*International Journal of Recent Engineering Science (IJRES),*
ISSN: 2349-7157, *Volume4 Issue 2 March to April 2017*

# Log-loss SVM Classification for Imbalanced Data

Shuxia Lu, Mi Zhou
*(Key Lab. of Machine Learning and Computational Intelligence, College of Mathematics and Information Science, Hebei University, China)*

***Abstract***

*SVM generally makes the separator incline to the minority class, and this leads to the problem that the minority class examples are more easily misclassified than the majority class examples for imbalanced classification problem. In order to deal with the large-scale imbalanced data classification problems, a method named stochastic gradient descent algorithm for SVM with log-loss function is proposed. To resist the separator incline, we define the weight according to the size of positive and negative dataset. Then, a weighted stochastic gradient descent algorithm is proposed to solve large-scale SVM classification. Experimental results on real datasets show that the proposed method is effective and can be used in many applications.*

**Keywords**-*Stochastic gradient descent, Weight, Imbalanced data, Log-loss function, Support vector machines*

## I. INTRODUCTION

As one of the most popular classification algorithms, support vector machine (SVM) can be seen as a learning approach trying to maximize the margin on the training data. It acquires a good theoretical foundation for the generalization performance from margin theory. CVM and SMO are the successful methods to train SVM from large data. Pegasos [1] performed stochastic gradient descent (SGD) on the primal objective with a carefully chosen step size, and SGD can be used for online training of SVM. Krzysztof [2] proposed stochastic gradient descent with Barzilai-Borwein update step for SVM. Wang [3] gives budgeted stochastic gradient descent for large-scale SVM training. This is achieved by controlling the number of SVs through one of the several budget maintenance strategies. Nicolas [4] proposed a bi-level stochastic gradient for large-scale support vector machine with automatic selection of the hyperparameter. Recently there are many improved approaches for SGD [5-11], such as quasi-Newton stochastic gradient descent, accelerated proximal stochastic dual coordinate ascent, stochastic dual coordinate ascent methods, SGD based on smart sampling techniques. Though SVM are state–of–the-art methods whether on theoretical results or many experimental results, it is not satisfactory when the training data is imbalanced.

Many researchers have worked to solve the problem of imbalanced data classification so that the classification performance of the majority class and that of minority class are good at the same time. There are three kinds of methods: data-processing, algorithmic approach, and boosting approach [12-15]. Data-processing is based on sampling method, and the second type is based on sample weighting method. The third type uses the cost of misclassifications to update the training distribution on successive boosting rounds.

SVM generally makes the separator incline to the minority class, and this leads to the problem that the minority class examples are more easily misclassified than the majority class examples. The minority class is usually more interesting or costly.

In this paper, we focus on the large and imbalanced datasets effective classification problem, a stochastic gradient descent algorithm for SVM with log-loss is proposed. We replace the hinge-loss function by the log-loss function in SVM problem. To resist the separator incline, we define the weight according to the size of positive and negative dataset. Then, a weighted stochastic gradient descent algorithm is proposed to solve large-scale SVM classification. Experiments on large classification datasets also demonstrated that the proposed method has comparable performance.

## II. PEGASOS ALGORITHM

We describe a simple algorithm of stochastic sub-gradient descent for SVM, which is Pegasos [1]. The runtime of Pegasos algorithm does not depend on the number of training examples and thus this algorithm is especially suited for large datasets.

Consider a binary classification problem with examples $S=\{(\mathbf{x}_i, y_i),\ \ i=1,\cdots,N\}$ , where instance $\mathbf{x}_i \in R^d$ is a $d$-dimensional input vector and $y_i \in \{+1,-1\}$ is the label. Training an SVM classifier $f(\mathbf{x})=\mathrm{sgn}(\mathbf{w}^T\mathbf{x})$ using $S$, where $\mathbf{w}$ is a vector associated with each input, which is formulated as solving the following optimization problem

$$\min\ \ p_t(\mathbf{w})=\frac{\lambda}{2}\|\mathbf{w}\|^2+l(\mathbf{w};(\mathbf{x}_t,y_t))\ , \qquad (1)$$

where $l(\mathbf{w};(\mathbf{x}_t,y_t))=\max(0,1-y_t\mathbf{w}^T\mathbf{x}_t)$ is the *hinge loss* function and $\lambda \geq 0$ is a regularization parameter

used to control model complexity.

