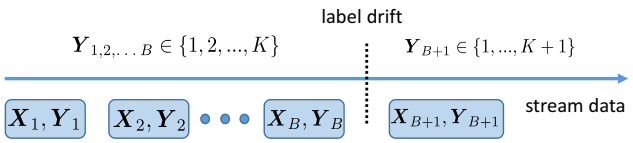

Figure 1: Label drift illustration. Let $\boldsymbol{X}_b$ be the feature blocks, $\boldsymbol{Y}_b$ be the label block, the label drift occurs when the new label $K+1$ arises in the new block $B+1$.

III. Label-drift Classification

In classification, the number of classes is usually known and fixed. However, in some situations especially streaming data, the number of classes is unknown and changed over time. For example, we first train the model with two classes of *dog* and *cat*, then we store the model and discard the data. Later on we receive the additional data for class of *monkey*. We want to update the model incrementally using the new data without retraining from scratch, in a way that the model can effectively capture the new class of *monkey,* along with the old classes of *dog* and *cat*. We term this setting *label-drift classification*, in a similar spirit of the so-called concept-drift problem in the literature [9], [10], [11]. The label-drift classification refers to an online classification system that the label variable changes over time in unforeseen ways (e.g., previously training data points with K classes, and later on seeing a data point with class $K+1$) as in Fig. 1.

The label-drift poses challenges to the existing classification techniques for the two main reasons. First, it requires classifiers to handle the streaming data (e.g., online learning). Second, the algorithm needs to be invariant to the data ordering or permutation. For example, due to the simplicity of naive Bayes which does not depend on the data ordering, naive Bayes will get the same accuracy even if we shuffle the data order at any kind. In contrast, shuffling data is an important factor to prevent stochastic gradient descent (SGD) from the bias optimization problem [12]. Therefore, the label-drift brings problems for SGD due to the data ordering. To the best of our knowledge, there is no existing work of logistic regression for effectively handling label-drift problem.

Although the label-drift can be seen as a special case of the concept-drift, the former is still different from the latter that focuses on detecting and handling the changes in feature-label joint distribution $p(\boldsymbol{x}, y)$ [9], [10], [11], [6].

IV. One-pass Logistic Regression

We propose a novel variant of logistic regression (LR) for label-drift setting. Our model roots in the Polya-Gamma augmentation for logistic regression presented in Sec. II. Our intuition is that, instead of sampling the entire distributions of the parameters, we exploit the Polya-Gamma (PG) auxiliary approach to compute the MAP estimation of the parameters. From the MAP estimation, we can obtain a handy update for sufficient statistics, resulting in a simple and fast computation [13], [8]. In particular, we later demonstrate in the experiments

Algorithm 1 One-pass logistic regression.

Input: $y_{1:N}, \boldsymbol{x}_{1:N}$

$\boldsymbol{P} = \mathbf{I}$, $\boldsymbol{Q}_k = \mathbf{0}^{D\times 1}$, $\mathbf{w}_k = \mathbf{1}^{D\times 1} : \forall k = 1, 2, ..., K$

for each data point $i = 1, ..., N$ **do**

  $\lambda_i = \frac{1}{2\times \mathbf{w}_{y_i}^T x_i} \tanh\left(\frac{\mathbf{w}_{y_i}^T x_i}{2}\right)$

  $\boldsymbol{P} = \boldsymbol{P} + \lambda_i \boldsymbol{x}_i \boldsymbol{x}_i^T$

end for

for each class $k = 1, ..., K$ **do**

  $\boldsymbol{Q}_k = \sum_{\forall i|y_i=k} \boldsymbol{x}_i - \sum_{\forall j|y_j\neq k} \boldsymbol{x}_j$

end for

$\mathbf{w}_k = \boldsymbol{P}\backslash\boldsymbol{Q}_k, \forall k = 1, 2, ..., K$

Output: $\mathbf{w}_{1:K}, \boldsymbol{P}, \boldsymbol{Q}_{1:K}$

that MAP estimation improves at least 100 times faster in speed without any hurt in accuracy. Importantly, the MAP estimation via manipulating the sufficient statistic matrices enables us handle the label-drift effectively. Our algorithm iterates through all data once to ensure the property of online label-drift and for computational efficiency. Thus, our proposed approach is termed *one-pass logistic regression* (OLR).

