Stochastic Gradient Descent with Momentum and Line Searches

David Nardi
MSc student in Artificial Intelligence, Univeristy of Florence

1st March 2024

Abstract

In recent years, tailored line search approaches have proposed to define the step-size, or learning rate, in SGD-type algorithms for finite-sum problems. In particular, a stochastic extension of standard Armijo line search has been proposed in Vaswani, Mishkin, Laradji *et al.* [1]. The development of this kind of techniques is relevant, because it shall allow to enforce a stronger converging behaviour (due to the Armijo condition), similar to that of standard GD, within SGD methods that are commonly employed with large scale training problems.

However, the stochastic line search is not immediately employable when the momentum term is part of the update equation, as the search direction might not be a descent direction (which is a necessary condition for the Armijo condition). This problem is addressed in Fan, Vaswani, Thrampoulidis *et al.* [2], where a strategy is proposed to guarantee the descent property with momentum.

Contents

1 Introduction

Different SGD-type algorithms proposed by the literature were implemented and tested on different datasets for solving the ℓ_2-regularized Logistic Regression training problem.

Those algorithms can be divided in basic SGD and SGD with line search due to common computations, follows a list of the implemented algorithms

- Mini-batch Gradient Descent with fixed step-size and momentum term, and decreasing step-size, algorithm 1 on page 8;
- Mini-batch Gradient Descent with Armijo line search and momentum term restart and correction, algorithm 2 on page 9;

This section describes the Machine Learning (ML) problem and the related optimization problem, then section 2 on page 4 summarizes the approaches proposed from the retrieved papers. Section 3 on page 11 describes the experiments performed for showing the behaviour of the algorithms on different datasets.

1.1 Classification task

Given a dataset as follows

$$\mathcal{D} = \{(x^{(i)}, y^{(i)}) \mid x^{(i)} \in \mathcal{X}, y^{(i)} \in \mathcal{Y}, i = 1, 2, \dots, N\}$$

the general machine learning optimization problem in the context of *supervised learning* is

$$\min_w f(w) = L(w) + \lambda\Omega(w) \longrightarrow \begin{cases} L(w) = \frac{1}{N}\sum_{i=1}^N \ell_i(w) \\ \Omega_{\ell_2} = \frac{1}{2}\|w\|_2^2 \end{cases}$$

where $L(w)$ is the *loss function* which is dived by the total number of samples in the dataset and $\Omega(w)$ is the *regularization term* with its coefficient λ. There are three regularization possible choices, the ℓ_2 regularization was chosen for the problem that we want to address. The vector w contains the model weights associated to the dataset features.

The task performed is the *binary classification* (so the allowed values for the response variable are $\mathcal{Y} = \{-1, 1\}$), using the Logistic Regression model. The selected loss function is the *log-loss*, for one dataset sample is

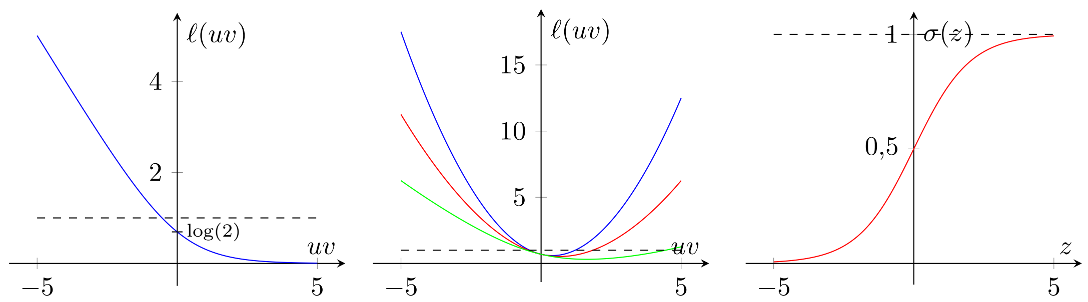
$$\ell_i(w) = \log\bigl(1 + \exp(-y^{(i)} w^T x^{(i)})\bigr) \tag{1}$$

figure 1a on the following page shows a plot of the loss function $\ell(uv) = \log\bigl(1 + \exp(-uv)\bigr)$ where $u = y^{(i)}$ and $v = w^T x^{(i)}$.

