Linear and Nonlinear Analysis
Volume 4, Number 1, 2018, 29–61

Yokohama Publishers
ISSN 2188-8167
Copyright 2018

EVALUATION OF STOCHASTIC APPROXIMATION ALGORITHM AND VARIANTS FOR LEARNING SUPPORT VECTOR MACHINES

KEIGO FUJIWARA, KAZUHIRO HISHINUMA, AND HIDEAKI IIDUKA

Abstract. This paper considers a convex stochastic optimization problem and presents several variants of an algorithm for solving it that is defined by combining an existing stochastic approximation algorithm for convex stochastic optimization with search directions for unconstrained nonconvex optimization. It shows that, under certain assumptions, the expected error of the current solution in terms of the distance to the solution of the problem is of order $\mathcal{O}(n^{-1/2})$, where n is the number of iterations. The variants of the proposed algorithm are numerically compared with the Pegasos machine learning algorithm in binary and multiclass classification experiments using LIBSVM datasets. The results show that the proposed stochastic optimization algorithm and its variants have higher classifier accuracy; in particular, the results of t-tests show that their average performance is significantly different from that of Pegasos.

1. Introduction

This paper considers a convex stochastic optimization problem [10, (1.1)] for minimizing the expectation of convex functions. This problem appears in subproblems of learning support vector machines. Support vector machines are an effective and widely used classification learning tool [15, Section 1]. The task of learning a support vector machine can be expressed as a constrained quadratic loss optimization problem [15, Section 1]. The task is an empirical loss minimization problem with a penalty term for the norm of the classifier that is being learned [15, Problem (1)].

There are several practical optimization algorithms for solving a convex stochastic optimization problem. The stochastic approximation (SA) algorithm [2, 10, 13] for example, uses the subgradients of randomly chosen objective functions. The primal estimated sub-gradient solver (Pegasos) algorithm for support vector machines [15] is an SA algorithm programmed for learning support vector machines [15]. It is

2010 *Mathematics Subject Classification.* 90C15, 90C25.

Key words and phrases. Convex stochastic optimization, stochastic approximation, support vector machine learning.

This work was supported by the Japan Society for the Promotion of Science through a Grant-in-Aid for Scientific Research (C) (JP15K04763) and Grant-in-Aid for JSPS Research Fellow (JP17J09220).

Many iterative algorithms have been presented for unconstrained smooth optimization to accelerate the steepest descent method. Two examples are the conjugate gradient algorithm [7] [11, Chapters 5] and the *three-term conjugate gradient algorithm* [3–6, 9]. In this paper, we present a novel algorithm (plus several variants) obtained by combining the SA algorithm with the three-term conjugate gradient algorithm. The proposed algorithm (plus variants) shortens the computation time and finds better solutions than the Pegasos learning algorithm. We show that, under certain assumptions, the expected error of the current solution in terms of the distance to the solution of the problem is of order $\mathcal{O}(n^{-1/2})$, where n is the number of iterations.

We numerically compared the proposed learning algorithm (plus variants) with the Pegasos learning algorithm by applying them to both binary and multiclass classification problems using LIBSVM machine learning datasets [8]. We compared their performances in terms of the classification scores and computation times. The results showed that our proposed algorithm (plus variants) had higher classification scores than Pegasos and that it required less computation time to learn support vector machines. Moreover, the results of t-tests showed that its average performance was significantly different from that of Pegasos.

This paper is organized as follows. Section 2 covers the mathematical preliminaries and presents the convex stochastic optimization problem considered. Section 3 presents the algorithm proposed for solving the problem. Section 4 evaluates the application of the proposed algorithm (plus 10 variants) to support vector machine learning using 15 LIBSVM machine learning datasets [8] in comparison with that of the Pegasos learning algorithm. Section 5 concludes the paper with a summary of the key points.

2. Mathematical preliminaries

2.1. **Notation and definitions.** Let $\mathbb{R}^N$ be an N-dimensional Euclidean space with inner product $\langle \cdot, \cdot \rangle$ and its induced norm $\| \cdot \|$. Let $\mathbb{N}$ be the set of all positive integers including zero. A mapping $T \colon \mathbb{R}^N \to \mathbb{R}^N$ is said to be *nonexpansive* [1, Definition 4.1(ii)] if it is Lipschitz continuous with constant 1; i.e., $\|T(x) - T(y)\| \leq \|x - y\|$ for all $x, y \in \mathbb{R}^N$. Given a nonempty, closed convex set $C \subset \mathbb{R}^N$, the metric projection onto C, denoted by P_C, is defined for all $x \in \mathbb{R}^N$ by $P_C(x) \in C$ and $\|x - P_C(x)\| = \inf_{y \in C} \|x - y\|$. Let $c > 0$. A function $f \colon \mathbb{R}^n \to \mathbb{R}$ is said to be *c-strongly convex* [1, Definition 10.5] if, for all $x, y \in \mathbb{R}^n$ and for all $\alpha \in (0,1)$, $f(\alpha x + (1-\alpha)y) + (c\alpha(1-\alpha)/2)\|x-y\|^2 \leq \alpha f(x) + (1-\alpha) f(y)$. The *subdifferential* [1, Definition 16.1, Corollary 16.14] of a convex function $f \colon \mathbb{R}^N \to \mathbb{R}$ is the set-valued operator ∂f defined for all $x \in \mathbb{R}^N$ by $\partial f(x) = \{u \in H \colon f(y) \geq f(x) + \langle y - x, u\rangle \ (y \in \mathbb{R}^N)\} \neq \emptyset$. Let $\mathcal{O}$ denote Landau's symbol; i.e., $y_n = \mathcal{O}(x_n)$ if there exist $c > 0$ and $n_0 \in \mathbb{N}$ such that $y_n \leq c x_n$ for all $n \geq n_0$.

2.2. **Convex stochastic optimization problem.** In this paper, we discuss the following convex stochastic optimization problem [10, (1.1)]:

Problem 2.1. Assume that

(A1) $f^{(m)} \colon \mathbb{R}^N \to \mathbb{R}$ ($m \in \mathcal{M} := \{1, 2, \ldots, M\}$) is convex;

(A2) $C \subset \mathbb{R}^N$ is a nonempty, bounded, closed convex set onto which the metric projection P_C can be efficiently computed.

Then

$$\text{minimize } f(w) := \mathbb{E}\left[f^{(\xi)}(w)\right] \text{ subject to } w \in C,$$

where $f^{(\xi)}$ is a function involving a random variable $\xi \in \mathcal{M}$, and one assumes that

(i) the expectation $\mathbb{E}[f^{(\xi)}(w)]$ is well defined and finite valued for all $w \in \mathbb{R}^N$, and f is Lipschitz continuous and strongly convex with a constant c;

(ii) there is an independent identically distributed sample $\xi_0, \xi_1, \ldots$ of realizations of the random variable ξ;

(iii) there is an oracle such that, for $(w, \xi) \in \mathbb{R}^N \times \mathcal{M}$, it returns a stochastic subgradient $\mathsf{G}^{(\xi)}(w)$ such that $\mathsf{g}(w) := \mathbb{E}[\mathsf{G}^{(\xi)}(w)]$ is well defined and is a subgradient of f at w.