A. MAP Estimation of $\mathbf{w}$ and Auxiliary Variable λ_i

Recall from Sec. II that the BLR with Polya-Gamma (PG) distribution estimates $\mathbf{w}$ by iteratively sampling the auxiliary variable λ_i and $\mathbf{w}$ as in Eqs. (3,4). We utilize the property of PG distribution that the expectation can be calculated efficiently as follows [3].

Lemma 1. *Let* $\lambda_i \sim PG\left(b = 1, c = \boldsymbol{x}_i^T \mathbf{w}\right)$, *we have*

$$\mathbb{E}\left[\lambda_i\right] = \frac{1}{2\mathbf{w}^T \boldsymbol{x}_i} \tanh\left(\frac{\mathbf{w}^T \boldsymbol{x}_i}{2}\right). \tag{5}$$

Similarly, we assign $\mathbf{w}$ to the MAP (cf. Eq. (4)) as $\mathbf{w} = \boldsymbol{\mu}_N = \boldsymbol{\Sigma}_N \boldsymbol{X}^T \boldsymbol{Y}$ [3] where the prior distribution is $p\left(\mathbf{w} \mid \boldsymbol{\mu}_0, \boldsymbol{\Sigma}_0\right) \sim \mathcal{N}\left(0, \mathbf{I}\right)$. For estimating $\mathbf{w}$, we can compute $\boldsymbol{\Sigma}_N^{-1} = \boldsymbol{X}^T \text{diag}\left(\boldsymbol{\lambda}\right) \boldsymbol{X} + \boldsymbol{\Sigma}_0^{-1}$ directly from the data, but not $\boldsymbol{\Sigma}_N$. To avoid computational inefficiency of matrix inversion (the cost is $\mathcal{O}\left(d^3\right)$ and the inversion accuracy is not always high), we compute $\mathbf{w}$ by solving the system of linear equations (e.g., $\boldsymbol{A}x = \boldsymbol{b}$) then the solution is

$$\mathbf{w} = \left(\boldsymbol{X}^T \text{diag}\left(\boldsymbol{\lambda}\right) \boldsymbol{X} + \mathbf{I}\right) \backslash \boldsymbol{X}^T \boldsymbol{Y} \tag{6}$$

where the backslash ($\backslash$) indicates for solving the linear system of equations which the computational cost is generally at $\mathcal{O}\left(d^{2.376}\right)$ [14].

One can iteratively sample λ_i and $\mathbf{w}$ following the above approximation. However, in practice we find that a single iteration gains sufficiently good performance. In addition, a single iteration is beneficial for working on big data since it can save computational time cost and enable the label-drift learning without retraining from the scratch.

Algorithm 2 OLR learning with label-drift.

Input: $\boldsymbol{P}^{\text{old}}, \boldsymbol{Q}_{1:K}^{\text{old}}, \boldsymbol{Q}_0^{\text{old}}, \boldsymbol{x}_{\text{new}}, y_{\text{new}} = K + 1$

$\mathbf{w}_{K+1} = \mathbf{1}^{D\times 1}$

$\lambda_{\text{new}} = \frac{1}{2\times \mathbf{w}_{y_{\text{new}}}^T x^{\text{new}}} \tanh\left(\frac{\mathbf{w}_{y_{\text{new}}}^T x^{\text{new}}}{2}\right)$