Prediction

Once the model is trained, we use the sigmoid function, see figure 1c on the next page, to classify (as positive or negative class) unseen data as follows

$$y^{(i)} = \begin{cases} 1 & \text{if } w^T x^{(i)} > 0.5 \\ -1 & \text{if } w^T x^{(i)} \leq 0.5 \end{cases}$$



(a) *Log-loss, equation (1). if $uv \gg 0$ then the example is labelled correctly; if $uv \ll 0$ then the label is the wrong one; if $uv \approx 0$ then w is the null model.*

(b) *Influence of the regularization term on the loss function, equation (3a), $\lambda = 1, 0.5, 0.1$*

(c) *Sigmoid function. Used for prediction with encoding: if $v > 0.5 \Rightarrow \hat{u} = 1$ and if $v \leq 0.5 \Rightarrow \hat{u} = -1$.*

1.2 Optimization problem

Putting together the loss function and the regularization term, we can obtain the optimization problem that we want to solve using Stochastic Gradient Descent (SGD) algorithm variants

$$\min_{w \in \mathbb{R}^{(p+1)}} f(w) = \frac{1}{N} \sum_{i=1}^{N} \log\bigl(1 + \exp(-y^{(i)} w^T x^{(i)})\bigr) + \lambda \frac{1}{2} \|w\|^2 \tag{2}$$

where $i = 1, \ldots, N$ are the dataset samples, $\mathcal{X} \subseteq \mathbb{R}^{(p+1)}$ where $p+1$ means that there are p features and the intercept. We define the matrix associated to the dataset and the model weights as follows

$$X^T = \begin{pmatrix} 1 & x_1^{(1)} & x_2^{(1)} & \ldots & x_p^{(1)} \\ 1 & x_1^{(2)} & x_2^{(2)} & \ldots & x_p^{(2)} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & x_1^{(N)} & x_2^{(N)} & \ldots & x_p^{(N)} \end{pmatrix} \in \mathbb{R}^{N \times (p+1)} \qquad x^{(i)} = \begin{pmatrix} 1 \\ x_1^{(i)} \\ x_2^{(i)} \\ \vdots \\ x_p^{(i)} \end{pmatrix} \qquad w = \begin{pmatrix} b \\ w_1 \\ w_2 \\ \vdots \\ w_p \end{pmatrix}$$

the constant column is meant for the intercept, also known as *bias*, the b weight in vector w. A compact definition for the dataset matrix is $X = (x^{(1)}, x^{(2)}, \ldots, x^{(N)})$.

The objective function $f \colon \mathbb{R}^{(p+1)} \to \mathbb{R}$ is of class $f \in C^2(\mathbb{R}^{(p+1)})$, we compute the first and second order derivatives

$$f(w) = \frac{1}{N} \sum_{i=1}^{N} \log\bigl(1 + \exp(-y^{(i)} w^T x^{(i)})\bigr) + \lambda \frac{1}{2} \|w\|^2 \tag{3a}$$

$$\nabla f(w) = \frac{1}{N} X r + \lambda w \tag{3b}$$

$$\nabla^2 f(w) = \frac{1}{N} X D X^T + \lambda I_{(p+1)} \tag{3c}$$

where $r \in \mathbb{R}^N$ is a vector of the same length as the total number of samples, whose elements are $r_i = -y^{(i)} \sigma(-y^{(i)} w^T x^{(i)})$, note that $\sigma(z)$ is the sigmoid function, $D \in \mathbb{R}^{N \times N}$ is a diagonal matrix whose elements are $d_{ii} = \sigma(y^{(i)} w^T x^{(i)}) \sigma(-y^{(i)} w^T x^{(i)})$ which implies $d_{ii} \in (0, 1)$, and $I_{(p+1)}$ is the identity matrix with size $p+1$. Dividing by N means dividing by the total number of samples involved.

The next proposition allows to solve the optimization problem.

Proposition 1. *Problem (2) admits a unique optimal solution.*

Proof. We need to prove the existence and the uniqueness of the global minimum.
(i) *Existence* of a optimal solution. The problem is quadratic and the objective function is coercive, in fact $\forall \{w^k\}$ s.t. $\lim_{k\to\infty}\|w^k\| = \infty$ holds

$$\lim_{k\to\infty} f(w^k) \geq \lim_{k\to\infty} \lambda \frac{1}{2}\|w^k\|^2 = \infty \Rightarrow \lim_{k\to\infty} f(w^k) = \infty$$

hence by a corollary of the Weirstrass theorem, the problem admits global minimum in $\mathbb{R}^{(p+1)}$.
(ii) *Unicity* of the optimal solution. We now prove that the hessian matrix (3c) is positive definite

$$w^T \nabla^2 f(w) w = w^T X D X^T w + \lambda w^T I w = \underbrace{y^T D y}_{\geq 0} + \lambda \|w\|^2 \geq \lambda \|w\|^2 > 0 \quad \forall w$$

the hessian matrix positive definite implies that the objective function is strictly convex and that implies that the global minimum, if exists, is unique. Being in the convex case, the global minimum is a $w^* \in \mathbb{R}^{(p+1)}$ s.t. $\nabla f(w^*) = 0$ for first-order optimality conditions. ∎

