Characterizing Inverse Time Dependency in Multi-class Learning

Danqi Chen
Institute for Interdisciplinary Information Sciences
Tsinghua University
Beijing, China
cdq10131@gmail.com

Weizhu Chen
Microsoft Research Asia
Beijing, China
wzchen@microsoft.com

Qiang Yang
Hong Kong University of Science & Technology
Hong Kong
qyang@cse.ust.hk

***Abstract*—The training time of most learning algorithms increases as the size of training data increases. Yet, recent advances in linear binary SVM and LR challenge this commonsense by proposing an inverse dependency property, where the training time decreases as the size of training data increases. In this paper, we study the inverse dependency property of multi-class classification problem. We describe a general framework for multi-class classification problem with a single objective to achieve inverse dependency and extend it to three popular multi-class algorithms. We present theoretical results demonstrating its convergence and inverse dependency guarantee. We conduct experiments to empirically verify the inverse dependency of all the three algorithms on large-scale datasets as well as to ensure the accuracy.**

***Keywords*-multi-class learning; large-scale classification; supervised learning; inverse dependency;**

I. Introduction

In conventional regularized learning, as the size of the training data expands, a learning algorithm is expected to take more runtime to learn from the data. Yet, recent works propose a counter-intuitive observation, such that given a fixed degree of accuracy, the running time of some linear algorithms may decrease as the size of the training data increases. [1] first put forward a linear SVM solver called Pegasos and demonstrated the inverse time-dependency property of linear binary SVM. [2] generalized the inverse time-dependency property by a general Primal Gradient Solver to L_p norms, and extended the property to binary logistic regression algorithms and the least square loss functions.

The inverse dependency property is indeed a very desirable one due to its sub-linear complexity in terms of data size. This makes learning algorithms very efficient and scalable. For example, Pegasos and related algorithms can train an effective binary classifier for the RCV1 dataset with about 600,000 samples in seconds. Previous works have demonstrated this effectiveness for binary classification problems. However, in many real world scenarios, machine learning applications belong to the multi-class problems, where the outputs have K categories such that $K > 2$. Yet, how to achieve the inverse dependency property for the multi-class problem is an under-explored problem.

Contributions In this paper, we make several advances in inverse time dependency. First, we put forward a general framework for multi-class classification algorithms with a single objective to achieve inverse dependency. We then extend it to three popular multi-class algorithms: multi-class logistic regression, multi-class SVM and multi-class perceptron. Second, we present theoretical results to prove the algorithms' convergence. Simultaneously, we provide a proof of the inverse dependency property thus that all three algorithms can achieve a sub-linear complexity for large-scale datasets. It is worth noting that the convergence and proofs of inverse time are the keys to ensure the inverse dependency property. Third, we conduct experiments on three large-scale datasets. We show that all three algorithms achieve inverse dependency with high efficiency. Finally, to ensure the effectiveness of the algorithms, we compare them with popular alternatives in the literature. Empirical results substantiate our theoretical results and confirm their effectiveness in terms of accuracy.

II. Preliminaries

Before delving into details of inverse dependency, we describe the generalization objective for multi-class classification in a regularized learning setting. We consider the loss functions of the three selected popular learning algorithms to demonstrate the inverse dependency property.

A. Regularized Learning

Given a training set $\mathcal{X} = \{(\mathbf{x}_i, y_i)\}_{i=1}^N \subset R^d \times \{1, 2, \ldots, K\}$ with N samples, where d is the dimension of features and K is the number of classes, we assume that input samples are i.i.d. from an unknown probability distribution P. A general multi-class classification algorithm focuses on learning a linear predictor $\mathbf{w_k}$ for each class k where $k \in \{1, 2, \ldots, K\}$; we may use $\mathbf{w}$ to denote the collection of all $\mathbf{w_k}$. Following the regularized learning theory in [3], [4], we may express the generalization objective as:

$$F_\lambda = \lambda \cdot re(\mathbf{w_1}, \ldots, \mathbf{w_K}) + l(\mathbf{w_1}, \ldots, \mathbf{w_K}) \quad (1)$$

$$= \lambda \cdot re(\mathbf{w}) + \mathbb{E}_{(\mathbf{x},y)\sim P}\left(l(\mathbf{w}; (\mathbf{x}, y))\right) \quad (2)$$

where $re(\mathbf{w_1}, \ldots, \mathbf{w_K})$ is the regularized term with a positive regularization parameter λ, and $l(\mathbf{w_1}, \ldots, \mathbf{w_K})$ is the loss function measuring the discrepancy between the predicted output and the real output. Although the goal is to minimize the *generalization objective*, a practical approach

often tries to minimize the *empirical objective*, where the averaged loss over N samples is used to approximate the generalized loss. As a special case of treating the L_2 norm as the regularizer:

$$f = \frac{\lambda}{2}\sum_{k=1}^{K}\mathbf{w}_k^\intercal\mathbf{w}_k + \frac{1}{N}\sum_{i=1}^{N} l(\mathbf{w}_1,\ldots,\mathbf{w}_K;(\mathbf{x}_i,y_i)) \quad (3)$$

where the loss function $l(\mathbf{w}_1,\ldots,\mathbf{w}_K;(\mathbf{x}_i,y_i))$ may vary in different algorithms.