An application of Problem 2.1 is the support vector machine (SVM) optimization problem defined as follows: given $\lambda > 0$ and a training set $\{(x_m, y_m)\}_{m=1}^M$, where $x_m \in \mathbb{R}^N$ and $y_m \in \{-1, +1\}$ $(m \in \mathcal{M})$,

$$\text{minimize } \frac{\lambda}{2}\|w\|^2 + \frac{1}{M}\sum_{m=1}^{M} \max\{0, 1 - y_m\langle w, x_m\rangle\} \text{ subject to } \|w\| \leq \frac{1}{\lambda}. \tag{2.1}$$

For all $m \in \mathcal{M}$ and for all $w \in \mathbb{R}^N$, we define $f^{(m)}(w) := (1/M)[(\lambda/2)\|w\|^2 + \max\{0, 1 - y_m\langle w, x_m\rangle\}]$. A useful algorithm for solving problem (2.1) is the primal estimated sub-gradient solver (Pegasos) algorithm [15, Figure 1]:

Algorithm 1 Pegasos [15, Figure 1]

```
Require: $\lambda > 0$
1: $n \leftarrow 0$, $w_0 \in \mathbb{R}^N$
2: loop
3:     $\alpha_n := 1/(\lambda(n+1))$
4:     $\mathsf{G}^{(\xi_n)}(w_n) \in \partial f^{(\xi_n)}(w_n)$
5:     $w_{n+1} := P_C(w_n - \alpha_n \mathsf{G}^{(\xi_n)}(w_n))$
6:     $n \leftarrow n + 1$
7: end loop
```

Step 5 in Algorithm 1 is stochastic approximation (SA) [2, (5.4.1)], [10,13] based on the stochastic subgradient direction $d_n := -\mathsf{G}^{(\xi_n)}(w_n)$. The discussion in [10, Subsection 2.1] leads to the conclusion that Algorithm 1 with $\alpha_n = \mathcal{O}(1/n)$ $(n \geq 1)$ satisfies, under the assumptions in Problem 2.1,

$$\mathbb{E}\left[\|w_n - w^\star\|^2\right] = \mathcal{O}\left(\frac{1}{n}\right),$$

where $w^\star$ is the solution to Problem 2.1.

Many iterative methods have been presented for unconstrained smooth optimization to accelerate the steepest descent method defined by $x_{n+1} := x_n + \lambda_n d_n^{\mathrm{sd}} =$

$x_n - \lambda_n \nabla h(x_n)$, where $x_0 \in \mathbb{R}^N$, $h \colon \mathbb{R}^N \to \mathbb{R}$ is differentiable, and ∇h is the gradient of h. An example method uses the *three-term conjugate gradient direction* [5,9], defined as

$$d_n^{\mathrm{ttcg}} := -\nabla h(x_n) + \beta_n d_{n-1} + \gamma_n z_n, \tag{2.2}$$

where $(z_n)_{n\in\mathbb{N}} \subset \mathbb{R}^N$ is an arbitrary sequence, and $(\beta_n)_{n\in\mathbb{R}}$ and $(\gamma_n)_{n\in\mathbb{N}} \subset \mathbb{R}$. We can see that d_n^{ttcg} defined by (2.2) with $\gamma_n := 0$ $(n \in \mathbb{N})$ coincides with the conjugate gradient direction (see [11] for details of conjugate gradient methods). From the above discussion, we derived a novel algorithm that can be obtained by combining SA (step 5 in Algorithm 1) with the three-term conjugate direction (2.2) with $\nabla h = \mathsf{G}^{(\xi_n)}$.

3. Stochastic approximation algorithm with three-term conjugate gradient direction

The proposed algorithm for solving Problem 2.1 is given as

Algorithm 2 Stochastic approximation algorithm for Problem 2.1

```
Require: $(\alpha_n)_{n\in\mathbb{N}} \subset [0,1]$, $(\beta_n)_{n\in\mathbb{N}} \subset [0,\infty)$, $(\gamma_n)_{n\in\mathbb{N}} \subset [0,\infty)$, $(z_n)_{n\in\mathbb{N}} \subset \mathbb{R}^N$
1: $n \leftarrow 0$, $w_0 \in \mathbb{R}^N$, $d_0 := -\mathsf{G}^{(\xi_0)}(w_0)$
2: loop
3:     $d_n := -\mathsf{G}^{(\xi_n)}(w_n) + \beta_n d_{n-1} + \gamma_n z_n$
4:     $w_{n+1} := P_C(w_n + \alpha_n d_n)$
5:     $n \leftarrow n + 1$
6: end loop
```

Step 3 in Algorithm 2 is based on the three-term conjugate gradient direction (2.2). Algorithm 2 coincides with Algorithm 1 when $\alpha_n := 1/(\lambda(n+1))$ and $\beta_n = \gamma_n := 0$ $(n \in \mathbb{N})$. Moreover, Algorithm 2 is almost the same as [14, (1.5)] when $\gamma_n := 0$ $(n \in \mathbb{N})$.

The proofs of [14, Theorem 3.1] lead to the following.

Theorem 3.1. *Let $w^\star \in C$ be the solution of Problem 2.1. Assume that the assumptions in Problem 2.1 hold and there exists a positive number B such that $\mathbb{E}[\|\mathsf{G}^{(\xi)}(w)\|^2] \le B^2$ for all $(w, \xi) \in \mathbb{R}^N \times \mathcal{M}$. Then the sequence $(w_n)_{n\in\mathbb{N}}$ generated by Algorithm 2 satisfies the following conditions:*

(a) *If $\alpha_n := 1/(cn)$ and $\beta_n, \gamma_n \le 1/n$ for all $n \ge 1$, then, for all $n \ge 1$,*

$$\mathbb{E}\left[\|w_n - w^\star\|^2\right] = \mathcal{O}\left(\frac{1 + \log n}{n}\right).$$

(b) *If $\alpha_n := 2/(c(n+1))$ and $\beta_n, \gamma_n \le 1/n$ for all $n \ge 1$, then, for all $n \ge 1$,*

$$\mathbb{E}\left[\|w_n - w^\star\|^2\right] = \mathcal{O}\left(\frac{1}{n}\right).$$

Since $\beta_n = \gamma_n := 0$ $(n \in \mathbb{N})$ satisfies the conditions in Theorem 3.1(b), Algorithm 2 satisfies $\mathbb{E}[\|w_n - w^\star\|^2] = \mathcal{O}(1/n)$ when $\beta_n = \gamma_n := 0$ $(n \in \mathbb{N})$ (i.e., Algorithm 1), which implies that Theorem 3.1 is a generalization of the results in [10, Subsection 2.1].

4. Application to learning support vector machines

We evaluated the performance of the proposed algorithm (Algorithm 2 plus several variants) in comparison with that of Pegasos (Algorithm 1) for learning support vector machines by using a large number of classification problems formulated as machine learning datasets. Here we present the results and discuss the advantages and disadvantages of each variant.

4.1. **Experimental method and environment.**

The experimental programs were written in Python 3 (version 3.6.3) using the NumPy and scikit-learn [12] packages. They were run on an Apple Macbook Air with a 1.3-GHz Intel Core i5 CPU, 4-GB DDR3 memory, and the Mac OSX 10.8.5 operating system.

The classification problems corresponded to 15 machine learning datasets, as listed in Table 1. The datasets were obtained from LIBSVM datasets [8]. The *a1a*,

Table 1. Datasets used for experiments

Name	No. of classes	Training size	Testing size	No. of features
a1a	2	1,605	30,956	123
a2a	2	2,265	30,296	123
a3a	2	3,185	29,376	123
diabetes	2	576	192	8
dna	3	2,000	1,186	180
iris	3	112	38	4
svmguide1	2	3,089	4,000	4
svmguide2	3	293	98	20
wine	3	133	45	13
segment	7	1732	578	19
breast-cancer	2	512	171	10
sonar	2	156	52	60
splice	2	1,000	2,175	60
german.numer	2	750	250	24
australian	2	517	173	14

a2a, *a3a*, *dna*, *svmguide1*, and *splice* datasets were obtained as pairs of training and test datasets. We used the training datasets to learn the support vector machines and the test datasets to evaluate the performance of the learned classifiers. The remaining datasets (*diabetes*, *iris*, *svmguide2*, *wine*, *segment*, *breast-cancer*, *sonar*, *german.numer* and *australian* datasets) were not obtained as pairs. We thus divided them into training and test datasets using the **sklearn.model_selection.train_test_split** function provided in the scikit-learn package.