SGD works iteratively. It starts with an initial guess of the model weight $\mathbf{w}_1$, and at *t*-th round it updates the current weight $\mathbf{w}_t$ as

$$\begin{aligned}\mathbf{w}_{t+1} &= \mathbf{w}_t - \eta_t \nabla_t p_t(\mathbf{w}_t) \\ &= (1-\eta_t\lambda)\mathbf{w}_t + \eta_t \mathbf{1}[\mathrm{y}_t \langle \mathbf{w}_t, \mathbf{x}_t \rangle] \mathrm{y}_t \mathbf{x}_t\end{aligned} \quad (2)$$

where

$$\mathbf{1}[\mathrm{y}_t \langle \mathbf{w}_t, \mathbf{x}_t \rangle < 1] = \begin{cases} 1, & if\ \mathrm{y}_t \langle \mathbf{w}_t, \mathbf{x}_t \rangle < 1 \\ 0, & otherwise. \end{cases} \quad (3)$$

which is the indicator function which takes a value of one if its argument is true (**w** yields non-zero loss on the example (**x**, y)), and zero otherwise. We then update using a step size of $\eta_t = 1/(\lambda \mathrm{t})$. After a predetermined number *T* of iterations, we output the last iterate $\mathbf{w}_{T+1}$.

Then, the decision function for SVM with SGD is as follows

$$f_{t+1}(\mathbf{x}) = \mathrm{sgn}(\mathbf{w}_{t+1}{}^T \mathbf{x})$$

Pegasos algorithm also extends to non-linear kernels while working solely on the primal objective function, in this case the runtime does depend linearly on the training size.

## III. LOG-LOSS SVM CLASSIFICATION FOR IMBALANCED DATA

In order to deal with the large-scale imbalanced data classification problems, we propose algorithms of stochastic gradient descent for SVM with log-loss function. We replace the hinge-loss function by the log-loss function in SVM problem, log-loss function can be regarded as a maximum likelihood estimate.

The weighted linear stochastic gradient descent for SVM with log-loss (WLSGD)

Training an SVM classifier using *S*, which is formulated as solving the following optimization problem

$$\min \ p_t(\mathbf{w}) = \frac{\lambda}{2}\|\mathbf{w}\|^2 + s_t \cdot l\left(\mathbf{w}; (\mathbf{x}_t, y_t)\right), \quad (4)$$

where

$$l(\mathbf{w}; (\mathbf{x}_t, \mathrm{y}_t)) = \log(1 + \exp(-\mathrm{y}_t, < \mathbf{w}, \mathbf{x}_t >)) \quad (5)$$

is the *log-loss* function and $\lambda \geq 0$ is a regularization parameter used to control model complexity. The log-loss is a convex function whose gradient w.r.t. **w** satisfies $\|\nabla l\| \leq \|\mathbf{x}\|$. To resist the separator incline, we define the weight $s_t$ according to the size of positive and negative dataset.

SGD works iteratively. It starts with an initial guess of the model weight $\mathbf{w}_1$, and at *t*-th round it updates the current weight $\mathbf{w}_t$ as

$$\begin{aligned}\mathbf{w}_{t+1} &= \mathbf{w}_t - \eta_t \nabla_t p_t(\mathbf{w}_t) \\ &= (1-\eta_t\lambda)\mathbf{w}_t + s_t\eta_t \frac{1}{\exp(\mathrm{y}_t < \mathbf{w}_t, \mathbf{x}_t >) + 1} \mathrm{y}_t \mathbf{x}_t\end{aligned} \quad (6)$$

We then update using a step size of $\eta_t = 1/(\lambda \mathrm{t})$. After a predetermined number *T* of iterations, we output the last iterate $\mathbf{w}_{T+1}$.