Remark 1. Since the log-loss is convex, the regularization term makes the objective function also *strongly convex*, this should speed up the optimization process.

2 Mini-batch gradient descent variants

In this section we tackle the algorithmic part, specifically the SGD-type is the Mini-batch Gradient Descent where the mini-batch size M is greater than 1 and much less than the dataset size, i.e. $1 < |B| = M \ll N$, however, we will call it SGD.

In order to use the algorithm, it is necessary to make further assumptions on the objective function and the gradients (how far the gradient samples are from the *true gradients*)

- the objective function in problem 2 is a loss function plus a quadratic regularization term, f is bounded below by some value f^* as we can also see in figure 1a;
- for some constant $G > 0$ the magnitude of all gradients samples is bounded $\forall w \in \mathbb{R}^{(p+1)}$, by $\|\nabla f_i(w)\| \leq G$;
- other than twice continuously differentiable, we assume that f has Lipschitz-continuous gradients with constant $L > 0$, one can also say that f is L-smooth.

The algorithm is globally convergent, so the starting point will be an arbitrary $w^0 \in \mathbb{R}^{(p+1)}$.

Stopping criterion and failures

Regarding the implementation of the algorithm, it is essential to define a stopping criterion. Given a small $\varepsilon > 0$ there are two possible choices

$$\|\nabla f(w^k)\| \leq \begin{cases} \varepsilon \\ \varepsilon\big(1 + |f(w^k)|\big) = \varepsilon\big(1 + f(w^k)\big) \end{cases}$$

unlike the first one, the second is independent from the scale of the objective function. Note that the criterion uses the full gradient.

Other than the stopping criterion, we can add conditions of premature termination like

- exceeding a threshold for the epochs number k^* or function and gradient evaluations;
- internal failures when computing w^{k+1}, for example exceeding q^* iterations during the line search (as you will se later, for the step-size α as well as the momentum term β).

Mini-batch gradient

Now we spend few words about the notation and the computation of the mini-batch gradient. Being on epoch k at iteration t, a model update starting from a w^k has the following form

$$y_{t+1} = y_t + \alpha_t d_t \tag{4}$$

the update uses information from the mini-batch B_t in the direction d_t and the step-size α_t follows a certain rule.*

The direction is an expression involving the gradient, so we want to compute the gradient w.r.t. y_t on the mini-batch B_t whose indices are randomly chosen $i_t \subset \{1, \dots, N\}$. Knowing that $\nabla f_i(w^k) = x^{(i)} r_i + \lambda w^k$

$$\begin{aligned} \nabla f_{i_t}(y_t) &= \frac{1}{M} \sum_{i \in B_t} \nabla \ell_i(y_t) + \lambda \nabla \Omega(y_t) \\ &= \frac{1}{M} \underbrace{Xr}_{i \in B_t} + \lambda y_t \end{aligned} \tag{5}$$

the expression is the same as the full gradient (3b) except that the dataset matrix contains just the mini-batch samples, so the r vector.

2.1 Stochastic gradient descent

The basic SGD version has the following update rule

$$y_{t+1} = y_t - \alpha_t \nabla f_{i_t}(y_t) \tag{6}$$

so the direction is defined as $d_t = -\nabla f_{i_t}(y_t)$ that is the *anti-gradient* evaluated on the considered mini-batch, we know that on average is a *descent direction* so the objective function doesn't decrease necessarily at each step.

Given an initial step-size $\alpha_0 \in \mathbb{R}^+$, the first two basic version are

- SGD-Fixed: constant step-size $\alpha_t = \alpha_0$;
- SGD-Decreasing: decreasing step-size $\alpha_t = \frac{\alpha_0}{k+1}$.