B. Multi-class Logistic Regression (LR)

A multi-class logistic regression model estimates the conditional class probabilities as

$$Pr(C_k|\mathbf{w}_1,\ldots,\mathbf{w}_K;\mathbf{x}) = \frac{e^{\mathbf{w}_k^\intercal\mathbf{x}}}{\sum_{j=1}^{K} e^{\mathbf{w}_j^\intercal\mathbf{x}}} \quad (4)$$

for $k = 1,2,\ldots,K$ and C_k denotes class k. The parameters $\mathbf{w}_1,\ldots,\mathbf{w}_K$ are learned by the maximum likelihood approach, which is equivalent to minimizing the following negative log-likelihood loss:

$$\begin{aligned} E(\mathbf{w}_1,\ldots,\mathbf{w}_K|\mathcal{X}) &= -\sum_{i=1}^{N}\sum_{k=1}^{K}\delta_{y_i,k}\log Pr(C_k|\mathbf{w};\mathbf{x}_i) \\ &= \sum_{i=1}^{N}\left(-\mathbf{w}_{y_i}^\intercal\mathbf{x}_i + \log(\sum_{j=1}^{K} e^{\mathbf{w}_j^\intercal\mathbf{x}_i})\right) \end{aligned} \quad (5)$$

where $\delta_{i,j}$ is the Kronecker delta function with $\delta_{i,j} = 1$ when $i = j$ and $\delta_{i,j} = 0$ otherwise. Thus, the loss function of multi-class LR in Eq.3 is

$$l_{LR} = -\mathbf{w}_y^\intercal\mathbf{x} + \log(\sum_{j=1}^{K} e^{\mathbf{w}_j^\intercal\mathbf{x}}). \quad (6)$$

C. Multi-class SVM

In [5], a solution for multi-class SVM by solving a single optimization problem was proposed. For this solution, the optimization problem for multi-class SVM is formulated as:

$$\min_{\mathbf{w},\xi}\frac{\lambda}{2}\sum_{k=1}^{K}\mathbf{w}_k^\intercal\mathbf{w}_k + \frac{1}{N}\sum_{i=1}^{N}\xi_i \quad (7)$$

$$\text{subject to } \mathbf{w}_{y_i}^\intercal\mathbf{x}_i - \mathbf{w}_k^\intercal\mathbf{x}_i \geq e_i^k - \xi_i, i = 1,2,\ldots,N \quad (8)$$

where $e_i^k = 1 - \delta_{y_i,k}$. The loss function is

$$l_{SVM} = \max\{0, \max_{k\neq y}\{1 + \mathbf{w}_k^\intercal\mathbf{x} - \mathbf{w}_y^\intercal\mathbf{x}\}\}. \quad (9)$$

D. Muli-class Perceptron

Multi-class perceptron model aims to find the K paramaters $\mathbf{w}_1,\ldots,\mathbf{w}_K$ to maximize

$$\sum_i\left(\mathbf{w}_{y_i}^\intercal\mathbf{x}_i - \max_k \mathbf{w}_k^\intercal\mathbf{x}_i\right), \quad (10)$$

which can be understood intuitively as penalizing the largest incorrect labelling. We may take its negative value as the loss function :

$$l_{Per} = \max_k\{\mathbf{w}_k^\intercal\mathbf{x}\} - \mathbf{w}_y^\intercal\mathbf{x} \quad (11)$$

III. The Main Results

In this section, we propose a general stochastic sub-gradient descent framework for solving multi-class classification problems, which is inspired by the Pegasos algorithm [1] for solving 2-class SVM. Then we explore the required properties of loss functions and apply the framework to the three multi-class algorithms.

The basic idea to solve a linear model is based on the sub-gradient descent:

$$\begin{aligned} (\mathbf{w}_1^{(t)},\ldots,\mathbf{w}_K^{(t)}) = \quad & (\mathbf{w}_1^{(t-1)},\ldots,\mathbf{w}_K^{(t-1)}) \\ & -\eta(\tfrac{\partial f}{\partial\mathbf{w}_1},\ldots,\tfrac{\partial f}{\partial\mathbf{w}_K}) \end{aligned} \quad (12)$$

where η is the learning rate. Note that if f is not differentiable at $(\mathbf{w}'_1,\ldots,\mathbf{w}'_K)$, it picks any sub-gradient belonging to $\partial f(\mathbf{w}'_1,\ldots,\mathbf{w}'_K)$ as $(\frac{\partial f}{\partial\mathbf{w}_1},\ldots,\frac{\partial f}{\partial\mathbf{w}_K})$. The computation cost is expensive since the empirical objective has as many terms as the amount of training data. An alternative solution is to randomly pick a fixed-size subset $A_t \subseteq \mathcal{X}$ where $|A_t| = r$ in each iteration, and then use the temporal objective function $f^{(t)}$ to approximate f:

$$f^{(t)} = \frac{\lambda}{2}\sum_{k=1}^{K}\mathbf{w}_k^\intercal\mathbf{w}_k + \frac{1}{r}\sum_{(\mathbf{x},y)\in A_t} l(\mathbf{w}_1,\ldots,\mathbf{w}_K;(\mathbf{x},y)) \quad (13)$$