$\boldsymbol{P}^{\text{new}} = \boldsymbol{P}^{\text{old}} + \lambda_{\text{new}} \boldsymbol{x}_{\text{new}} \boldsymbol{x}_{\text{new}}^T$

for $k = 1, ..., K$ **do**

  $\boldsymbol{Q}_k^{\text{new}} = \boldsymbol{Q}_k^{\text{old}} - \boldsymbol{x}_{\text{new}}$; $\mathbf{w}_k = \boldsymbol{P}^{\text{new}}\backslash\boldsymbol{Q}_k^{\text{new}}$

end for

$\boldsymbol{Q}_{K+1}^{\text{new}} = \boldsymbol{Q}_0^{\text{old}} + \boldsymbol{x}_{\text{new}}$; $\mathbf{w}_{K+1} = \boldsymbol{P}^{\text{new}}\backslash\boldsymbol{Q}_{K+1}^{\text{new}}$

$\boldsymbol{Q}_0^{\text{new}} = \boldsymbol{Q}_0^{\text{old}} - \boldsymbol{x}_{\text{new}}$

Output: $\mathbf{w}_{1:K+1}, \boldsymbol{P}^{\text{new}}, \boldsymbol{Q}_{1:K+1}^{\text{new}}, \boldsymbol{Q}_0^{\text{new}}$

B. OLR Algorithms in Updating Sufficient Statistic Matrices

From the MAP estimation described previously, we further present algorithms to learn OLR for multiclass classification and label-drift classification by efficiently updating the sufficient statistics matrices P and Q.

1) OLR for multiclass classification: The classification procedure of OLR consists of computing λ_i independently for each data point i, and estimating $\mathbf{w}$ following Sec. IV-A. More specifically, we need to compute:

$$\boldsymbol{P} = \boldsymbol{X}^T \text{diag}\left(\boldsymbol{\lambda}\right) \boldsymbol{X} + \mathbf{I} = \sum_{i=1}^{N} \lambda_i \boldsymbol{x}_i \boldsymbol{x}_i^T + \mathbf{I} \tag{7}$$

$$\boldsymbol{Q}_k = \sum_{\forall i|y_i=k} \boldsymbol{x}_i - \sum_{\forall j|y_j\neq k} \boldsymbol{x}_j \tag{8}$$

to obtain $\mathbf{w}_k = \boldsymbol{P}\backslash\boldsymbol{Q}_k$ in Eq. 6 for multi-classification (i.e., one vs all setting).

The pseudo-code for learning OLR multi-classification representing the concept of one-pass is described in Algorithm 1. To predict label of new data instance, we compute the linear function and select the one with the highest score: $\hat{y}_{\text{new}} = \underset{k}{\text{argmax}}\left[\mathbf{w}_k^T \boldsymbol{x}_{\text{new}}\right]$.

2) OLR for label-drift classification: The ability of learning the classifier for label-drift classification is necessary for the robustness and sustainability of the classification system since the data is growing and coming up with new classes over time.

For multiclass OLR, we update $\boldsymbol{P}$ in Eq. 7 and $\boldsymbol{Q}_{1...K}$ in Eq. 8. When the new class $K + 1$ appears in the new block $B + 1$ (cf. Fig. 1), the computation for $\boldsymbol{Q}_{K+1}$ is required. Although the previous data points are ignored in a spirit of online setting, we aim to store the previous information in a sufficient statistic matrix to compute $\boldsymbol{Q}_{K+1}$. Particularly, we utilize the fact that all of the previous data points do not belong to the new class $K+1$ as a definition of label-drift setting (see Sec. III and Fig. 1). Therefore, we introduce a new sufficient statistic matrix $\boldsymbol{Q}_0 \in \mathbb{R}^{D\times 1}$ such that $\boldsymbol{Q}_o = -\sum_{\forall \boldsymbol{x}_i \in \boldsymbol{X}_{1:B}} \boldsymbol{x}_i$ to penalize all previous data points which do not belong to a new class $K + 1$. Then we take $\boldsymbol{Q}_{K+1} = \boldsymbol{Q}_0$ and continue updating $\boldsymbol{Q}_0$ for the future classes. We present label-drift OLR learning in Algorithm 2.