As preprocessing for learning classifiers, we standardized the features of the datasets using the **sklearn.preprocessing.StandardScaler** class, which removes the mean from the features and scales the features to unit variance.

We set $w_0 := (0, 0, \dots, 0)^\top$ as an initial point to each algorithm. We ran ten variants of Algorithm 2 for comparison.

Variant 1: $\alpha_n = \frac{1}{\lambda(n+1)}$, $\beta_n = 0$, $\gamma_n = 0$ (Algorithm 1);
Variant 2: $\alpha_n = \frac{1}{\lambda(n+1)}$, $\beta_n = \frac{1}{n+1}$, $\gamma_n = 0$;
Variant 3: $\alpha_n = \frac{1}{\lambda(n+1)}$, $\beta_n = \frac{1}{n+1}$, $\gamma_n = \beta_n$;
Variant 4: $\alpha_n = \frac{1}{\lambda(n+1)}$, $\beta_n = \frac{1}{(n+1)^2}$, $\gamma_n = 0$;
Variant 5: $\alpha_n = \frac{1}{\lambda(n+1)}$, $\beta_n = \frac{1}{(n+1)^2}$, $\gamma_n = \beta_n$;
Variant 6: $\alpha_n = \frac{2}{\lambda(n+2)}$, $\beta_n = 0$, $\gamma_n = 0$;
Variant 7: $\alpha_n = \frac{2}{\lambda(n+2)}$, $\beta_n = \frac{1}{n+1}$, $\gamma_n = 0$;
Variant 8: $\alpha_n = \frac{2}{\lambda(n+2)}$, $\beta_n = \frac{1}{n+1}$, $\gamma_n = \beta_n$;
Variant 9: $\alpha_n = \frac{2}{\lambda(n+2)}$, $\beta_n = \frac{1}{(n+1)^2}$, $\gamma_n = 0$;
Variant 10: $\alpha_n = \frac{2}{\lambda(n+2)}$, $\beta_n = \frac{1}{(n+1)^2}$, $\gamma_n = \beta_n$;

Variant 1 coincides with Algorithm 1 (Pegasos).

Now, we recall the task of learning support vector machines.

Problem 4.1 (Task of learning support vector machines)**.** Suppose that a training set $\{(x_m, y_m)\}_{m=1}^{M} \subset \mathbb{R}^N \times \{-1, +1\}$ and a parameter $\lambda > 0$ are given. Let $f^{(m)}(w) := (1/M)[(\lambda/2)\|w\|^2 + \max\{0, 1 - y_m\langle w, x_m\rangle\}]$ and let $C := \{x \in \mathbb{R}^N : \|x\| \leq 1/\lambda\}$. Then we would like to

$$\text{minimize } f(w) := \mathbb{E}\left[f^{(\xi)}(w)\right] \text{ subject to } w \in C.$$

We applied the variants of Algorithm 2 to Problem 4.1 for each dataset listed in Table 1. For each variant, we evaluated the computation time and classification score of the generated classifier w_n $(n = 10^3, 10^4)$.

4.2. **Experimental results when $\lambda := 1.0$.** First, we evaluated the performance of the variants when $\lambda := 1.0$ was used as the parameter of Problem 4.1. Tables 2–16 show the computation times for acquiring classifier w_n $(n = 10^3, 10^4)$ and the classification scores for the test datasets. The scores were calculated using the **score** method derived for each classifier from the **sklearn.base.ClassifierMixin** class.

Table 2. Computation times and classification scores for learning w_n $(n = 10^3, 10^4)$ for *a1a* dataset using each variant of Algorithm 2 when $\lambda := 1.0$ was used

	1,000 iterations		10,000 iterations	
Variant	Time [s]	Score [%]	Time [s]	Score [%]
1	0.18	71.59	0.48	71.10
2	0.18	71.55	0.59	70.84
3	0.18	71.35	0.58	70.94
4	0.18	71.52	0.58	71.04
5	0.28	72.03	0.48	70.98
6	0.18	70.37	0.63	70.98
7	0.17	70.53	0.48	71.08
8	0.18	69.03	0.48	70.92
9	0.18	70.97	0.48	71.12
10	0.18	71.18	0.58	70.98

Table 3. Computation times and classification scores for learning w_n $(n = 10^3, 10^4)$ for *a2a* dataset using each variant of Algorithm 2 when $\lambda := 1.0$ was used

	1,000 iterations		10,000 iterations	
Variant	Time [s]	Score [%]	Time [s]	Score [%]
1	0.18	71.35	0.48	70.89
2	0.18	71.01	0.48	70.80
3	0.29	71.73	0.69	71.10
4	0.18	70.83	0.59	71.10
5	0.17	71.17	0.58	71.11
6	0.18	71.17	0.60	71.07
7	0.17	70.10	0.47	70.96
8	0.17	70.60	0.48	70.99
9	0.17	71.06	0.48	71.36
10	0.18	71.99	0.48	71.15

Table 4. Computation times and classification scores for learning $w_n\,(n = 10^3, 10^4)$ for *a3a* dataset using each variant of Algorithm 2 when $\lambda := 1.0$ was used

	1,000 iterations		10,000 iterations	
Variant	Time [s]	Score [%]	Time [s]	Score [%]
1	0.17	70.32	0.48	70.87
2	0.28	70.79	0.58	70.81
3	0.28	70.41	0.58	71.03
4	0.28	70.78	0.59	71.27
5	0.17	70.81	0.58	70.96
6	0.28	70.72	0.59	70.80
7	0.17	70.98	0.48	70.52
8	0.18	71.03	0.58	70.85
9	0.17	70.84	0.59	70.86
10	0.28	70.47	0.58	70.65

Table 5. Computation times and classification scores for learning $w_n\,(n = 10^3, 10^4)$ for *diabetes* dataset using each variant of Algorithm 2 when $\lambda := 1.0$ was used

	1,000 iterations		10,000 iterations	
Variant	Time [s]	Score [%]	Time [s]	Score [%]
1	0.18	66.67	0.37	67.71
2	0.20	68.75	0.48	68.23
3	0.17	68.23	0.58	69.27
4	0.18	67.71	0.48	68.23
5	0.18	67.19	0.48	68.75
6	0.18	68.23	0.48	69.27
7	0.18	68.23	0.48	69.79
8	0.18	70.31	0.48	68.23
9	0.19	69.27	0.49	68.2
10	0.18	68.23	0.50	68.2

TABLE 6. Computation times and classification scores for learning w_n $(n = 10^3, 10^4)$ for *dna* dataset using each variant of Algorithm 2 when $\lambda := 1.0$ was used

	1,000 iterations		10,000 iterations	
Variant	Time [s]	Score [%]	Time [s]	Score [%]
1	0.27	86.68	0.68	87.18
2	0.29	85.92	0.89	86.76
3	0.29	85.33	1.11	87.10
4	0.28	84.99	0.88	86.68
5	0.27	84.91	0.78	86.93
6	0.28	84.57	0.79	87.02
7	0.28	84.57	0.68	86.76
8	0.28	84.82	0.78	87.10
9	0.30	85.33	0.89	87.18
10	0.28	85.08	0.88	87.10

TABLE 7. Computation times and classification scores for learning w_n $(n = 10^3, 10^4)$ for *iris* dataset using each variant of Algorithm 2 when $\lambda := 1.0$ was used

	1,000 iterations		10,000 iterations	
Variant	Time [s]	Score [%]	Time [s]	Score [%]
1	0.17	81.58	0.58	81.58
2	0.18	81.58	0.78	81.58
3	0.17	81.58	0.89	81.58
4	0.18	81.58	0.78	81.58
5	0.18	81.58	0.68	81.58
6	0.18	81.58	0.69	81.58
7	0.17	81.58	0.69	81.58
8	0.17	81.58	0.57	81.58
9	0.17	81.58	0.68	81.58
10	0.17	81.58	0.68	81.58