The first choice sees the same step-size between the epochs and so the iterations. The second choice changes the step-size at every epoch, while being constant between iterations, that particular form ensures the convergence. This two version are shown in algorithm 1 on page 8 which is a general version that includes the momentum term (see section 2.2), for this two cases we set $\beta_0 = 0$.

* *Iterations* is defined as the total number of mini-batches extracted from the dataset, while one *epoch* is when the entire dataset is passed forward. The counter for the mini-batch currently processed is t while k is for the epoch.

2.1.1 Stochastic line search

Now we move forward to the approach by Vaswani, Mishkin, Laradji *et al.* [1]. For using the algorithm proposed by the paper, one more assumption is needed, that is, the model is able to *interpolate* the data, this property requires that the gradient w.r.t. each samples converges to zero at the optimal solution

$$\text{if } w^* \mid \nabla f(w^*) = 0 \Rightarrow \nabla f_i(w^*) = 0 \;\; \forall i = 1, \dots, N$$

The proposed approach applies the Armijo line search to the SGD algorithm at every iteration, specializing the sufficient reduction condition in the context of finite-sum problems. Hence the *Armijo condition* has the following form

$$f_{i_t}\big(y_t - \alpha_t \nabla f_{i_t}(y_t)\big) \le f_{i_t}(y_t) - \gamma \alpha_t \|\nabla f_{i_t}(y_t)\|^2 \tag{7}$$

the coefficient γ is an hyper-parameter that will be set to $1/2$ for convergence properties stated by the paper.

As the standard Armijo method, the proposed line search uses a *backtracking* technique that iteratively decreases the initial step-size $\alpha_0 \in \mathbb{R}^+$ by a constant factor δ usually set to $1/2$ until the condition is satisfied.

The authors also gave heuristics in order to avoid unnecessary function evaluations by *restarting* at each iteration the step-size, to the previous multiplied by the factor $a^{M/N}/\delta$, see algorithm 3 on the next page.

The SGD with Armijo Line Search `SGD-Armijo` is shown in algorithm 2 on page 9.

2.2 Adding momentum term

The iteration performed over the mini-batches is (4) what differs from the previous versions is the direction that is

$$d_t = -\big((1-\beta_0)\nabla f_{i_t}(y_t) + \beta_0 d_{t-1}\big)$$

in a finite-sum problem the momentum term lies in a specific range $\beta_0 \in (0, 1)$ and is a constant value, the algorithm that uses this direction is the `SGDM`, the resulting iteration

$$y_{t+1} = y_t - \alpha_t\big((1-\beta)\nabla f_{i_t}(y_t) + \beta d_{t-1}\big) \tag{8}$$

which is applied as the general update rule in algorithm 1 on page 8, in this case the momentum term is constant $\beta = \beta_0$. To be clear we have the following cases

$$y_{t+1} = y_t - \alpha_t\big((1-\beta_0)\nabla f_{i_t}(y_t) + \beta_0 d_{t-1}\big) \begin{cases} \xrightarrow{\beta_0 = 0} (6)\ \texttt{SGD-Fixed},\ \texttt{SGD-Decreasing} \\ \xrightarrow[\beta_0 \in (0,1)]{} (8)\ \texttt{SGDM} \end{cases}$$

2.2.1 Stochastic line search

As the paper by Fan, Vaswani, Thrampoulidis *et al.* [2] says, when using the momentum term together with a line search, β_0 complicates the selection of a suitable step-size. The Armijo line search applied to the `SGDM` algorithm has the following condition

$$f_{i_t}(y_{t+1}) \le f_{i_t}(y_t) - \gamma \alpha_t \nabla f_{i_t}(y_t)^T \big((1-\beta_0)\nabla f_{i_t}(y_t) + \beta_0 d_{t-1}\big) \tag{9}$$

but this approach is not robust to the choice of the momentum term as the paper says.

The problem is that $\nabla f_{i_t}(y_t)^T d_t < 0$ isn't always guaranteed, i.e. the direction is not descent, therefore the line search doesn't converge. Starting from an initial $\beta_0 \in (0, 1)$, there are two situations that can be resolved as follows

$$\begin{tikzcd}
 & \nabla f(y_t)^T d_t \arrow[dl, "<0"'] \arrow[dr, "\geq 0"] & \\
\texttt{evaluate}\ (9) & & \beta = \delta\beta \arrow[ul, bend right, "\texttt{check}"']
\end{tikzcd}$$

in algorithmic terms, until the direction is descent, damp the momentum term by a factor δ, which is usually set to 0.5 like in the line search. Using this procedure, a descent direction d_t is guaranteed and it is possible to apply the algorithm 4, the procedure is called *momentum correction*, see algorithm 5 on page 10. The resulting algorithm is MSL-SGDM-C.