Then, the decision function for WLSGD is as follows

$$f_{t+1}(\mathbf{x}) = \mathrm{sgn}(\mathbf{w}_{t+1}{}^T \mathbf{x}) \quad (7)$$

To deal with imbalanced dataset, we simply set the weight according to the size of positive and negative dataset. The data in the majority class have to receive lower weight than those in the minority class receives.

When the size of positive dataset is $N_{pos}$ and that of negative dataset is $N_{neg}$, the weights are defined as

$$s_i = \begin{cases} 1/N_{pos} & if\ y_i = 1, \\ 1/N_{neg} & otherwise. \end{cases} \quad (8)$$

The weighted linear stochastic gradient descent for SVM with log-loss (WLSGD) is given in algorithm 1.

**Algorithm 1** WLSGD

1. Input: data *S*, regularization parameter $\lambda$, a predetermined number *T* of iterations ;
2. Initialize**:** $\mathbf{w}_1 = \mathbf{0}$ ;
3. Compute the weight $s_t$ according to the formulation (8);
4. for $t = 1, \cdots, T$ do
5. choose $(\mathbf{x}_t, \mathrm{y}_t)$ uniformly at random;
6. $\mathbf{w}_{t+1} \leftarrow (1-\eta_t\lambda)\mathbf{w}_t$
7. $\mathbf{w}_{t+1} \leftarrow \mathbf{w}_{t+1} + s_t\eta_t \frac{1}{\exp(\mathrm{y}_t < \mathbf{w}_t, \mathbf{x}_t >) + 1} \mathrm{y}_t \mathbf{x}_t$ // compute $\mathbf{w}_{t+1}$ according to the formulation (6)
8. end for
9. Output**:** $\mathbf{w}_{T+1}$ .

The weighted kernelized stochastic gradient descent for SVM with log-loss (WKSGD)

SGD for SVM can be used to solve non-linear problems when combined with Mercer kernels. After introducing a nonlinear function $\phi$ that maps **x** from the input to the feature space and replacing **x** with $\phi(\mathbf{x})$, the optimization problem can be described as

$$\min \ p_t(\mathbf{w}) = \frac{\lambda}{2}\|\mathbf{w}\|^2 + s_t \cdot l\left(\mathbf{w}; (\phi(\mathbf{x}_t), y_t)\right) \quad (9)$$

where $l(\mathbf{w}; (\mathbf{x}_t, \mathrm{y}_t)) = \log(1 + \exp(-\mathrm{y}_t, < \mathbf{w}, \mathbf{x}_t >))$ is the *log-loss* function.

It starts with an initial $\mathbf{w}_1 = \mathbf{0}$, update a step size $\eta_t = 1/(\lambda \mathrm{t})$, and for all *t* we can rewrite $\mathbf{w}_{t+1}$ as

$$
\begin{aligned}
&\mathbf{w}_{t+1} = \mathbf{w}_t - \eta_t \nabla_t p_t(\mathbf{w}_t) \\
&= (1-\lambda\eta_t)\mathbf{w}_t + s_t\eta_t \frac{1}{\exp(\mathrm{y}_t \langle \mathbf{w}_t, \phi(\mathbf{x}_t)\rangle)+1} \mathrm{y}_t \phi(\mathbf{x}_t) \\
&\square\ (1-\lambda\eta_t)\mathbf{w}_t + \beta_t\phi(\mathbf{x}_t) \\
&= (1-\lambda\eta_t)(1-\lambda\eta_{t-1})\mathbf{w}_{t-1} + (1-\lambda\eta_t)\beta_{t-1}\phi(\mathbf{x}_{t-1}) + \beta_t\phi(\mathbf{x}_t) \\
&= \cdots \\
&= \prod_{k=2}^{t}(1-\lambda\eta_k)\beta_1\phi(\mathbf{x}_1) + \prod_{k=3}^{t}(1-\lambda\eta_k)\beta_2\phi(\mathbf{x}_2) + \cdots + \\
&\quad (1-\lambda\eta_t)\beta_{t-1}\phi(\mathbf{x}_{t-1}) + \beta_t\phi(\mathbf{x}_t) \\
&= \frac{1}{t}\beta_1\phi(\mathbf{x}_1) + \frac{2}{t}\beta_2\phi(\mathbf{x}_2) + \cdots + \frac{t-1}{t}\beta_{t-1}\phi(\mathbf{x}_{t-1}) + \beta_t\phi(\mathbf{x}_t) \\
&= \sum_{j=1}^{t}\frac{j}{t}\beta_j\phi(\mathbf{x}_j)
\end{aligned}
\tag{10}
$$