3) OLR computation complexity: The cost for computing $\boldsymbol{P} = \sum_{i=1}^{N} \lambda_i \boldsymbol{x}_i \boldsymbol{x}_i^T + \mathbf{I}$ is $\mathcal{O}\left(N \times D^2\right)$ as $\boldsymbol{x}_i$ multiplies by itself. The cost for computing $\boldsymbol{Q}_k = \boldsymbol{X}^T \boldsymbol{Y}$ is $\mathcal{O}\left(N \times D\right)$. The cost for solving system of linear equation $\boldsymbol{P} \backslash \boldsymbol{Q}_k$ is $\mathcal{O}\left(D^{2.376}\right)$ [14] where $\boldsymbol{P} \in \mathbb{R}^{D \times D}$ is a square, symmetric and positive definite matrix, and $\boldsymbol{Q}_k \in \mathbb{R}^{D \times 1}$.

C. Distributed OLR on Apache Spark

We further address the large-scale classification problem where the number of data points is often significantly greater than the number of features, i.e., $N \gg D$. We parallelize in computing $\boldsymbol{P}$ and $\boldsymbol{Q}$ in a distributed system, specifically on Apache Spark. The Eqs. (7,8) can be reformulated using M parts as: $\boldsymbol{P} = \sum_{m=1}^{M} \boldsymbol{P}^{\langle m \rangle} + \boldsymbol{I},\ \boldsymbol{Q} = \sum_{m=1}^{M} \boldsymbol{Q}^{\langle m \rangle}$ wherein the m-th parts are computed as:

$$\boldsymbol{P}^{\langle m \rangle} = \boldsymbol{X}^{\langle m \rangle^\top} \text{diag}\left(\boldsymbol{\lambda}^{\langle m \rangle}\right) \boldsymbol{X}^{\langle m \rangle}$$

$$\boldsymbol{Q}^{\langle m \rangle} = \sum_{\forall \boldsymbol{x}_i \in \boldsymbol{X}^{\langle m \rangle} | y_i = k} \boldsymbol{x}_i - \sum_{\forall \boldsymbol{x}_j \in \boldsymbol{X}^{\langle m \rangle} | y_j \neq k} \boldsymbol{x}_j$$

It is clear that we only need the data part $\boldsymbol{X}^{\langle m \rangle}$ and auxiliary variable part $\boldsymbol{\lambda}^{\langle m \rangle}$ to compute $\boldsymbol{P}^{\langle m \rangle}$ and $\boldsymbol{Q}^{\langle m \rangle}$. Therefore, in the Spark system, the computation can be done in parallel with each partition being processed by one worker node. Once the matrices $\boldsymbol{P}$ and $\boldsymbol{Q}$ are fully specified, the driver will compute the parameters $\mathbf{w}$ by solving the system of linear equations. It is worthy to note that this operator is conducted on the driver node, thus not distributed.

V. Experiment

In this section, we conduct comprehensive experiments to quantitatively evaluate the performance of our proposed OLR and Spark-OLR on standard classification and label-drift classification. Our main goal is to demonstrate the scalability of OLR and Spark-OLR in training large-scale datasets, and their capabilities in handling the arrival of data points with new labels. In addition, we examine the trade-off between the computational cost and the predictive performance.

We use Matlab to implement the proposed OLR, and Python API of Spark version 1.4.1 to implement the Spark-OLR. *All source codes, data and experimental setups are published for reproducibility*[1].

A. Classification on medium size datasets

In the first experiment, we use 5 medium size datasets obtained from LIBSVM collection and UCI repository. We use the classification accuracy on the test set for evaluation. The computational cost is the sum of training time and testing time. Table I summarizes the classification results and running time of our proposed model and baselines.

In terms of running time, our proposed OLR outperforms all of the baselines methods in all used datasets by a large margin. The OLR is fast because it passes through each data point only once to compute the sufficient statistic ($\boldsymbol{P}$ and $\boldsymbol{Q}$), then solving the system of linear equations to obtain $\mathbf{w}$. When $N \gg D$, the OLR's complexity reduces to $\mathcal{O}(N)$. Although the naive Bayes, LDA and LR-SGD have the same complexity of $\mathcal{O}\left(N\right)$, our OLR is still the fastest as its parameters are computed efficiently using multithreaded programming (default by Matlab for independent loop, matrix multiplication and solving system of linear equation). We note that LR-SGD cannot utilize the matrix operation since it needs to iterate through each data point to update the model. In terms of predictive performance, the OLR achieves comparable results compared with those of the best baselines.