This procedure can be expensive, so the paper suggests another approach called *momentum restart*, when the descent direction condition for d_t isn't satisfied, the procedure restarts that direction by setting $d_{t-1} = d_0$, the paper suggests $d_0 = 0$, in general

$$\begin{tikzcd}
 & \nabla f(y_t)^T d_t \arrow[dl, "<0"'] \arrow[dr, "\geq 0"] & \\
\texttt{evaluate}\ (9) & & d_{t-1} = d_0 \arrow[d, "d_t = -\left((1-\beta_0)\nabla f_{i_t}(y_t) + \beta_0 d_0\right)"] \\
 & & \texttt{evaluate}\ (9)
\end{tikzcd}$$

so if $d_0 = 0$ the direction will be $d_t = -(1-\beta_0)\nabla f_{i_t}(y_t)$ that is a descent direction, see algorithm 6 on page 10. The resulting algorithm is MSL-SGDM-R.

The authors suggest to set the momentum term to $\beta_0 = 0.9$.

Algorithm 3: reset

Data: $a \in \mathbb{R}^+$,
opt $\in \{0, 1, 2\}$
Input: α, α_0, M, N, t
if $t = 0$ **then**
  return α_0
else if opt $= 0$ **then**
  $\alpha \leftarrow \alpha$
else if opt $= 1$ **then**
  $\alpha \leftarrow \alpha_0$
else if opt $= 2$ **then**
  $\alpha \leftarrow \alpha a^{M/N}$
end
Output: α

Algorithm 4: armijo

Data: $\gamma \in (0, 1)$, $\delta \in (0, 1)$, q^*
Input: y_t, d_t, α
$\alpha_t \leftarrow \alpha$;
$q \leftarrow 0$;
repeat
  $\alpha_t \leftarrow \delta\alpha_t$;
  $y_{t+1} \leftarrow y_t + \alpha_t d_t$;
  $q \leftarrow q + 1$;
until $f_{i_t}(y_{t+1}) \leq f_{i_t}(y_t) + \gamma\alpha_t\nabla f_{i_t}(y_{t-1})^T d_t$ **or** $q \geq q^*$;
Output: α_t

Algorithm 1: SGD-Fixed, SGD-Decreasing, SGDM

Data: $w^0 \in \mathbb{R}^{(p+1)}$, $M > 1$, k^*, $\varepsilon > 0$, $\alpha_0 \in \mathbb{R}^+$, $\beta_0 \in (0, 1)$

if *SGD-Fixed* **then**
  $\{\alpha_k\} \leftarrow \alpha_0$, $\{\beta_k\} \leftarrow 0$;
else if *SGD-Decreasing* **then**
  $\{\alpha_k\} \leftarrow \frac{\alpha_0}{k+1}$, $\{\beta_k\} \leftarrow 0$;
else if *SGDM* **then**
  $\{\alpha_k\} \leftarrow \alpha_0$, $\{\beta_k\} \leftarrow \beta_0$;
end
$k \leftarrow 0$;
while $\|\nabla f(w^k)\| > \varepsilon$ **and** $k < k^*$ **do**
  create mini-batches $B_0, \ldots, B_{N/M-1}$;
  $y_0 \leftarrow w^k$;
  $d_{-1} \leftarrow 0$;
  for $t = 0$ **to** $N/M - 1$ **do**
    get indices i_t from B_t;
    $\nabla f_{i_t}(y_t) \leftarrow \frac{1}{M} \sum_{j \in B_t} \nabla f_j(y_t)$;
    $d_t \leftarrow -\big((1 - \beta_0)\nabla f_t(y_t) + \beta_0 d_{t-1}\big)$;
    $y_{t+1} \leftarrow y_t + \alpha_k d_t$;
  end
  $w^{k+1} \leftarrow y_{N/M}$;
  $k \leftarrow k + 1$;
end