where

$$
\begin{aligned}
\beta_j &= s_j\eta_j \frac{1}{\exp(\mathrm{y}_j \langle \mathbf{w}_j, \phi(\mathbf{x}_j)\rangle)+1} \mathrm{y}_j \\
&= s_j \frac{1}{\lambda j} \frac{1}{\exp(\mathrm{y}_j \left\langle \sum_{i=1}^{j-1}\frac{i}{t}\beta_i\phi(\mathbf{x}_i), \phi(\mathbf{x}_j)\right\rangle)+1} \mathrm{y}_j \\
&= s_j \frac{1}{\lambda j} \frac{1}{\exp(\mathrm{y}_j \sum_{i=1}^{j-1}\frac{i}{t}\beta_i \mathrm{k}(\mathbf{x}_i, \mathbf{x}_j))+1} \mathrm{y}_j \\
\beta_1 &= s_1\eta_1 \frac{1}{\exp(\mathrm{y}_1 \langle \mathbf{w}_1, \phi(\mathbf{x}_1)\rangle)+1} \mathrm{y}_1 \\
&= s_1 \frac{1}{\lambda} \frac{1}{\exp(\mathrm{y}_1 \langle \mathbf{0}, \phi(\mathbf{x}_1)\rangle)+1} \mathrm{y}_1 = s_1 \frac{y_1}{2\lambda}
\end{aligned}
\tag{11}
$$

Then, the decision function for WKSGD is as follows:

$$
f_{t+1}(\mathbf{x}) = \mathrm{sgn}(\mathbf{w}_{t+1}{}^T\phi(\mathbf{x})) = \mathrm{sgn}(\frac{1}{t}\sum_{j=1}^{t} j\beta_j k(\mathbf{x}_j, \mathbf{x}))
\tag{12}
$$

Only one element of $\beta$ is changed at each iteration. The algorithm does not refer to the implicit mapping $\phi(\cdot)$ and only use the kernel function. This WKSGD implementation is given in algorithm 2.

**Algorithm 2** WKSGD

1. Input: data *S*, regularization parameter $\lambda$ , a predetermined number *T* of iterations ;
2. Initialize: $\mathbf{w}_1 = \mathbf{0}$
3. for $t = 1, \cdots, T$ do
4. Choose $(\mathbf{x}_t, \mathrm{y}_t)$ uniformly at random;
5. Set $\eta_t = 1/(\lambda \mathrm{t})$;
6. Compute the weight $s_t$ according to the formulation (8);
7. Compute $\beta_t$ according to the formulation (11);
8. end for
9. Output: $\beta_T$ .

## IV. EXPERIMENTAL RESULTS

In this section, we conduct the performance comparison of the four methods for real problems: MNIST, Ijcnnl, Shuttle, Letter, Usps, Adult. Most of the datasets are taken from the UCI machine learning repository [16]. Usps is taken from database [17]. The multi-classification dataset are artificially divided into binary classification dataset, which constitute the imbalanced dataset. The description of datasets is shown in Table 1.