For $k = 1,2,\ldots,K$, we compute

$$\frac{\partial f^{(t)}}{\partial\mathbf{w}_k} = \lambda\mathbf{w}_k + \frac{1}{r}\sum_{(\mathbf{x},y)\in A_t}\frac{\partial l(\mathbf{w}_1,\ldots,\mathbf{w}_K;(\mathbf{x},y))}{\partial\mathbf{w}_k}$$

at $(\mathbf{w}_1^{(t-1)},\ldots,\mathbf{w}_K^{(t-1)})$ and use it as an approximation of $\frac{\partial f}{\partial\mathbf{w}_k}$, and then apply the sub-gradient descent Eq.12 to compute $(\mathbf{w}_1^{(t)},\ldots,\mathbf{w}_K^{(t)})$. Here we set $\eta_t = \frac{1}{\lambda t}$, where t is the number of iterations. Hence, Eq.12 becomes

$$\begin{aligned} (\mathbf{w}_1^{(t)},\ldots,\mathbf{w}_K^{(t)}) = \quad & \tfrac{t-1}{t}(\mathbf{w}_1^{(t-1)},\ldots,\mathbf{w}_K^{(t-1)}) \\ & -\tfrac{1}{\lambda tr}(\mathbf{v}_1,\ldots,\mathbf{v}_K) \end{aligned} \quad (14)$$

with $\mathbf{v}_k = \sum_{(\mathbf{x},y)\in A_t}\frac{\partial l(\mathbf{w}_1,\ldots,\mathbf{w}_K;(\mathbf{x},y))}{\partial\mathbf{w}_k}(\mathbf{w}_1^{(t-1)},\ldots,\mathbf{w}_K^{(t-1)})$.

The framework of our multi-class inverse dependency solver *MCID Solver* is given in Alg.1. It is worth noting that we sample a fixed-size subset in each iteration, thus the

Algorithm 1 MCID Solver

INPUT: $\mathcal{X},\lambda,T,r$

$\mathbf{w}_1^{(0)},\ldots,\mathbf{w}_K^{(0)} \leftarrow \mathbf{0}$
for $t = 1,2,\ldots,T$ **do**
  Randomly choose $A_t \subset \mathcal{X}$ where $|A_t| = r$
  for $k = 1,2,\ldots,K$ **do**
    $\mathbf{w}_k^{(t)} \leftarrow \frac{t-1}{t}\mathbf{w}_k^{(t-1)}$
  for $(\mathbf{x},y) \in A_t$ **do**
    for $k = 1,2,\ldots,K$ **do**
      $\mathbf{w}_k^{(t)} \leftarrow \mathbf{w}_k^{(t)} - \frac{1}{\lambda tr}\frac{\partial l}{\partial\mathbf{w}_k}(\mathbf{w}_1^{(t-1)},\ldots,\mathbf{w}_K^{(t-1)})$

OUTPUT: $\mathbf{w}_1^{(T)},\ldots,\mathbf{w}_K^{(T)}$

runtime of a single iteration does not increase as the data size increases, which provides a guarantee for the inverse time-dependency property.

A. Applications

In this sub-section, we will utilize the above framework into the three linear classification algorithms respectively. To adapt the three different loss functions, we compute the sub-gradients first.

- Multi-class logistic regression: Given Eq.6, we can compute $\frac{\partial l_{LR}}{\mathbf{w}_k}$ directly:

$$\frac{\partial l_{LR}}{\partial \mathbf{w}_k} = -\delta_{y,k}\mathbf{x} + \frac{e^{\mathbf{w}_k^\intercal \mathbf{x}} \cdot \mathbf{x}}{\sum_{j=1}^{K} e^{\mathbf{w}_j^\intercal \mathbf{x}}} \tag{15}$$

- Multi-class SVM: Given Eq.9. Let $k^* = \arg\max_{k \neq y} \mathbf{w}_k^\intercal \mathbf{x}$[1],$l_{SVM} = \max\{0, 1 + \mathbf{w}_{k^*}^\intercal \mathbf{x} - \mathbf{w}_y^\intercal \mathbf{x}\}$

$$\frac{\partial l_{SVM}}{\partial \mathbf{w}_k} = \begin{cases} \delta_{k^*,k}\mathbf{x} - \delta_{y,k}\mathbf{x} & 1 + \mathbf{w}_{k^*}^\intercal \mathbf{x} - \mathbf{w}_y^\intercal \mathbf{x} > 0 \\ 0 & \text{otherwise} \end{cases} \tag{16}$$

- Multi-class perceptron: Given Eq.11. Similarly, we define $k^* = \arg\max_k \mathbf{w}_k^\intercal \mathbf{x}$, l_{Per} becomes

$$\begin{aligned} l_{Per} &= \mathbf{w}_{k^*}^\intercal \mathbf{x} - \mathbf{w}_y^\intercal \mathbf{x} \\ \frac{\partial l_{Per}}{\partial \mathbf{w}_k} &= \delta_{k^*,k}\mathbf{x} - \delta_{y,k}\mathbf{x} \end{aligned} \tag{17}$$

The pseudo-codes of multi-class logistic regression, multi-class SVM and multi-class perceptron are given in Alg.2, Alg.3 and Alg.4. Note that we only write the update procedure here (line $6 \sim 8$ in Alg.1).