Algorithm 2: SGD-Armijo, MSL-SGDM-C, MSL-SGDM-R

Data: $w^0 \in \mathbb{R}^{(p+1)}$, $M > 1$, k^*, $\varepsilon > 0$, α_0, β_0

$k \leftarrow 0$;
while $\|\nabla f(w^k)\| > \varepsilon$ **and** $k < k^*$ **do**
  create mini-batches $B_0, \ldots, B_{N/M-1}$;
  $y_0 \leftarrow w^k$;
  $d_{-1} \leftarrow 0$;
  for $t = 0$ **to** $N/M - 1$ **do**
    get indices i_t from B_t;
    $\nabla f_{i_t}(y_t) \leftarrow \frac{1}{M} \sum_{j \in B_t} \nabla f_j(y_t)$;
    if *SGD-Armijo* **then**
      $d_t \leftarrow -\nabla f_{i_t}(y_t)$;
    else if *MSL-SGDM-C* **then**
      $d_t \leftarrow \texttt{correction}(\beta_0, \nabla f_{i_t}(y_t), d_{t-1})$ see 5;
    else if *MSL-SGDM-R* **then**
      $d_t \leftarrow \texttt{restart}(\beta_0, \nabla f_{i_t}(y_t), d_{t-1})$ see 6;
    end
    $\alpha \leftarrow \texttt{reset}(\alpha_{t-1}, \alpha_0, M, N, t)/\delta$ see 3;
    $\alpha_t \leftarrow \texttt{armijo}(y_t, d_t, \alpha)$ see 4;
    $y_{t+1} \leftarrow y_t + \alpha_t d_t$
  end
  $w^{k+1} \leftarrow y_{N/M}$;
  $k \leftarrow k + 1$;
end

Algorithm 5: correction

Data: $\delta \in (0,1)$, q^*
Input: β_0, $\nabla f_{i_t}(y_t)$, d_{t-1}

```
β ← β_0;
q ← 0;
repeat
    β ← δβ;
    d_t ← −((1 − β)∇f_{i_t}(y_t) + β d_{t−1});
    q ← q + 1;
until ∇f_{i_t}(y_t)^T d_t < 0 or q ≥ q*;
β_t ← β;
d_t ← −((1 − β_t)∇f_{i_t}(y_t) + β_t d_{t−1});
```

Output: d_t

Algorithm 6: restart

Data: d_0
Input: β_0, $\nabla f_{i_t}(y_t)$, d_{t-1}

```
q ← 0;
d_t ← −((1 − β_0)∇f_{i_t}(y_t) + β_0 d_{t−1});
if not ∇f_{i_t}(y_t)^T d_t < 0 then
    d_{t−1} ← d_0;
    d_t ← −((1 − β_0)∇f_{i_t}(y_t) + β_0 d_{t−1});
end
```

Output: d_t

3 Experiments and results discussion

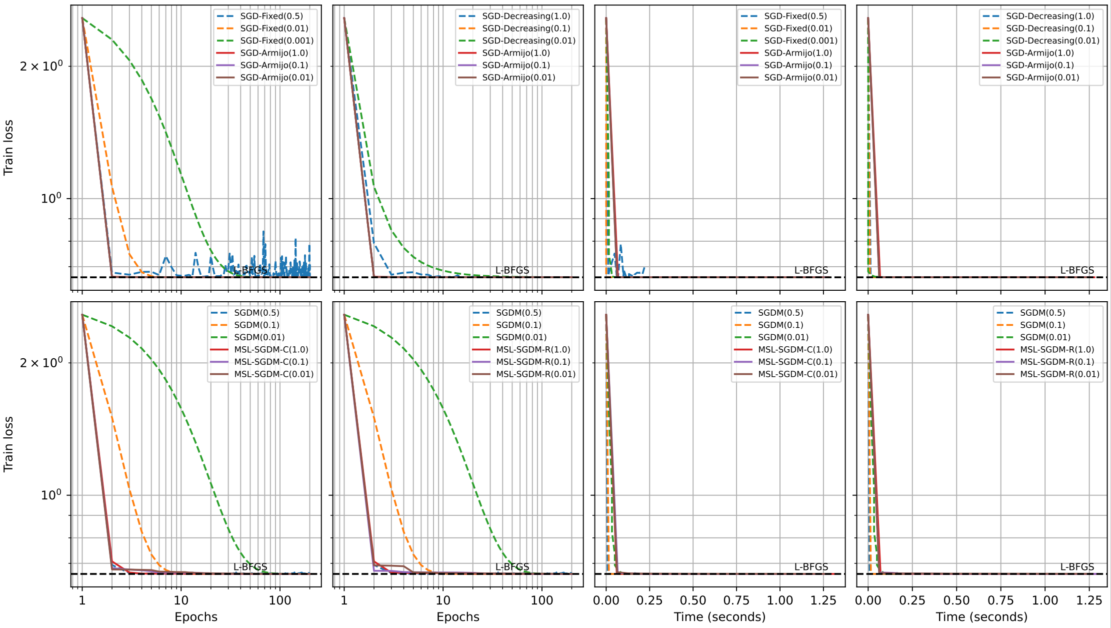
To test the efficiency the algorithms, a benchmark of six datasets retrieved from LIBSVM.