Table 8. Computation times and classification scores for learning $w_n\,(n = 10^3, 10^4)$ for *svmguide1* dataset using each variant of Algorithm 2 when $\lambda := 1.0$ was used

	1,000 iterations		10,000 iterations	
Variant	Time [s]	Score [%]	Time [s]	Score [%]
1	0.17	84.48	0.37	84.68
2	0.19	84.40	0.50	84.78
3	0.17	84.58	0.58	84.90
4	0.19	84.65	0.53	84.60
5	0.17	83.90	0.48	84.83
6	0.18	84.63	0.49	84.78
7	0.17	83.80	0.38	84.70
8	0.17	84.68	0.37	84.63
9	0.17	84.28	0.38	84.73
10	0.17	84.20	0.48	84.90

Table 9. Computation times and classification scores for learning $w_n\,(n = 10^3, 10^4)$ for *svmguide2* dataset using each variant of Algorithm 2 when $\lambda := 1.0$ was used

	1,000 iterations		10,000 iterations	
Variant	Time [s]	Score [%]	Time [s]	Score [%]
1	0.17	80.61	0.58	74.49
2	0.18	73.47	0.70	74.49
3	0.28	74.49	1.00	76.53
4	0.18	78.57	0.82	76.53
5	0.17	71.43	0.68	76.53
6	0.18	73.47	0.73	76.53
7	0.17	73.47	0.68	73.47
8	0.17	75.51	0.68	76.53
9	0.18	75.51	0.69	74.49
10	0.17	74.49	0.78	74.49

TABLE 10. Computation times and classification scores for learning w_n $(n = 10^3, 10^4)$ for *wine* dataset using each variant of Algorithm 2 when $\lambda := 1.0$ was used

	1,000 iterations		10,000 iterations	
Variant	Time [s]	Score [%]	Time [s]	Score [%]
1	0.17	95.56	0.58	95.56
2	0.18	97.78	0.68	95.56
3	0.17	95.56	0.68	95.56
4	0.28	95.56	0.69	95.56
5	0.17	95.56	0.67	95.56
6	0.18	97.78	0.59	95.56
7	0.18	95.56	0.79	95.56
8	0.17	95.56	0.68	95.56
9	0.18	95.56	0.79	97.78
10	0.17	97.78	0.68	95.56

TABLE 11. Computation times and classification scores for learning w_n $(n = 10^3, 10^4)$ for *segment* dataset using each variant of Algorithm 2 when $\lambda := 1.0$ was used

	1,000 iterations		10,000 iterations	
Variant	Time [s]	Score [%]	Time [s]	Score [%]
1	0.28	78.03	1.18	77.85
2	0.29	75.09	1.39	78.20
3	0.29	77.51	1.59	77.85
4	0.28	77.68	1.49	77.51
5	0.28	77.34	1.49	77.68
6	0.29	79.24	1.52	76.99
7	0.29	77.85	1.50	76.82
8	0.28	76.12	1.38	78.37
9	0.28	77.85	1.49	78.55
10	0.28	76.82	1.60	77.51

TABLE 12. Computation times and classification scores for learning $w_n\,(n = 10^3, 10^4)$ for *breast-cancer* dataset using each variant of Algorithm 2 when $\lambda := 1.0$ was used

	1,000 iterations		10,000 iterations	
Variant	Time [s]	Score [%]	Time [s]	Score [%]
1	0.17	97.08	0.38	97.08
2	0.18	97.08	0.48	97.08
3	0.17	97.08	0.48	97.08
4	0.18	97.08	0.49	97.08
5	0.17	97.08	0.48	97.08
6	0.18	97.08	0.38	97.08
7	0.18	97.08	0.38	97.08
8	0.17	97.08	0.47	97.08
9	0.17	97.08	0.49	97.08
10	0.17	97.08	0.47	97.08

TABLE 13. Computation times and classification scores for learning $w_n\,(n = 10^3, 10^4)$ for *sonar* dataset using each variant of Algorithm 2 when $\lambda := 1.0$ was used

	1,000 iterations		10,000 iterations	
Variant	Time [s]	Score [%]	Time [s]	Score [%]
1	0.17	65.38	0.59	71.15
2	0.17	69.23	0.49	69.23
3	0.18	71.15	0.48	71.15
4	0.18	69.23	0.49	69.23
5	0.18	67.31	0.47	71.15
6	0.19	71.15	0.48	69.23
7	0.18	67.31	0.48	69.23
8	0.18	71.15	0.48	69.23
9	0.18	71.15	0.48	71.15
10	0.17	73.08	0.48	69.23

Table 14. Computation times and classification scores for learning w_n $(n = 10^3, 10^4)$ for *splice* dataset using each variant of Algorithm 2 when $\lambda := 1.0$ was used

	1,000 iterations		10,000 iterations	
Variant	Time [s]	Score [%]	Time [s]	Score [%]
1	0.18	83.40	0.60	84.74
2	0.18	84.46	0.48	84.51
3	0.17	83.08	0.48	84.55
4	0.18	83.17	0.48	84.69
5	0.17	83.77	0.48	84.46
6	0.19	82.94	0.48	84.28
7	0.18	82.11	0.52	85.10
8	0.17	81.84	0.48	84.78
9	0.17	83.03	0.48	84.64
10	0.18	82.71	0.48	84.60

Table 15. Computation times and classification scores for learning w_n $(n = 10^3, 10^4)$ for *german.numer* dataset using each variant of Algorithm 2 when $\lambda := 1.0$ was used

	1,000 iterations		10,000 iterations	
Variant	Time [s]	Score [%]	Time [s]	Score [%]
1	0.19	69.60	0.48	70.00
2	0.18	69.20	0.48	70.40
3	0.17	71.60	0.48	70.80
4	0.18	66.40	0.49	69.60
5	0.18	69.60	0.48	68.40
6	0.18	68.40	0.48	70.40
7	0.18	67.60	0.48	70.80
8	0.17	68.40	0.47	68.00
9	0.17	66.80	0.48	68.80
10	0.17	67.60	0.48	70.80

Table 16. Computation times and classification scores for learning $w_n\,(n = 10^3, 10^4)$ for *australian* dataset using each variant of Algorithm 2 when $\lambda := 1.0$ was used

	1,000 iterations		10,000 iterations	
Variant	Time [s]	Score [%]	Time [s]	Score [%]
1	0.22	87.28	0.49	88.44
2	0.18	88.44	0.48	89.02
3	0.17	88.44	0.47	88.44
4	0.18	87.86	0.49	89.02
5	0.18	87.86	0.48	88.44
6	0.18	88.44	0.48	87.86
7	0.18	87.86	0.48	88.44
8	0.17	87.86	0.47	89.02
9	0.17	88.44	0.48	88.44
10	0.17	86.71	0.48	88.44

In general, these results show that differences between any two variants did not cause a significant difference in terms of computation time or classification score. However, for some datasets (Tables 3, 10, and 13), Variant 10 generated classifiers with scores higher than those generated by the other variants when $n = 10^3$. This indicates that Variant 10 more quickly converges and finds better approximations than the other variants.

Furthermore, as shown in Table 17, the performances of the variants were similar. Table 18 shows the t-test results for Variants 2–10 compared to Variant 1 calculated

Table 17. Average time and score for each variant with $\lambda := 1.0$

	1,000 iterations		10,000 iterations	
Variant	Time [s]	Score [%]	Time [s]	Score [%]
1	0.19	79.31	0.56	79.55
2	0.20	79.25	0.63	79.48
3	0.21	79.47	0.71	79.86
4	0.21	79.17	0.66	79.58
5	0.20	78.77	0.62	79.63
6	0.20	79.32	0.63	79.56
7	0.19	78.58	0.60	79.46
8	0.19	79.04	0.59	79.52
9	0.19	79.25	0.62	79.73
10	0.20	79.26	0.64	79.49

using the data in Tables 2 to 16. The results for most variants were similar to those for Variant 1.