Algorithm 2 Multi-class Logistic Regression

for $(\mathbf{x}, y) \in A_t$ **do**
  for $k = 1, \ldots, K$ **do**
    $o_k \leftarrow e^{\mathbf{w}_k^{(t-1)\intercal} \mathbf{x}}$
  $z \leftarrow \sum_{k=1}^{K} o_k$
  for $k = 1, \ldots, K$ **do**
    $\mathbf{w}_k^{(t)} \leftarrow \mathbf{w}_k^{(t)} - \frac{1}{\lambda t r} \frac{o_k}{z} \mathbf{x}$
  $\mathbf{w}_y^{(t)} \leftarrow \mathbf{w}_y^{(t)} + \frac{1}{\lambda t r} \mathbf{x}$

Algorithm 3 Multi-class SVM

for $(\mathbf{x}, y) \in A_t$ **do**
  $k^* \leftarrow \arg\max_{k \neq y} \mathbf{w}_k^{(t-1)\intercal} \mathbf{x}$
  if $1 + \mathbf{w}_{k^*}^{(t-1)\intercal} \mathbf{x} - \mathbf{w}_y^{(t-1)\intercal} \mathbf{x} > 0$ **then**
    $\mathbf{w}_{k^*}^{(t)} \leftarrow \mathbf{w}_{k^*}^{(t)} - \frac{1}{\lambda t r} \mathbf{x}$
    $\mathbf{w}_y^{(t)} \leftarrow \mathbf{w}_y^{(t)} + \frac{1}{\lambda t r} \mathbf{x}$

IV. Analysis

In this section, we prove two main theorems in the paper. One theorem shows that our algorithms can converge with a rate $\frac{1}{T}$, and the other shows that our algorithms have an inverse time dependency on the size of training data when their loss functions satisfy the convexity and boundedness properties. Before discussing these results, we first prove that the three loss functions l_{LR}, l_{SVM}, l_{Per} all satisfy convexity and boundedness, so that the latter proofs can be easily adapted to the three algorithms.

[1]Note that if there are more than one k achieving the maximum $\mathbf{w}_k^\intercal \mathbf{x}_i$, we can pick any one for k^*.

Algorithm 4 Multi-class Perceptron

for $(\mathbf{x}, y) \in A_t$ **do**
  $k^* \leftarrow \arg\max_k \mathbf{w}_k^{(t-1)\intercal} \mathbf{x}$
  $\mathbf{w}_{k^*}^{(t)} \leftarrow \mathbf{w}_{k^*}^{(t)} - \frac{1}{\lambda t r} \mathbf{x}$
  $\mathbf{w}_y^{(t)} \leftarrow \mathbf{w}_y^{(t)} + \frac{1}{\lambda t r} \mathbf{x}$

A. Convexity & Boundedness

To achieve the inverse dependency and the consistency property of the *MCID Solver*, there are two specific requirements: the **convexity** of the loss functions and the **boundedness** of the sub-gradient of the objective function:

- Convexity: $l(\mathbf{w}_1, \ldots, \mathbf{w}_K; (\mathbf{x}, y))$ satisfies the convexity w.r.t $\mathbf{w}_1, \ldots, \mathbf{w}_K$. i.e.: For any $0 < \alpha < 1$, $(\mathbf{w}_1^{(1)}, \ldots, \mathbf{w}_K^{(1)}), (\mathbf{w}_1^{(2)}, \ldots, \mathbf{w}_K^{(2)})$, then

$$\begin{aligned} & l(\alpha(\mathbf{w}_1^{(1)}, \ldots, \mathbf{w}_K^{(1)}) + (1-\alpha)(\mathbf{w}_1^{(2)}, \ldots, \mathbf{w}_K^{(2)})) \\ \leq\ & \alpha l(\mathbf{w}_1^{(1)}, \ldots, \mathbf{w}_K^{(1)}) + (1-\alpha) l(\mathbf{w}_1^{(2)}, \ldots, \mathbf{w}_K^{(2)}) \end{aligned}$$

- Boundedness: the sub-gradient can be bounded by a positive constant:

$$\exists C > 0, \forall 1 \leq k \leq K, |\frac{\partial f}{\partial \mathbf{w}_k}| \leq C$$

Theorem 1. *The three loss functions l_{LR}, l_{SVM}, l_{Per} that are given in Eq.6, Eq.9, Eq.11 all satisfy the properties of convexity and boundedness.*