Quadrant visualization: The computational efficiency may compromise with predictive capability. In our experiment, all methods share a common property that is a trade-off between accuracy and speed. Studying this trade-off is an inherent challenge that should be investigated in a principled way. It is desirable to have a classifier that achieves a high prediction accuracy score at a low computational cost. To that end, we propose the *quadrant visualization* and the *quadrant score* to measure the relationship between speed and accuracy. For each algorithm, we record the accuracy and the computation time. Then, we plot them into the quarter of polar (cf. Fig. 2) which the x-axis corresponds to computational time and y-axis is classification accuracy.

The quadrant score (QS) is the distance between the projected point (by two axes) to the origin, mathematically: $QS = \sqrt{\left(100 \times \frac{Time}{Limit}\right)^2 + \left(100 - Acc\right)^2}$. The running time ($Time$) is scaled w.r.t. the time axis limit ($Limit$) to guarantee that the time factor does not overwhelm (or be overwhelmed by) the accuracy factor where $Limit$ is provided by user subject to the overall speed of other methods in the dataset.

The ideal algorithm with computation time 0 second and accuracy 100% would get $QS = 0$. In other words, the lower quadrant score indicates the better algorithm. We use quadrant visualization in Fig. 2 to provide an alternative view on comparison the classification performance to supplement the results in Table I.

B. Classification on large-scale datasets

Next we demonstrate the scalability of Spark-OLR using two large-scale public datasets (MNIST8M [15] and Airlines datasets) in which the number of training samples is much larger than the feature dimension.

The classification performance is evaluated using the accuracy for MNIST8M datasets whilst using F1 for the Airlines data as it is unbalanced with only 10% delayed labels (cf. Fig. 3). For comparison, we use 8 baselines implemented in Spark: logistic regression using stochastic gradient descent (Spark-LR-SGD) and limited-memory BFGS (Spark-LR-LBFGS); Naive Bayes (Spark-NB); linear SVM (Spark-LSVM); decision tree (Spark-DT); random forest (Spark-RF) [16], Spark-LIBLINEAR with logistic regression (Spark-LLin-LR) and with L2-loss SVM (Spark-LLin-SVM) [17]. All Spark-based methods run on a Hadoop-Spark cluster with 8 worker nodes, each node equipped with 32 vcores CPU, 128GB RAM.

[1] https://github.com/ntienvu/ICDM2016_OLR

Dataset N=#data points; D=#features	DNA N=3,186; D=180		Covtype N=581,012; D=54		SenseIT Vehicle N=98,528; D=100		MNIST N=70,000; D=784		KDD99 N=4,898,431; D=41	
Method	Acc	Time	Acc	Time	Acc	Time	Acc	Time	Acc	Time
Naive Bayes	91.23	0.16	43.52	1.76	75.93	0.82	83.61	6.05	88.33	12.9
Decision tree	90.64	0.43	**61.6**	20.96	76.51	66.69	87.8	60.64	**99.99**	88.3
k-nearest neighbors	76.56	1.76	–	25,000+	76.15	953.49	**96.9**	6,547	–	25,000+
Linear discrimination analysis	**94.1**	0.31	54.53	5.31	**80.78**	2.33	87.3	29.58	**99.24**	98.7
Linear SVM	**94.43**	0.09	55.8	78.3	**80.79**	130.22	91.68	126.7	91.11	2755
LR conjugate gradient descent (LR-CGD)	**94.77**	1.02	56.54	162.9	**79.89**	19.89	91.74	298	**99.09**	3444
LR stochastic gradient descent (LR-SGD)	**94.1**	0.24	52.04	115.37	74.00	11.45	85.94	28.22	98.87	1528
Bayesian LR Laplacian (BLR-LAP)	**94.86**	3.94	56.67	131.58	**80.88**	21.00	91.86	4021	**99.82**	2319
BLR Polya-Gamma (BLR-PG)	92.16	10.74	56.23	2603.52	**80.38**	391.77	84.71	391.93	57.35	21462
One-pass logistic regression (OLR)	**94.01**	**0.03**	56.42	**0.55**	**80.42**	**0.11**	86.47	**1.59**	**99.4**	**6.2**

Table I: Classification accuracy – Acc (%) and execution time – Time (seconds). The performance results within 1% of the best one are in **bold**. The kNN exceeds $25,000$ seconds, when running on Covtype and KDD datasets.