TABLE 18. T-test results for Variants 2–10 compared to Variant 1 with $\lambda := 1.0$

Variant	1,000 iterations	10,000 iterations
2	0.93	0.66
3	0.79	0.09
4	0.75	0.91
5	0.42	0.71
6	0.99	0.98
7	0.20	0.69
8	0.69	0.91
9	0.93	0.34
10	0.96	0.66

4.3. **Experimental results when $\lambda := 0.1$.** Next, we evaluated the performance of each variant of Algorithm 2 when a smaller value of λ ($\lambda := 0.1$) was used as the parameter of Problem 4.1. Tables 19–33 show the computation times for acquiring classifier w_n ($n = 10^3, 10^4$) and the classification scores for the test datasets. The computation method used for evaluating the classification scores was the same as that described in Subsection 4.2.

TABLE 19. Computation times and classification scores for learning w_n ($n = 10^3, 10^4$) for *a1a* dataset using each variant of Algorithm 2 when $\lambda := 0.1$ was used

	1,000 iterations		10,000 iterations	
Variant	Time [s]	Score [%]	Time [s]	Score [%]
1	0.18	70.96	0.48	73.29
2	0.18	71.08	0.49	72.64
3	0.18	72.71	0.48	72.32
4	0.18	68.79	0.48	73.05
5	0.18	70.51	0.48	71.87
6	0.29	70.12	0.48	71.74
7	0.17	70.66	0.48	72.53
8	0.17	66.88	0.47	72.81
9	0.28	69.95	0.48	72.81
10	0.18	69.89	0.48	72.69

Table 20. Computation times and classification scores for learning $w_n\,(n = 10^3, 10^4)$ for *a2a* dataset using each variant of Algorithm 2 when $\lambda := 0.1$ was used

	1,000 iterations		10,000 iterations	
Variant	Time [s]	Score [%]	Time [s]	Score [%]
1	0.18	69.94	0.48	71.97
2	0.18	69.30	0.48	72.37
3	0.18	70.73	0.48	71.89
4	0.18	71.02	0.48	71.23
5	0.17	70.72	0.47	71.65
6	0.18	68.11	0.48	71.94
7	0.17	68.02	0.48	72.13
8	0.29	68.63	0.59	72.65
9	0.18	70.39	0.48	71.34
10	0.18	69.49	0.48	72.80

Table 21. Computation times and classification scores for learning $w_n\,(n = 10^3, 10^4)$ for *a3a* dataset using each variant of Algorithm 2 when $\lambda := 0.1$ was used

	1,000 iterations		10,000 iterations	
Variant	Time [s]	Score [%]	Time [s]	Score [%]
1	0.18	70.00	0.48	71.91
2	0.18	69.72	0.48	71.92
3	0.17	71.42	0.48	71.58
4	0.17	71.09	0.48	71.76
5	0.28	68.53	0.58	71.22
6	0.28	67.98	0.48	71.67
7	0.17	65.09	0.48	71.30
8	0.28	64.98	0.48	71.47
9	0.18	66.63	0.48	71.15
10	0.17	69.93	0.59	70.80

Table 22. Computation times and classification scores for learning w_n $(n = 10^3, 10^4)$ for *diabetes* dataset using each variant of Algorithm 2 when $\lambda := 0.1$ was used

	1,000 iterations		10,000 iterations	
Variant	Time [s]	Score [%]	Time [s]	Score [%]
1	0.17	69.79	0.38	69.27
2	0.18	68.75	0.38	69.79
3	0.17	69.79	0.48	69.79
4	0.18	68.23	0.48	69.79
5	0.18	69.27	0.48	68.75
6	0.17	69.79	0.37	68.23
7	0.17	69.79	0.48	68.75
8	0.18	75.00	0.48	69.27
9	0.17	68.75	0.48	69.79
10	0.20	71.88	0.48	69.79

Table 23. Computation times and classification scores for learning w_n $(n = 10^3, 10^4)$ for *dna* dataset using each variant of Algorithm 2 when $\lambda := 0.1$ was used

	1,000 iterations		10,000 iterations	
Variant	Time [s]	Score [%]	Time [s]	Score [%]
1	0.27	81.87	0.69	87.61
2	0.27	82.97	0.68	86.76
3	0.28	83.22	0.78	87.27
4	0.27	82.88	0.67	87.35
5	0.27	83.14	0.78	88.20
6	0.27	77.82	0.68	86.93
7	0.28	80.27	0.68	87.27
8	0.29	79.85	0.88	87.18
9	0.28	79.85	0.78	87.44
10	0.29	79.34	0.81	85.75

TABLE 24. Computation times and classification scores for learning $w_n\,(n = 10^3, 10^4)$ for *iris* dataset using each variant of Algorithm 2 when $\lambda := 0.1$ was used

	1,000 iterations		10,000 iterations	
Variant	Time [s]	Score [%]	Time [s]	Score [%]
1	0.18	84.21	0.58	84.21
2	0.17	84.21	0.58	84.21
3	0.18	84.21	0.68	84.21
4	0.17	84.21	0.58	84.21
5	0.17	84.21	0.57	84.21
6	0.17	84.21	0.58	84.21
7	0.17	84.21	0.57	84.21
8	0.17	84.21	0.58	84.21
9	0.17	84.21	0.57	84.21
10	0.17	84.21	0.68	84.21

TABLE 25. Computation times and classification scores for learning $w_n\,(n = 10^3, 10^4)$ for *svmguide1* dataset using each variant of Algorithm 2 when $\lambda := 0.1$ was used

	1,000 iterations		10,000 iterations	
Variant	Time [s]	Score [%]	Time [s]	Score [%]
1	0.18	84.35	0.38	84.38
2	0.17	84.48	0.37	84.50
3	0.17	84.40	0.38	84.53
4	0.17	83.85	0.38	84.38
5	0.17	84.53	0.38	84.48
6	0.17	83.95	0.38	84.40
7	0.17	84.38	0.38	84.25
8	0.17	84.68	0.37	84.40
9	0.17	84.10	0.37	84.45
10	0.17	84.63	0.37	84.68

Table 26. Computation times and classification scores for learning $w_n\,(n = 10^3, 10^4)$ for *svmguide2* dataset using each variant of Algorithm 2 when $\lambda := 0.1$ was used

	1,000 iterations		10,000 iterations	
Variant	Time [s]	Score [%]	Time [s]	Score [%]
1	0.17	72.45	0.58	74.49
2	0.17	71.43	0.58	74.49
3	0.17	72.45	0.58	77.55
4	0.18	72.45	0.59	73.47
5	0.17	76.53	0.68	76.53
6	0.17	65.31	0.58	75.51
7	0.17	71.43	0.57	75.51
8	0.17	72.45	0.68	75.51
9	0.17	69.39	0.68	73.47
10	0.17	81.63	0.68	74.49

Table 27. Computation times and classification scores for learning $w_n\,(n = 10^3, 10^4)$ for *wine* dataset using each variant of Algorithm 2 when $\lambda := 0.1$ was used

	1,000 iterations		10,000 iterations	
Variant	Time [s]	Score [%]	Time [s]	Score [%]
1	0.17	97.78	0.59	97.78
2	0.17	97.78	0.57	97.78
3	0.17	97.78	0.68	97.78
4	0.17	97.78	0.57	97.78
5	0.17	97.78	0.58	97.78
6	0.17	97.78	0.47	97.78
7	0.17	97.78	0.57	97.78
8	0.18	97.78	0.68	97.78
9	0.17	97.78	0.58	97.78
10	0.17	97.78	0.58	97.78