Proof: For convexity, we prove l_{SVM} here, the convexity of l_{LR} and l_{Per} can be proved similarly. Let $h_{k,y}(\mathbf{w}_1, \ldots, \mathbf{w}_K) = 1 + \mathbf{w}_k^\intercal \mathbf{x} - \mathbf{w}_y^\intercal \mathbf{x}$, then $l_{SVM} = \max\{0, \max_{k \neq y} h_{k,y}(\mathbf{w})\}$. Since

$$\begin{aligned} & \max_{k \neq y}\{h_{k,y}(\alpha \mathbf{w}^{(1)} + (1-\alpha)\mathbf{w}^{(2)})\} \\ =\ & \max_{k \neq y}\{\alpha h_{k,y}(\mathbf{w}^{(1)}) + (1-\alpha) h_{k,y}(\mathbf{w}^{(2)})\} \\ \leq\ & \alpha \max_{k \neq y} h_{k,y}(\mathbf{w}^{(1)}) + (1-\alpha) \max_{k \neq y} h_{k,y}(\mathbf{w}^{(2)}) \\ \leq\ & \alpha \max\{0, \max_{k \neq y} h_{k,y}(\mathbf{w}^{(1)})\} + (1-\alpha) \max\{0, \max_{k \neq y} h_{k,y}(\mathbf{w}^{(2)})\} \\ =\ & \alpha l_{SVM}(\mathbf{w}^{(1)}) + (1-\alpha) l_{SVM}(\mathbf{w}^{(2)}) \end{aligned}$$

and $0 \leq \alpha l_{SVM}(\mathbf{w}^{(1)}) + (1-\alpha) l_{SVM}(\mathbf{w}^{(2)})$, we reach our conclusion. For boundedness, we assume that $(\mathbf{x}, y) \in \mathcal{X}$, $\|\mathbf{x}\| \leq R$. We can easily verify that the subgradients in Eq.15, Eq.16, Eq.17 all satisfy

$$|\frac{\partial l_{LR}}{\partial \mathbf{w}_k}| \leq \|\mathbf{x}\|, |\frac{\partial l_{SVM}}{\partial \mathbf{w}_k}| \leq \|\mathbf{x}\|, |\frac{\partial l_{Per}}{\partial \mathbf{w}_k}| \leq \|\mathbf{x}\|$$

Thus the sub-gradients are bounded by R. ■

B. Convergence

Next, we prove that the framework has the desired converge rate in this subsection. For this proof, we start by introducing a lemma:

Lemma 1. *Let $f^{(1)}, f^{(2)}, \ldots, f^{(T)}$ be a sequence of functions such that for all $t = 1, \ldots, T$, $f^{(t)} = \lambda h + g^{(t)}$ with h being strongly convex w.r.t $\|\cdot\|_2$ and $g^{(t)}$ is a convex function. Let $\mathbf{w}^{(0)}, \mathbf{w}^{(1)}, \ldots, \mathbf{w}^{(T)}$ be a sequence of vectors such that $\mathbf{w}^{(0)} = \mathbf{0}$, for $t \geq 1$, $\mathbf{w}^{(t)} = \mathbf{w}^{(t-1)} - \frac{1}{\lambda t}(\lambda \mathbf{w}^{(t-1)} + \mathbf{v}^{(t)})$ where $\mathbf{v}^{(t)} \in \partial g^{(t)}(\mathbf{w}^{(t-1)})$. Then*

$$\sum_{t=1}^{T} f^{(t)}(\mathbf{w}^{(t)}) - \min_{\mathbf{w}} \sum_{t=1}^{T} f^{(t)}(\mathbf{w}) \leq \frac{1}{2} \sum_{t=1}^{T} \frac{\|\mathbf{v}^{(t)}\|_2^2}{\lambda t}$$

This proof can be learnt as a special case of Theorem 2 in [6].

Theorem 2. *Assume that $f, f^{(t)}$ are respectively given in the form of Eq.3 and Eq.13 where l satisfies convexity and boundedness. Let $\delta \in (0, 1)$, then with a probability of at least $1 - \delta$ over the choices of $A_1, \ldots, A_T$, we have*

$$f(\mathbf{w}_1^{(T)}, \ldots, \mathbf{w}_K^{(T)}) \leq \min f(\mathbf{w}_1, \ldots, \mathbf{w}_K) + \frac{c \log T}{\delta \lambda T} \quad (18)$$

Proof: Since l is convex, it is straightforward to show that $g^{(t)} = \frac{1}{r} \sum_{(\mathbf{x},y) \in A_t} l(\mathbf{w}_1, \ldots, \mathbf{w}_K; (\mathbf{x}, y))$ is convex. Plus that $h = \frac{1}{2}\|(\mathbf{w}_1, \ldots, \mathbf{w}_K)\|_2^2$ is strongly convex, $\{f^{(t)}\}$ satisfies the condition in Lemma 1, therefore $\sum_{t=1}^{T} f^{(t)}(\mathbf{w}^{(t)}) - \min \sum_{t=1}^{T} f^{(t)}(\mathbf{w})$ can be bounded by $\frac{1}{2} \sum_{t=1}^{T} \frac{\|\mathbf{v}^{(t)}\|_2^2}{\lambda t}$. Furthermore, since l satisfies boundedness, thus $\sum_{t=1}^{T} f^{(t)}(\mathbf{w}^{(t)}) - \min \sum_{t=1}^{T} f^{(t)}(\mathbf{w})$ can be bounded by $\frac{c \log T}{\lambda}$ where c is a positive constant. Denote $(\mathbf{w}_1^*, \ldots, \mathbf{w}_K^*) = arg \min_{\mathbf{w}} f(\mathbf{w}_1, \ldots, \mathbf{w}_K)$, there is

$$\sum_{t=1}^{T} f^{(t)}(\mathbf{w}_1^{(t)}, \ldots, \mathbf{w}_K^{(t)}) - \sum_{t=1}^{T} f^{(t)}(\mathbf{w}_1^*, \ldots, \mathbf{w}_K^*) \leq \frac{c \log T}{\lambda}.$$