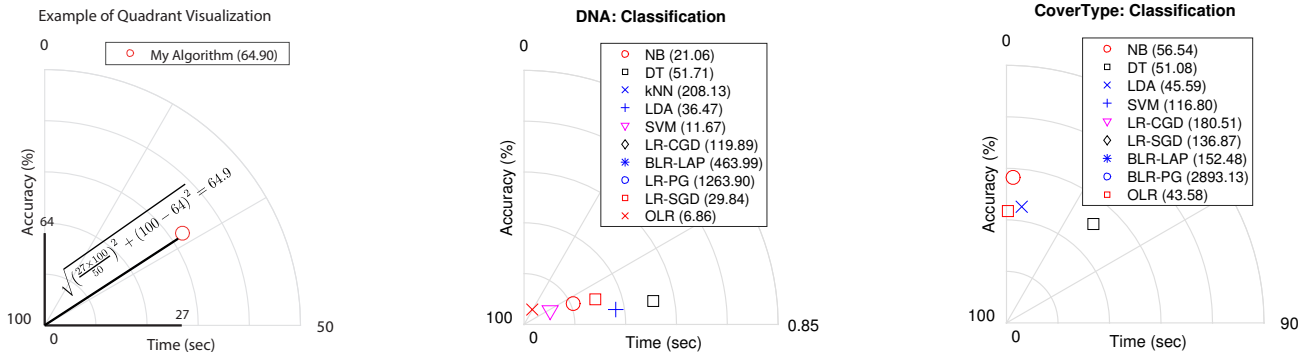

Figure 2: Quadrant visualization. The lower quadrant score (in the bracket) indicates the better algorithm that locates nearer to the origin. The algorithm is not shown if its quadrant score is greater than $100\sqrt{2}$.

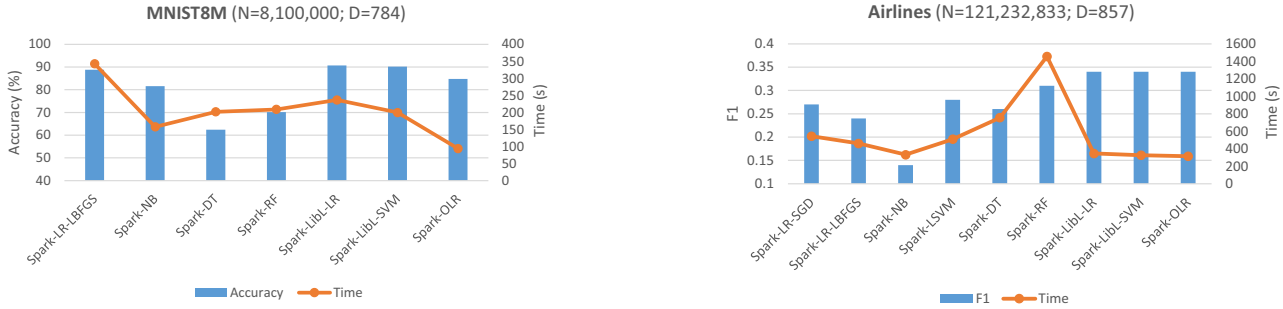

Figure 3: Comparison of classification performances of our Spark-OLR and baselines on four large-scale datasets.