First the six algorithms are tested on a fixed number of epochs, `bib2` set the value to 200, so we do the same. We keep track of the objective function value for every epoch and the running time that every epoch took; our aim is to show how the value decreases on every epoch and the running time that takes.

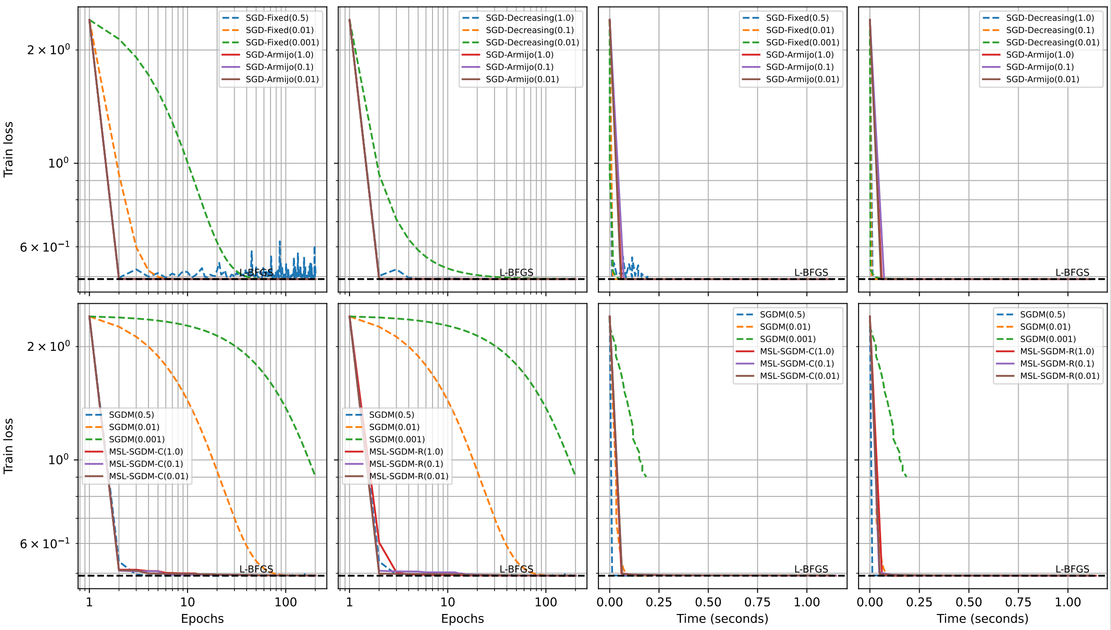
Once we have the algorithms performance at different step-size values, a fine-tuning of the hyper-parameter is done in order to obtain the best solver for every dataset based on the accuracy score. For a better comparison, the L-BFGS, Conjugate Gradient and Newton-CG algorithms are also tested.

Table 1: Diabetes dataset

Solver	α_0	Epochs	Run-time	Loss	Test score
Newton-CG	NaN	5	NaN	0.662 128	0.6429
CG	NaN	6	NaN	0.662 128	0.6429
L-BFGS	NaN	6	NaN	0.662 128	0.6429
SGD-Fixed	0.005	25	0.0000	0.662 128	0.6429
MSL-SGDM-R	0.800	186	0.0000	0.662 129	0.6429
SGDM	0.050	38	0.0000	0.662 129	0.6429
SGD-Armijo	0.500	98	0.0000	0.662 129	0.6429
SGD-Decreasing	1.000	101	0.0000	0.662 129	0.6429
MSL-SGDM-C	0.100	200	3.6822	0.662 188	0.6429