Taking the expectation of the above inequality and picking an index t_0 uniformly at random from $\{1, 2, \ldots, T\}$, we have

$$E(f(\mathbf{w}_1^{(t_0)}, \ldots, \mathbf{w}_K^{(t_0)})) - f(\mathbf{w}_1^*, \ldots, \mathbf{w}_K^*) \leq \frac{c \log T}{\lambda T}.$$

Applying Markov inequality, we can arrive at the conclusion that with probability of at least $1 - \delta$,

$$f(\mathbf{w}_1^{(t_0)}, \ldots, \mathbf{w}_K^{(t_0)}) - f(\mathbf{w}_1^*, \ldots, \mathbf{w}_K^*) \leq \frac{c \log T}{\delta \lambda T}.$$

Note that we can view T as a random index drawn from $\{1, 2, \ldots, \hat{T}\}$ where $\hat{T} > T$, thus we can return $(\mathbf{w}_1^{(T)}, \ldots, \mathbf{w}_K^{(T)})$ after T iterations instead of a random index. ■

Theorem 2 states that the framework in Alg.1 can converge with rate of $\frac{1}{T}$ whenever the loss functions satisfy convexity and boundedness. We have shown that the three loss functions l_{LR}, l_{SVM}, l_{Per} all satisfy the two properties, so our three algorithms can achieve an accuracy of ϵ with a confidence $1 - \delta$ with $\tilde{O}(\frac{1}{\delta \lambda \epsilon})$ iterations.

C. Inverse Time Dependency

In this subsection, we will show that to attain a fixed generalization error, more training data will yield less training time. Below is the main theorem.

Theorem 3. *For any $(\mathbf{w}'_1, \ldots, \mathbf{w}'_K)$, if we want to attain a generalization error $\mathbb{E}(l(\hat{\mathbf{w}}_1, \ldots, \hat{\mathbf{w}}_K)) \leq \mathbb{E}(l(\mathbf{w}'_1, \ldots, \mathbf{w}'_K)) + \alpha$ with a confidence $1 - \delta$ where l satisfies convexity and "boundedness", $(\hat{\mathbf{w}}_1, \ldots, \hat{\mathbf{w}}_K)$ is a solution of f in Eq.3 with an accuracy ϵ. i.e.,*

$$(\hat{\mathbf{w}}_1, \ldots, \hat{\mathbf{w}}_K) \leq \min f(\mathbf{w}_1, \ldots, \mathbf{w}_K) + \epsilon,$$

the runtime $\hat{T}$ of our algorithm has an inverse dependency on the size of training set N:

$$\hat{T} = \tilde{O}(\frac{d^* r K / \delta}{(\frac{\alpha}{\|(\mathbf{w}'_1, \ldots, \mathbf{w}'_K)\|})^2 - \tilde{O}(\frac{1}{N})}) \quad (19)$$

Proof: Following the "oracle" analysis in [7], we can conclude that

$$\begin{aligned} \mathbb{E}(l(\hat{\mathbf{w}}_1, \ldots, \hat{\mathbf{w}}_K)) &\leq \mathbb{E}(l(\mathbf{w}'_1, \ldots, \mathbf{w}'_K)) + \tilde{O}(\frac{K d^* r}{\delta \lambda \hat{T}}) \\ &+ \frac{\lambda}{2}\|(\mathbf{w}'_1, \ldots, \mathbf{w}'_K)\|^2 + \tilde{O}(\frac{1}{\lambda N}) \quad (20) \end{aligned}$$

The above conclusion is obtained from the main result in [3] and the result of Theorem 2: $O(\epsilon) = O(\frac{K d^* r}{\delta \lambda \hat{T}})$. If we choose

$$\lambda = \tilde{\Theta}\left(\frac{1}{\|(\mathbf{w}'_1, \ldots, \mathbf{w}'_K)\|} \sqrt{\tilde{O}(\frac{K d^* r}{\hat{T} \delta} + \frac{1}{N})}\right),$$

the bound becomes

$$\begin{aligned} \mathbb{E}(l(\hat{\mathbf{w}}_1, \ldots, \hat{\mathbf{w}}_K))) &\leq \mathbb{E}(l(\mathbf{w}'_1, \ldots, \mathbf{w}'_K)) \\ + \quad \tilde{O}&\left(\|(\mathbf{w}'_1, \ldots, \mathbf{w}'_K)\| \sqrt{\tilde{O}(\frac{K d^* r}{\hat{T} \delta} + \frac{1}{N})}\right) \end{aligned}$$

therefore $\hat{T} = \tilde{O}(\frac{K d^* r / \delta}{(\frac{\alpha}{\|(\mathbf{w}'_1, \ldots, \mathbf{w}'_K)\|})^2 - \tilde{O}(\frac{1}{N})})$, we conclude our proof with $\hat{T}$ showing an inverse dependency on N. ■