Table 28. Computation times and classification scores for learning $w_n\,(n = 10^3, 10^4)$ for *segment* dataset using each variant of Algorithm 2 when $\lambda := 0.1$ was used

	1,000 iterations		10,000 iterations	
Variant	Time [s]	Score [%]	Time [s]	Score [%]
1	0.28	80.62	1.18	80.10
2	0.28	77.85	1.29	79.76
3	0.30	78.89	1.69	78.72
4	0.28	81.49	1.39	79.58
5	0.28	78.03	1.88	78.55
6	0.28	79.07	1.18	80.80
7	0.28	75.61	1.28	79.24
8	0.29	78.20	1.40	79.58
9	0.28	80.62	1.29	79.41
10	0.28	79.58	1.39	79.93

Table 29. Computation times and classification scores for learning $w_n\,(n = 10^3, 10^4)$ for *breast-cancer* dataset using each variant of Algorithm 2 when $\lambda := 0.1$ was used

	1,000 iterations		10,000 iterations	
Variant	Time [s]	Score [%]	Time [s]	Score [%]
1	0.18	97.08	0.37	97.08
2	0.17	97.08	0.48	97.08
3	0.18	97.08	0.49	97.08
4	0.18	97.08	0.38	97.08
5	0.18	97.08	0.58	97.08
6	0.18	97.66	0.38	97.08
7	0.18	97.08	0.38	97.08
8	0.17	97.08	0.37	97.08
9	0.17	97.66	0.37	97.08
10	0.18	97.08	0.37	97.08

TABLE 30. Computation times and classification scores for learning w_n ($n = 10^3, 10^4$) for *sonar* dataset using each variant of Algorithm 2 when $\lambda := 0.1$ was used

	1,000 iterations		10,000 iterations	
Variant	Time [s]	Score [%]	Time [s]	Score [%]
1	0.17	67.31	0.38	65.38
2	0.17	71.15	0.48	65.38
3	0.17	59.62	0.48	71.15
4	0.17	65.38	0.37	67.31
5	0.17	67.31	0.47	67.31
6	0.17	71.15	0.37	63.46
7	0.17	67.31	0.37	71.15
8	0.17	67.31	0.48	67.31
9	0.17	65.38	0.48	71.15
10	0.17	71.15	0.49	67.31

TABLE 31. Computation times and classification scores for learning w_n ($n = 10^3, 10^4$) for *splice* dataset using each variant of Algorithm 2 when $\lambda := 0.1$ was used

	1,000 iterations		10,000 iterations	
Variant	Time [s]	Score [%]	Time [s]	Score [%]
1	0.17	83.03	0.38	84.09
2	0.17	80.41	0.38	83.54
3	0.17	82.44	0.48	84.37
4	0.17	81.06	0.48	83.82
5	0.17	82.62	0.47	83.82
6	0.17	76.37	0.37	83.72
7	0.17	79.31	0.38	83.63
8	0.17	84.14	0.47	83.95
9	0.18	80.46	0.49	83.45
10	0.18	79.22	0.59	84.09

TABLE 32. Computation times and classification scores for learning $w_n\,(n = 10^3, 10^4)$ for *german.numer* dataset using each variant of Algorithm 2 when $\lambda := 0.1$ was used

	1,000 iterations		10,000 iterations	
Variant	Time [s]	Score [%]	Time [s]	Score [%]
1	0.17	67.60	0.38	70.80
2	0.17	67.60	0.48	72.40
3	0.19	66.80	0.47	70.40
4	0.17	69.20	0.47	70.80
5	0.17	69.60	0.48	70.80
6	0.17	65.20	0.38	69.60
7	0.17	66.40	0.48	71.20
8	0.18	70.40	0.47	69.60
9	0.17	63.60	0.48	72.00
10	0.18	65.20	0.48	70.80

TABLE 33. Computation times and classification scores for learning $w_n\,(n = 10^3, 10^4)$ for *australian* dataset using each variant of Algorithm 2 when $\lambda := 0.1$ was used

	1,000 iterations		10,000 iterations	
Variant	Time [s]	Score [%]	Time [s]	Score [%]
1	0.17	86.13	0.37	86.71
2	0.17	86.71	0.38	86.71
3	0.17	87.86	0.48	86.71
4	0.18	87.28	0.37	86.71
5	0.17	87.28	0.49	86.71
6	0.17	80.92	0.38	86.71
7	0.18	85.55	0.38	86.71
8	0.17	86.71	0.37	86.71
9	0.18	86.71	0.48	86.71
10	0.17	86.71	0.48	86.71

Table 34 shows that Variants 5 and 10 had slightly higher scores than the other variants for 1000 iterations.

Table 34. Average time and score for each variant with $\lambda := 0.1$

	1,000 iterations		10,000 iterations	
Variant	Time [s]	Score [%]	Time [s]	Score [%]
1	0.19	78.87	0.51	79.94
2	0.19	78.70	0.54	79.96
3	0.19	78.63	0.61	80.36
4	0.19	78.79	0.54	79.89
5	0.19	79.14	0.63	79.93
6	0.20	77.03	0.50	79.58
7	0.19	77.52	0.53	80.18
8	0.20	78.55	0.59	79.97
9	0.19	77.70	0.57	80.15
10	0.19	79.18	0.60	79.93

Variants 3, 7, and 9 had average scores of over 80% for 10,000 iterations. Table 35 shows the t-test results for Variants 2–10 compared to Variant 1 calculated using the data in Tables 19 to 33.

Table 35. T-test results for Variants 2–10 compared to Variant 1 with $\lambda := 0.1$

Variant	1,000 iterations	10,000 iterations
2	0.67	0.90
3	0.68	0.37
4	0.79	0.77
5	0.51	0.98
6	**0.03**	0.11
7	**0.01**	0.56
8	0.63	0.88
9	**0.01**	0.62
10	0.71	0.96

The results for most variants were similar to those for Variant 1. On the whole, the results were a little lower than when $\lambda = 1.0$ (Table 18). For Variants 6, 7, and 9, the results for 1000 iterations were lower than 0.05, indicating a significant difference between Variant 1 and Variants 6, 7, and 9.

4.4. **Experimental results when $\lambda := 10$.** Finally, we evaluated the performance of each variant of Algorithm 2 when a larger value of λ ($\lambda := 10$) was used as the parameter of Problem 4.1. Tables 36–50 show the computation times for acquiring classifier w_n ($n = 10^3, 10^4$) and the classification scores for the test datasets. The computation method used for evaluating the classification scores was the same as that described in Subsection 4.2.