**Table 1 Characteristics of Datasets**

| Datasets | #Classes | #Training | #Testing ( $N_{pos}$、$N_{neg}$, ) | #features |
|---|---|---|---|---|
| MNIST | 10 | 60000 | 10000 ( 974、9026 ) | 780 |
| Ijcnnl | 2 | 91701 | 49990 ( 4853、45137 ) | 22 |
| Shuttle | 7 | 43500 | 14500 ( 2155、12345 ) | 9 |
| Letter | 26 | 10000 | 10000 ( 353、9647 ) | 16 |
| Usps | 10 | 7291 | 2007 ( 198、1809 ) | 256 |
| Adult | 2 | 24974 | 12554 ( 119、12435 ) | 123 |

In our experiments, LSGD is Pegasos in constructing linear SVMs, KSGD is the kernelized Pegasos. WLSGD and WKSGD are proposed two methods: the weighted linear stochastic gradient descent for SVM with log-loss (WLSGD), and The weighted kernelized stochastic gradient descent for SVM with log-loss (WKSGD). The Gaussian function is taken as the kernel function $k(x_i, x_j) = \exp(-\|x_i - x_j\|^2 / \sigma^2)$. Set the kernel function width $\sigma$ =1.5, a predetermined number of iterations $T = 10^6$. The regularization parameters and $\varepsilon$ are shown in Table 2.

**Table 2 Parameter Setting**

| Algorithms | $\varepsilon$ | $\lambda$ |
|---|---|---|
| LSGD | $10^{-6}$ | $10^{-4}$ |
| KSGD | $10^{-3}$ | $10^{-4}$ |
| WLSGD | $10^{-6}$ | $10^{-4}$ |
| WKSGD | $10^{-3}$ | $10^{-4}$(Mnist)<br>$10^{-4}$(Adult)<br>$10^{-9}$(Ijcnn)<br>$10^{-8}$(Shuttle)<br>$10^{-7}$(Letter)<br>$10^{-5}$(Usps) |

Considering the imbalanced nature of the training datasets, the geometric mean accuracy (G-mean) is adopted to evaluate the performance of our algorithms,

$$g = \sqrt{a^+ \cdot a^-}$$

where

$$a^+ = \frac{\#\text{positive samples correctly classified}}{\#\text{ total positive samples classified}} \times 100\%,$$

$$a^- = \frac{\#\text{negative samples correctly classified}}{\#\text{ total negative samples classified}} \times 100\%.$$

**Table 3 Comparison of the Testing G-Mean**

| Algorithms | LSGD | WLSGD | KSGD | WKSGD |
|---|---|---|---|---|
| MNIST | 78.86 | 71.61 | 87.60 | 93.75 |
| Ijcnnl | 51.42 | 78.79 | 55.14 | 72.40 |
| Shuttle | 0.00 | 48.84 | 58.15 | 92.43 |
| Letter | 23.14 | 77.96 | 75.63 | 90.21 |
| Usps | 90.83 | 90.63 | 94.66 | 95.11 |
| Adult | 0.00 | 78.33 | 9.16 | 74.70 |

Twenty trials were conducted for the four algorithms and the average results are shown in Table 3 and Table 4. Table 3 shows the performance comparison of the testing G-mean for the real-world problems; the testing G-mean of WLSGD and WKSGD is higher than LSGD and KSGD methods in most datasets. Table 4 shows the performance comparison of average training and testing time of the four methods for the real-world problems. As observed from the Table 4, WLSGD compare to LSGD with almost same learning speed in most datasets. The runtime of WKSGD is slightly more than that of KSGD.

**Table 4 Time Cost Comparison of the Average Training and Testing Time (s)**

| Algorithms | LSGD | WLSGD | KSGD | WKSGD |
|---|---|---|---|---|
| MNIST | 597.53 | 633.60 | 20.94 | 24.00 |
| | 6.15 | 6.07 | 777.34 | 815.03 |
| Ijcnnl | 22.94 | 2.25 | 0.20 | 0.83 |
| | 1.11 | 1.12 | 31.91 | 166.84 |
| Shuttle | 11.75 | 1.81 | 0.11 | 0.48 |
| | 0.16 | 0.16 | 5.03 | 26.04 |
| Letter | 21.84 | 21.14 | 0.08 | 0.74 |
| | 0.21 | 0.20 | 1.90 | 28.20 |
| Usps | 277.26 | 277.25 | 7.84 | 9.56 |
| | 0.56 | 0.54 | 58.74 | 74.41 |
| Adult | 128.71 | 101.92 | 0.45 | 2.67 |
| | 1.39 | 1.15 | 18.85 | 129.84 |