V. Experiments

In this section, we first present experimental results to demonstrate the inverse dependency property of our three algorithms and strengthen our theoretical results. We next compare the performance of our algorithms with the alternatives in the literature, including the SVM^{multiclass}[8] for multi-class SVM, and "one-versus-the-rest" (denoted as OVR) methods based on binary classification algorithms to indirectly solve the multi-class SVM and LR problem.

A. Datasets

Table I lists the datasets used in our experiments.

Table I
Datasets

Dataset	K	# training	# test	d	d^*
RCV1-4	4	665,265	19,806	47,236	73
RCV1-53	53	518,571	15,564	47,236	65
ODP	11	569,068	142,267	61,016	190

RCV1 We construct our RCV1-4 and RCV1-53 datasets from the Reuters RCV1 collection [9]. To generate the RCV1-4 dataset with four classes, we map the dataset to the first level {CCAT, ECAT, GCAT, MCAT} and remove

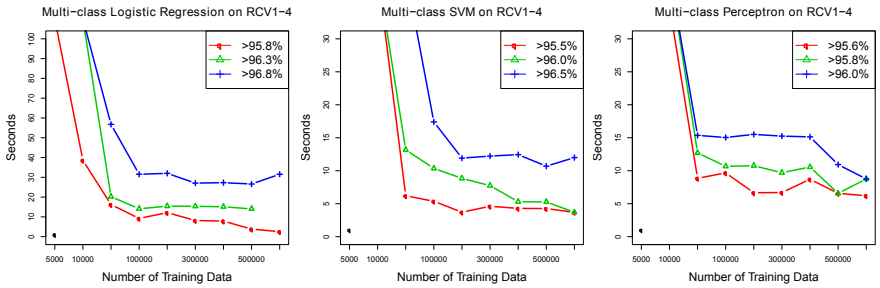

Figure 1. Inverse Dependency of 3 Algorithms on RCV1-4 Dataset.

Figure 2. Inverse Dependency of 3 Algorithms on ODP Dataset.

those documents which have more than one label. Similarly, we map the data set to the second level of the RCV1 topic hierarchy and remove the documents that only have labels of the first level or with multiple labels. This generates a 53-class dataset: RCV1-53.

ODP The original dataset has 714,875 samples with 3,637,434 features. Since the number of features is too large, we performed feature selection on the original dataset using document frequency (DF). We computed the document frequency for each feature and removed those features of frequency < 100 and removed documents with no features. We choose 80% of the samples for training and the others for testing. Consequently, we get a new dataset that includes $569,068$ training samples and $142,267$ testing samples with $61,016$ features.

B. Inverse Dependency Experiment

In this experiment, we verify the inverse dependency property of our algorithms on different datasets. It is worth noting that due to the randomness of the stochastic algorithm, we run all following experiments 10 times and take the average results. We verify the inverse dependency from two different perspectives. The first one is to measure the running time in seconds against various training data size, until the algorithms reach a certain accuracy on the testing set. We show the experimental results in Fig.1, which demonstrates that when the accuracy is fixed, the training time decreases as the size of training data increases, and this trend is consistent for different accuracy. The other experiment is to measure the the accuracy against various training data sizes until the algorithms finish various fixed number iterations. We present the results in Fig.2, which illustrates that given the same number of iterations, the accuracy of our algorithms increases as the size of training data increases. Both of the above results fully substantiate the theoretical result of inverse time dependency.

C. Comparison with OVR Method

In this experiment, we compare the accuracy of multi-class LR, multi-class SVM and the general OVR methods.To make the experiment more solid, we chose the 2-class SVM classifier Pegasos [1] and 2-class logistic regression classifier in [2] with the 2-norm regularizer, which have both been proved effective and satisfy inverse dependency. For simplification, we call the above 2 algorithms "K-Pegasos" and "K-binaryLR" respectively. The time complexity to train K binary classifiers of K-Pegasos and K-binaryLR is the same as our algorithms. Thus, in this experiment we fix the parameters λ, r, and then compare the accuracy of the algorithms against a set of distinct number of iterations. Fig.3 gives the results of the four algorithms on the three datasets. These results clearly indicate that our multi-class LR can outperform K-binaryLR and multi-class SVM can outperform K-Pegasos in the above datasets.