TABLE 36. Computation times and classification scores for learning $w_n \, (n = 10^3, 10^4)$ for *a1a* dataset using each variant of Algorithm 2 when $\lambda := 10$ was used

	1,000 iterations		10,000 iterations	
Variant	Time [s]	Score [%]	Time [s]	Score [%]
1	0.18	71.01	0.48	70.76
2	0.17	70.17	0.48	70.76
3	0.18	70.27	0.58	70.84
4	0.27	70.37	0.48	70.87
5	0.28	70.92	0.58	70.34
6	0.27	71.29	0.48	70.77
7	0.18	70.49	0.58	70.63
8	0.28	70.88	0.58	70.58
9	0.17	70.66	0.47	70.64
10	0.17	70.83	0.58	70.87

TABLE 37. Computation times and classification scores for learning $w_n \, (n = 10^3, 10^4)$ for *a2a* dataset using each variant of Algorithm 2 when $\lambda := 10$ was used

	1,000 iterations		10,000 iterations	
Variant	Time [s]	Score [%]	Time [s]	Score [%]
1	0.17	70.08	0.58	70.63
2	0.18	70.45	0.49	70.80
3	0.18	71.74	0.58	71.11
4	0.19	70.30	0.49	70.69
5	0.18	70.87	0.57	70.95
6	0.17	70.94	0.48	70.86
7	0.17	71.14	0.48	70.63
8	0.17	70.84	0.48	70.75
9	0.17	70.73	0.48	70.61
10	0.27	70.83	0.58	70.85

TABLE 38. Computation times and classification scores for learning w_n ($n = 10^3, 10^4$) for *a3a* dataset using each variant of Algorithm 2 when $\lambda := 10$ was used

	1,000 iterations		10,000 iterations	
Variant	Time [s]	Score [%]	Time [s]	Score [%]
1	0.28	70.92	0.48	70.30
2	0.27	70.13	0.48	70.51
3	0.28	70.29	0.58	70.34
4	0.17	70.81	0.48	70.81
5	0.18	70.29	0.58	70.35
6	0.17	70.47	0.48	70.49
7	0.18	70.68	0.48	70.14
8	0.17	70.05	0.58	70.01
9	0.18	70.59	0.48	70.26
10	0.27	70.33	0.58	70.27

TABLE 39. Computation times and classification scores for learning w_n ($n = 10^3, 10^4$) for *diabetes* dataset using each variant of Algorithm 2 when $\lambda := 10$ was used

	1,000 iterations		10,000 iterations	
Variant	Time [s]	Score [%]	Time [s]	Score [%]
1	0.17	67.71	0.38	67.19
2	0.18	67.19	0.48	67.71
3	0.18	67.19	0.48	68.23
4	0.18	68.75	0.48	69.27
5	0.19	67.71	0.59	67.19
6	0.17	67.71	0.38	68.23
7	0.17	68.23	0.48	67.71
8	0.18	69.79	0.48	67.19
9	0.17	67.71	0.48	67.71
10	0.18	68.23	0.49	66.67

Table 40. Computation times and classification scores for learning $w_n\,(n = 10^3, 10^4)$ for *dna* dataset using each variant of Algorithm 2 when $\lambda := 10$ was used

	1,000 iterations		10,000 iterations	
Variant	Time [s]	Score [%]	Time [s]	Score [%]
1	0.28	82.88	0.78	84.74
2	0.27	84.99	0.78	84.99
3	0.28	84.65	0.90	84.57
4	0.27	83.31	0.79	84.91
5	0.28	83.05	0.99	84.91
6	0.27	83.56	0.68	84.57
7	0.27	85.41	1.11	84.49
8	0.27	84.23	0.88	84.49
9	0.28	84.49	0.78	85.16
10	0.27	84.32	0.88	84.57

Table 41. Computation times and classification scores for learning $w_n\,(n = 10^3, 10^4)$ for *iris* dataset using each variant of Algorithm 2 when $\lambda := 10$ was used

	1,000 iterations		10,000 iterations	
Variant	Time [s]	Score [%]	Time [s]	Score [%]
1	0.17	73.68	0.58	73.68
2	0.17	73.68	0.57	73.68
3	0.18	73.68	0.70	73.68
4	0.17	73.68	0.68	73.68
5	0.18	73.68	0.67	73.68
6	0.17	73.68	0.58	73.68
7	0.18	73.68	0.68	73.68
8	0.17	73.68	0.68	73.68
9	0.17	73.68	0.58	73.68
10	0.17	73.68	0.67	73.68

Table 42. Computation times and classification scores for learning w_n $(n = 10^3, 10^4)$ for *svmguide1* dataset using each variant of Algorithm 2 when $\lambda := 10$ was used

	1,000 iterations		10,000 iterations	
Variant	Time [s]	Score [%]	Time [s]	Score [%]
1	0.18	84.18	0.37	84.38
2	0.18	84.25	0.38	84.43
3	0.18	84.58	0.48	84.35
4	0.17	84.43	0.48	84.43
5	0.17	84.43	0.47	84.33
6	0.17	83.98	0.38	84.35
7	0.17	84.08	0.37	84.40
8	0.18	84.10	0.38	84.23
9	0.17	84.43	0.37	84.35
10	0.17	84.33	0.47	84.38

Table 43. Computation times and classification scores for learning w_n $(n = 10^3, 10^4)$ for *svmguide2* dataset using each variant of Algorithm 2 when $\lambda := 10$ was used

	1,000 iterations		10,000 iterations	
Variant	Time [s]	Score [%]	Time [s]	Score [%]
1	0.17	69.39	0.59	71.43
2	0.18	66.33	0.68	69.39
3	0.19	72.45	0.88	69.39
4	0.17	69.39	0.68	70.41
5	0.18	67.35	0.69	68.37
6	0.18	69.39	0.58	70.41
7	0.17	69.39	0.79	70.41
8	0.17	72.45	0.68	70.41
9	0.17	67.35	0.68	69.39
10	0.18	72.45	0.67	69.39

Table 44. Computation times and classification scores for learning $w_n \, (n = 10^3, 10^4)$ for *wine* dataset using each variant of Algorithm 2 when $\lambda := 10$ was used

	1,000 iterations		10,000 iterations	
Variant	Time [s]	Score [%]	Time [s]	Score [%]
1	0.18	91.11	0.58	91.11
2	0.28	91.11	0.68	88.89
3	0.18	88.89	0.67	88.89
4	0.17	91.11	0.57	88.89
5	0.17	88.89	0.67	88.89
6	0.17	88.89	0.58	91.11
7	0.18	93.33	0.69	88.89
8	0.17	91.11	0.68	91.11
9	0.17	91.11	0.67	88.89
10	0.17	88.89	0.68	91.11

Table 45. Computation times and classification scores for learning $w_n \, (n = 10^3, 10^4)$ for *segment* dataset using each variant of Algorithm 2 when $\lambda := 10$ was used

	1,000 iterations		10,000 iterations	
Variant	Time [s]	Score [%]	Time [s]	Score [%]
1	0.28	59.86	1.29	62.63
2	0.29	63.84	1.52	61.94
3	0.29	66.09	1.60	62.63
4	0.28	60.55	1.39	62.46
5	0.28	60.55	1.48	64.88
6	0.28	61.07	1.40	63.49
7	0.28	65.57	1.40	64.88
8	0.28	64.88	1.49	65.92
9	0.28	59.86	1.28	63.84
10	0.28	59.52	1.48	63.49

Table 46. Computation times and classification scores for learning $w_n\,(n = 10^3, 10^4)$ for *breast-cancer* dataset using each variant of Algorithm 2 when $\lambda := 10$ was used

	1,000 iterations		10,000 iterations	
Variant	Time [s]	Score [%]	Time [s]	Score [%]
1	0.17	97.08	0.37	97.08
2	0.17	97.08	0.48	97.08
3	0.18	97.08	0.48	97.08
4	0.17	97.08	0.48	97.08
5	0.17	97.08	0.48	97.08
6	0.17	97.08	0.48	97.08
7	0.18	97.08	0.47	97.08
8	0.17	97.08	0.47	97.08
9	0.17	97.08	0.47	97.08
10	0.17	97.08	0.47	97.08

Table 47. Computation times and classification scores for learning $w_n\,(n = 10^3, 10^4)$ for *sonar* dataset using each variant of Algorithm 2 when $\lambda := 10$ was used

	1,000 iterations		10,000 iterations	
Variant	Time [s]	Score [%]	Time [s]	Score [%]
1	0.17	75.00	0.38	73.08
2	0.17	76.92	0.48	75.00
3	0.18	75.00	0.48	76.92
4	0.17	76.92	0.48	75.00
5	0.17	73.08	0.47	75.00
6	0.17	75.00	0.38	75.00
7	0.18	73.08	0.47	76.92
8	0.17	73.08	0.48	76.92
9	0.17	75.00	0.47	75.00
10	0.18	78.85	0.48	75.00