Fig. 3 presents the classification performance on the testing set and execution time. We note that the Python API of Spark-LR-SGD and Spark-LSVM in Spark verion 1.4.1 do not support multiclass classification, and the Spark-NB was not implemented to model Gaussian distribution (real-valued data). Hence, their results for some datasets are not available. Overall, the training time of Spark-OLR is consistently superior to the Spark-implement_ation baselines while keeping comparable predictive performances. Our method can achieve approximately ten times speedup in training on all datasets. Particularly, compared to the Spark-LIBLINEAR – the most competitive baseline, the Spark-OLR achieves comparable classification results and much faster speed.

C. *Label-drift Classification*

Our last experiment is to demonstrate the new classification capacity of our proposed methods in handling the label-drift classification problem using the MNIST8M dataset. The dataset is then partitioned into ten blocks such that the new class will appear in the subsequent blocks. After training the model on each block, we record its classification performance on the original testing part which contains $10,000$ images.

For comparison, we use four baselines: naive Bayes, perceptron [18], passive aggressive (PA) and logistic regression using SGD (LR-SGD). Since concept-drift approaches [11], [10] are designed to detect and handle the change in feature-label joint distribution, they are neither directly applicable to

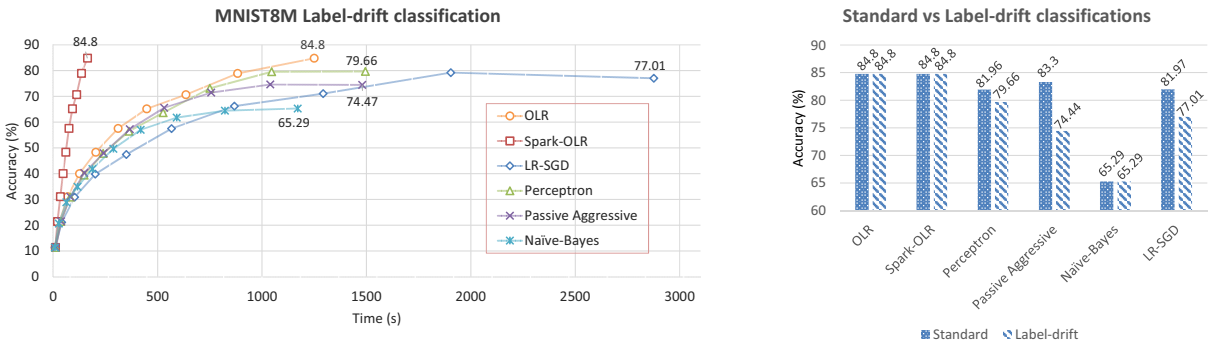

Figure 4: Label-drift classification performance (left), and the comparison with standard classification (right) on MNIST8M.

our label-drift setting nor scalable to big datasets. Thus, we cannot compare our methods with the concept-drift methods.

Fig. 4 (left) shows the performance with respect to classification accuracy and computational time cost. It is clear that our OLR and Spark-OLR achieve the best classification results of 84.8%. In addition, the Spark-OLR consistently performs an order of magnitude faster than other methods.

To investigate the impact of label-drift setting on the performance of algorithms, we compare the classification results in standard setting and label-drift setting in Fig. 4 (right). As can be seen from the figure, the label-drift classification scores of perceptron and LR-SGD drop by 2.8%, 6.1% compared with their results in standard setting, whilst the accuracy of PA significantly reduces from 83.3% in standard classification to 74.44% in label-drift task. This is because these algorithms iterate through each data point, if the learners suffer loss, they penalize the parameter that the data point is misclassified and reward the parameter of the true class. Therefore, the previous data points (before the existence of a new class) do not have a chance to penalize a new class because they are already ignored by online setting.

Different from the LR-SGD, Perceptron and PA, the naive Bayes (NB) and our proposed methods (OLR and Spark-OLR) can efficiently handle the new classes since they only need to maintain the sufficient statistic, i.e., mean and covariance in naive Bayes, and matrices $\boldsymbol{P}$ and $\boldsymbol{Q}$ in our methods. Therefore, the classification results are robust and invariant subject to changing from standard learning to label-drift learning.