D. Comparison with SVM^{multiclass}

The state-of-the-art multi-class SVM classifier SVM^{multiclass} [8] uses the same formulation as our multi-class SVM classifier and its runtime scales linearly with the number of training examples. We test our program on three datasets against the solution generated by SVM^{multiclass} and compare their performance. After running the SVM^{multiclass} on the datasets, we find that SVM^{multiclass} cannot be adapted to large-scale datasets very well. It causes the "out of memory" error on all three datasets in a server with 32G RAM. To ensure a comparison between these two algorithms, we take half of the training samples of RCV1-4 and RCV1-53 for training. The experimental results are presented in Fig.4, which illustrates that our SVM classifier has an advantage in time efficiency over SVM^{multiclass} in achieving the same

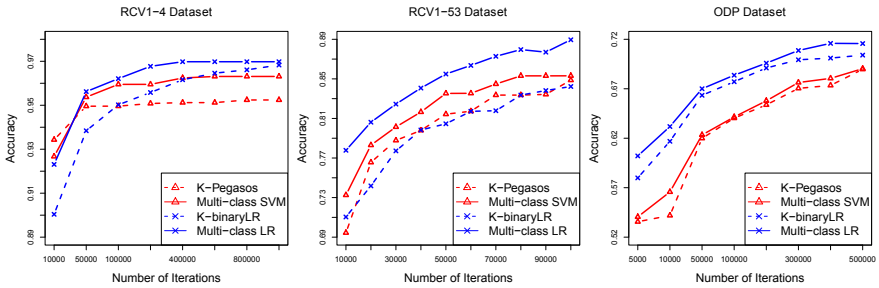

Figure 3. The Accuracy of 4 algorithms on 3 Datasets

accuracy. For example, SVM^{multiclass} takes 77.88 seconds to achieve 97.2% accuracy on RCV1-4 while our SVM classifier only takes 7.28 seconds. This may be attributed to the time complexity difference that our algorithms have a sub-linear complexity while SVM^{multiclass} incurs a linear complexity. We conclude that our algorithm performs faster on large-scale datasets than SVM^{multiclass}.

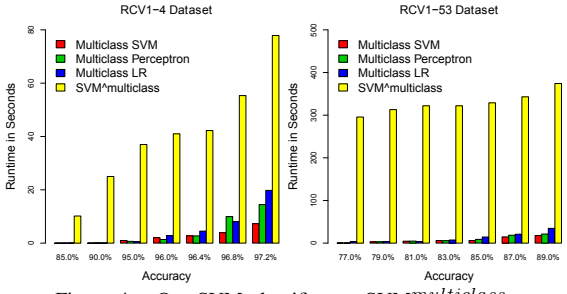

Figure 4. Our SVM classifier vs. SVM^{multiclass}.

VI. Conclusion

In this paper, we have studied the inverse dependency property for multi-class classification problems and applied it to three popular learning algorithms. We successfully demonstrated the algorithms' inverse dependency property theoretically and empirically. As a surprising result, our algorithms can train a multi-class classifier very efficiently with an order-of-magnitude speed-up as compared with the state-of-the-art multi-class classifier while ensuring a very high accuracy.

VII. Discussion & Future Work

Based on the oracle analysis in Eq.20, as we fix λ and $\hat{T}$, the upper bound of the expected error of the solution $\hat{\mathbf{w}}$ should monotonously decrease as we increase the training data size. However, if we examine the experimental results in Fig.1 and Fig.2 in more detail, we can observe an inconsistent phenomenon: when the size of the training data is small, the decrease of the training time is significant as we increase the training data; on the other hand, as the size of the training data reaches some value, the training time no longer declines with the increase of the training data i.e., the lines in above figures keep flat or fluctuating as we increase the training data after this value. This may contradict with the inverse time-depenency property. Therefore, to explore the applicability of inverse dependency with respect to the theoretical assumptions and the statistical properties of real datasets is an important but still under-explored problem. And this is one of our future work to further explore inverse dependency.

Acknowledgment

This work was supported in part by the National Basic Research Program of China Grant 2007CB807900, 2007CB807901, the National Natural Science Foundation of China Grant 61033001, 61061130540, 61073174, and the Hong Kong RGC grant 621010.

References

[1] S. Shalev-Shwartz, Y. Singer, and N. Srebro, "Pegasos: Primal estimated sub-gradient solver for svm," in *ICML*, 2007.

[2] Z. Zhu, W. Chen, C. Zhu, G. Wang, H. Wang, and Z. Cheng, "Inverse time dependency in convex regularized learning," in *ICDM*, 2009.

[3] K. Sridharan, N. Srebro, and S. Shalev-Shwartz, "Fast rates for regularized objectives," in *NIPS*, 2008.

[4] L. Bottou and O.Brousquet, "The tradeoff of large scale learning," in *NIPS*, 2007.

[5] K. Crammer and Y. Singer, "On the algorithmic implementation of multiclass kernel-based vector machines," *Technical report, School of Computer Science and Engineering, Hebrew University*, 2001.

[6] S. Kakade and S. Shalev-Shwartz, "Mind the duality gap: Logarithmic regret algorithms for online optimization," in *NIPS*, 2009.

[7] S. Shalev-Shwartz and N. Srebro, "Svm optimization: Inverse dependency on training set size," in *ICML*, 2008.

[8] T. Joachims, "Multi-class support vector machine," http://svmlight.joachims.org/svm_multiclass.html.

[9] D. Lewis, Y. Yang, T. Rose, and F. Li, "Rcv1: A new benchmark collection for text categorization research," *Journal of Machine Learning Research*, 2004.