Table 48. Computation times and classification scores for learning $w_n \, (n = 10^3, 10^4)$ for *splice* dataset using each variant of Algorithm 2 when $\lambda := 10$ was used

	1,000 iterations		10,000 iterations	
Variant	Time [s]	Score [%]	Time [s]	Score [%]
1	0.18	81.70	0.37	82.11
2	0.18	82.34	0.47	83.26
3	0.19	82.25	0.48	82.62
4	0.17	80.14	0.48	82.34
5	0.17	82.16	0.48	82.53
6	0.17	81.06	0.38	82.62
7	0.18	81.10	0.58	82.11
8	0.17	79.68	0.47	82.62
9	0.17	81.89	0.48	82.71
10	0.17	81.70	0.48	82.57

Table 49. Computation times and classification scores for learning $w_n \, (n = 10^3, 10^4)$ for *german.numer* dataset using each variant of Algorithm 2 when $\lambda := 10$ was used

	1,000 iterations		10,000 iterations	
Variant	Time [s]	Score [%]	Time [s]	Score [%]
1	0.18	65.60	0.38	70.00
2	0.17	69.20	0.48	70.00
3	0.17	69.20	0.49	70.40
4	0.17	64.00	0.48	66.80
5	0.17	68.80	0.48	69.60
6	0.17	70.80	0.48	69.60
7	0.17	70.40	0.47	71.20
8	0.17	68.80	0.48	68.80
9	0.17	69.20	0.48	70.40
10	0.17	67.60	0.48	70.00

Table 50. Computation times and classification scores for learning w_n $(n = 10^3, 10^4)$ for *australian* dataset using each variant of Algorithm 2 when $\lambda := 10$ was used

	1,000 iterations		10,000 iterations	
Variant	Time [s]	Score [%]	Time [s]	Score [%]
1	0.18	82.66	0.37	83.82
2	0.17	82.08	0.48	83.82
3	0.17	80.92	0.47	83.82
4	0.17	84.97	0.48	83.82
5	0.17	83.82	0.47	83.82
6	0.17	84.97	0.37	84.39
7	0.17	84.97	0.48	83.82
8	0.17	84.97	0.48	83.82
9	0.18	83.24	0.47	83.82
10	0.17	81.50	0.47	83.82

Table 51 shows that the average scores were lower than when $\lambda = 1.0$.

Table 51. Average time and score for each variant with $\lambda := 10$

	1,000 iterations		10,000 iterations	
Variant	Time [s]	Score [%]	Time [s]	Score [%]
1	0.20	76.19	0.53	76.86
2	0.20	76.65	0.60	76.82
3	0.20	76.95	0.66	76.99
4	0.19	76.39	0.59	76.76
5	0.20	76.18	0.65	76.79
6	0.19	76.66	0.54	77.11
7	0.19	77.24	0.64	77.13
8	0.19	77.04	0.62	77.17
9	0.19	76.47	0.58	76.90
10	0.20	76.67	0.63	76.92

Table 52 shows the T-test results for Variants 2–10 compared to Variant 1 calculated using the data in Tables 36 to 50.

Table 52. T-test results for Variants 2–10 compared to Variant 1 with $\lambda := 10$

Variant	1,000 iterations	10,000 iterations
2	0.34	0.87
3	0.20	0.72
4	0.48	0.78
5	0.97	0.84
6	0.29	0.18
7	0.07	0.46
8	0.12	0.40
9	0.38	0.88
10	0.25	0.80

The results for most variants were similar to those for Variant 1. On the whole the results were lower than when $\lambda = 1.0$ (Table 18) and $\lambda = 0.1$ (Table 35).

5. Conclusion

This paper presented a stochastic optimization algorithm plus variants for solving convex stochastic optimization problems. The proposed algorithm achieves a convergence rate of $\mathcal{O}(n^{-1/2})$. Numerical comparison of the performances of the variants with Pegasos for specific support vector machine optimization problems using LIBSVM datasets showed that their computation times for learning support vector machines were shorter and their classification scores were higher. In particular, t-test results showed that the average performances of the variants were significantly different from that of Pegasos.

Acknowledgments. We are sincerely grateful to the editor, Wataru Takahashi, and the anonymous reviewers for helping us improve the original manuscript.

References

[1] H. H. Bauschke and P. L. Combettes, *Convex Analysis and Monotone Operator Theory in Hilbert Spaces*, Springer, New York, 2011.

[2] V. S. Borkar, *Stochastic Approximation: A Dynamical Systems Viewpoint*, Cambridge University Press, Cambridge, New York, 2008.

[3] K. Fujiwara and H. Iiduka, *Modification of the Krasnosel'skiĭ-Mann fixed point algorithm by using three-term conjugate gradients*, Linear and Nonlinear Analysis **3** (2017), 189–202.

[4] H. Iiduka, *Three-term conjugate gradient method for the convex optimization problem over the fixed point set of a nonexpansive mapping*, Appl. Math. Comput. **217** (2011), 6315–6327.

[5] H. Iiduka, *Acceleration method for convex optimization over the fixed point set of a nonexpansive mapping*, Math. Program. **149** (2015), 131–165.

[6] H. Iiduka, *Optimization for Inconsistent Split Feasibility Problems*, Numer. Funct. Anal. Optim. **37** (2016), 186–205.

[7] H. Iiduka and I. Yamada, *A use of conjugate gradient direction for the convex optimization problem over the fixed point set of a nonexpansive mapping*, SIAM J. Optim. **19** (2009), 1881–1893.

[8] LIBSVM data: Classification, Regression, and Multi-label, `https://www.csie.ntu.edu.tw/~cjlin/libsvmtools/datasets/`.

[9] Y. Narushima, H. Yabe and J. A. Ford, *A three-term conjugate gradient method with sufficient descent property for unconstrained optimization*, SIAM J. Optim. **21** (2011), 212–230.

[10] A. Nemirovski, A. Juditsky, G. Lan and A. Shapiro, *Robust stochastic approximation approach to stochastic programming*, Springer, SIAM J. Optim. **19** (2009), 1574–1609.

[11] J. Nocedal and S. J. Wright, *Numerical Optimization*, 2nd edn. Springer, New York, 2006.

[12] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot and E. Duchesnay, *Scikit-learn: Machine Learning in Python*, J. Mach. Learn. Res. **12** (2011), 2825–2830.

[13] H. Robbins and S. Monro, *A Stochastic Approximation Method*, The Ann. Math. Statist. **22** (1951), 400–407.

[14] Y. Sekine and H. Iiduka, *Convergence rate analysis of projected stochastic subgradient method using conjugate gradient-like direction*, Linear and Nonlinear Analysis **3** (2017), 203–211.

[15] S. Shalev-Shwartz, Y. Singer, N. Srebro and A. Cotter, *Pegasos: primal estimated sub-gradient solver for SVM*, Math. Program. **127** (2011), 3–30.

Manuscript received 18 January 2018
revised 21 February 2018

K. Fujiwara
Department of Computer Science, Meiji University, 1-1-1 Higashimita, Tama-ku, Kawasaki-shi, Kanagawa 214-8571, Japan
E-mail address: `k5fujisan@cs.meiji.ac.jp`

K. Hishinuma
Department of Computer Science, Meiji University, 1-1-1 Higashimita, Tama-ku, Kawasaki-shi, Kanagawa 214-8571, Japan; Research Fellow of Japan Society for the Promotion of Science
E-mail address: `kaz@cs.meiji.ac.jp`

H. Iiduka
Department of Computer Science, Meiji University, 1-1-1 Higashimita, Tama-ku, Kawasaki-shi, Kanagawa 214-8571, Japan
E-mail address: `iiduka@cs.meiji.ac.jp`