VI. Conclusion

We have proposed a novel method – *one-pass logistic regression* (OLR) for handling the large-scale and label-drift classification without retraining the model from scratch. We further introduce the *sparkling OLR,* a distributed model for big data with hundreds million of examples.

Extensive experiments on large-scale datasets show that the predictive performances of our methods are comparable or better than those of state-of-the-art baselines, whilst the execution time is much faster at an order of magnitude. In addition, the OLR and Spark-OLR are invariant to data shuffling and have no hyperparameter to tune. We make the source codes and data available for reproducibility.

References

[1] D. R. Cox, "The regression analysis of binary sequences," *Journal of the Royal Statistical Society. Series B (Methodological)*, pp. 215–242, 1958.

[2] C. M. Bishop, *Pattern recognition and machine learning*. springer New York, 2006.

[3] N. G. Polson, J. G. Scott, and J. Windle, "Bayesian inference for logistic models using pólya–gamma latent variables," *Journal of the American Statistical Association*, vol. 108, no. 504, pp. 1339–1349, 2013.

[4] G. Widmer and M. Kubat, "Learning in the presence of concept drift and hidden contexts," *Machine learning*, vol. 23, no. 1, pp. 69–101, 1996.

[5] R. Klinkenberg and T. Joachims, "Detecting concept drift with support vector machines." in *ICML*, 2000, pp. 487–494.

[6] I. Žliobaitė, "Learning under concept drift: an overview," *arXiv preprint arXiv:1010.4784*, 2010.

[7] A. T. Pham, R. Raich, X. Z. Fern, J. P. Arriaga, and D. UNICAN, "Multi-instance multi-label learning in the presence of novel class instances," in *Proceedings of The 32nd ICML*, 2015, pp. 2427–2435.

[8] T. D. Nguyen, V. Nguyen, T. Le, and D. Phung, "Distributed data augmented support vector machine on spark," in *Pattern Recognition (ICPR), 2016 23st International Conference on*. IEEE, 2016.

[9] A. Tsymbal, "The problem of concept drift: definitions and related work," *Computer Science Department, Trinity College Dublin*, vol. 106, p. 2, 2004.

[10] J. Gama, I. Žliobaitė, A. Bifet, M. Pechenizkiy, and A. Bouchachia, "A survey on concept drift adaptation," *ACM Computing Surveys (CSUR)*, vol. 46, no. 4, p. 44, 2014.

[11] L. L. Minku and X. Yao, "Ddd: A new ensemble approach for dealing with concept drift," *Knowledge and Data Engineering, IEEE Transactions on*, vol. 24, no. 4, pp. 619–633, 2012.

[12] Y. Bengio, J. Louradour, R. Collobert, and J. Weston, "Curriculum learning," in *Proceedings of the 26th annual international conference on machine learning*. ACM, 2009, pp. 41–48.

[13] T. D. Nguyen, V. Nguyen, T. Le, and D. Phung, "Sparkling vector machines," in *Workshop on Machine Learning Systems at Neural Information Processing Systems (NIPS)*, 2015.

[14] G. H. Golub and C. F. Van Loan, *Matrix computations*. JHU Press, 2012, vol. 3.

[15] G. Loosli, S. Canu, and L. Bottou, "Training invariant support vector machines using selective sampling," *Large scale kernel machines*, pp. 301–320, 2007.

[16] X. Meng, J. Bradley, B. Yavuz, E. Sparks, S. Venkataraman, D. Liu, J. Freeman, D. Tsai, M. Amde, S. Owen *et al.*, "Mllib: Machine learning in apache spark," *arXiv preprint arXiv:1505.06807*, 2015.

[17] C.-Y. Lin, C.-H. Tsai, C.-P. Lee, and C.-J. Lin, "Large-scale logistic regression and linear support vector machines using spark," in *Proceedings of IEEE International Conference on Big Data*, 2015, pp. 519–528.

[18] S. C. Hoi, J. Wang, and P. Zhao, "Libol: a library for online learning algorithms," *The Journal of Machine Learning Research*, vol. 15, no. 1, pp. 495–499, 2014.