## V. CONCLUSION

We focus on the large and imbalanced datasets effective classification problem, a method named stochastic gradient descent algorithm for SVM with log-loss function is proposed. We replace the hinge-loss function by the log-loss function in SVM problem. To resist the separator incline, we define the weight according to the size of positive and negative dataset. Then, a weighted stochastic gradient descent algorithm is proposed to solve large-scale SVM classification. Experimental results on real datasets show that the proposed method is effective and can be used in many applications.

## ACKNOWLEDGEMENTS

This research is supported by the natural science foundation of Hebei Province No. F2015201185.

## REFERENCES

[1] Shalev-Shwartz, Y. Singer, N. Srebro, et al, Pegasos: Primal Estimated sub-Gradient Solver for SVM, Mathematical Programming, 127(1), 2011, 3-30.
[2] Krzysztof Sopyla, Pawel Drozda, Stochastic Gradient Descent with Barzilai-Borwein update step for SVM, Information Sciences, 316, 2015, 218-233.
[3] Zhuang Wang, Koby Crammer, Slobodan Vucetic. Breaking the Curse of Kernelization: Budgeted Stochastic Gradient Descent for Large-Scale SVM Training. Journal of Machine Learning Research, 13, 2013, 3103-3131.
[4] Nicolas Couellan, Wenjuan Wang. Bi-level stochastic gradient for large scale support vector machine. Neurocomputing, 153, 2015300-308.
[5] A. Bordes, L. Bottou, P. Gallinari. SGD-QN: careful quasi-Newton stochastic gradient descent. J. Mach. Learn, 10, 2009, 1737-1754.
[6] A. Bordes, L. Bottou, P. Gallinari, et al. Sgdqn is less careful than expected. J. Mach. Learn, 11, 2010, 2229-2240.
[7] Vijay Manikandan Janakiraman, XuanLong Nguyen, Dennis, et al. Stochastic gradient based extreme learning machines for stable online learning of advanced combustion engines. Neurocomputing, 177, 2016, 304-316.
[8] Shai Shalev-Shwartz, Tong Zhang. Accelerated proximal stochastic dual coordinate ascent for regularized loss minimization. Math. Program, 155, 2016, 105-145.
[9] Shalev-Shwartz, Zhang, et al. Stochastic dual coordinate ascent methods for regularized losss minimization. J. Mach. Learn, 14, 2013, 567-599.
[10] Stephan Clemencon, Aurelien Bellet, Ons Jelassi, et al. Scalability of Stochastic Gradient Descent based on Smart Sampling Techniques. Procedia Computer Science, 53, 2015, 308–315.
[11] Elad Hazan, Satyen Kale. Beyond the Regret Minimization Barrier: Optimal Algorithms for Stochastic Strongly Convex Optimization. Journal of Machine Learning Research, 15, 2014, 2489-2512.
[12] Kuan Li, Xiangfei Kong, Zhi Lu, et al. Boosting weighted ELM for imbalanced learning. Neurocomputing, 128, 2014, 15-21.
[13] H. He, E. A. Garcia. Learning from imbalanced data. IEEE Trans. Knowl. Data Eng, 21 (9), 2009, 1263–1284.
[14] Y. M. Sun, K. C. Wong, and M.S. Kamel. Classification of imbalanced data: a review. International journal of pattern recognition and artificial intelligence, 23(4), 2009, 687-719.
[15] Weiwei Zong, GB Huang, Yiqiang Chen. Weighted extreme learning machine for imbalance learning. Neurocomputing, 101, 2013, 229-242.
[16] A. Frank, A. Asuncion, UCI machine learning repository, 2010. Available from http: //archive. ics. uci. Edu / ml.
[17] J. J. Hull. A database for handwritten text recognition research. IEEE Trans. Pattern Anal. Mach. Intell., 16 (5), 1994, 550-554.