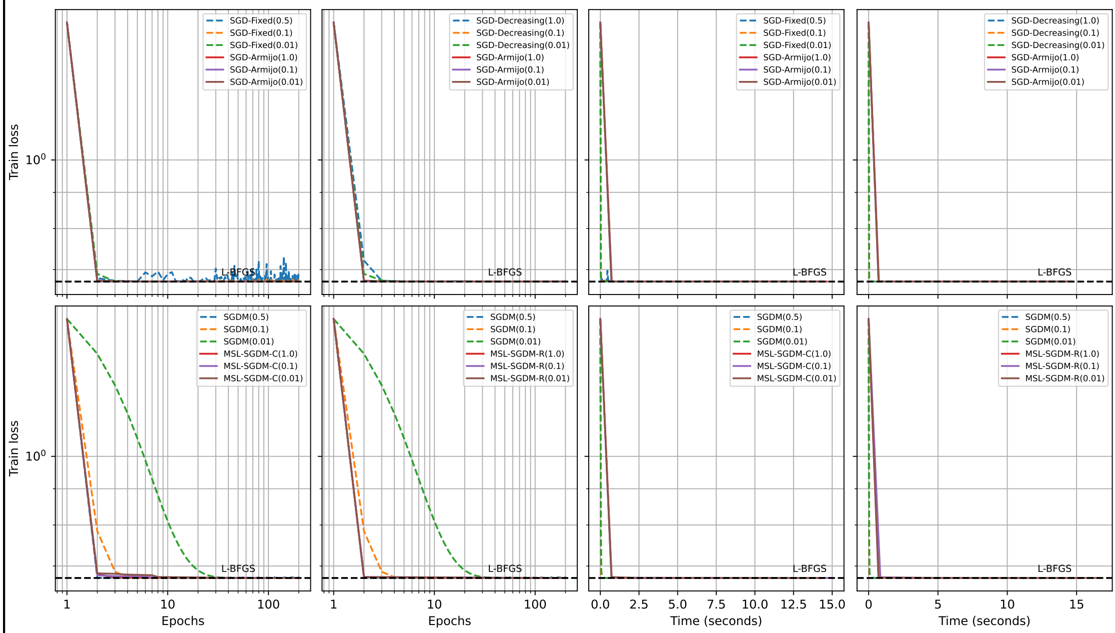


(a) *Diabetes dataset*

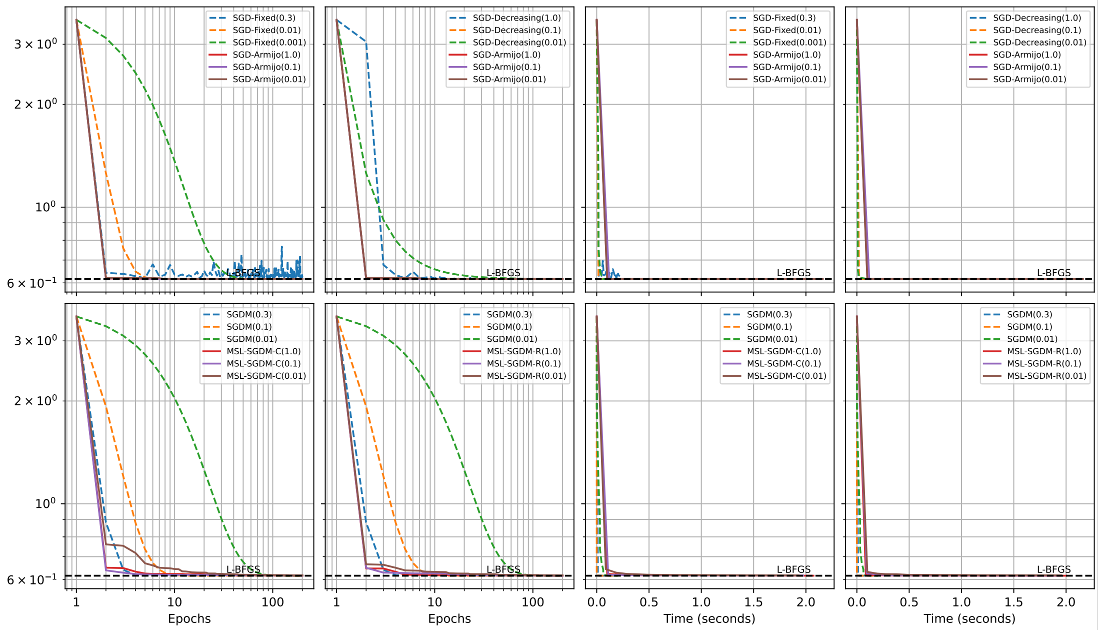


(b) *Breast cancer dataset*

Figure 1: Diabetes and Breast cancer

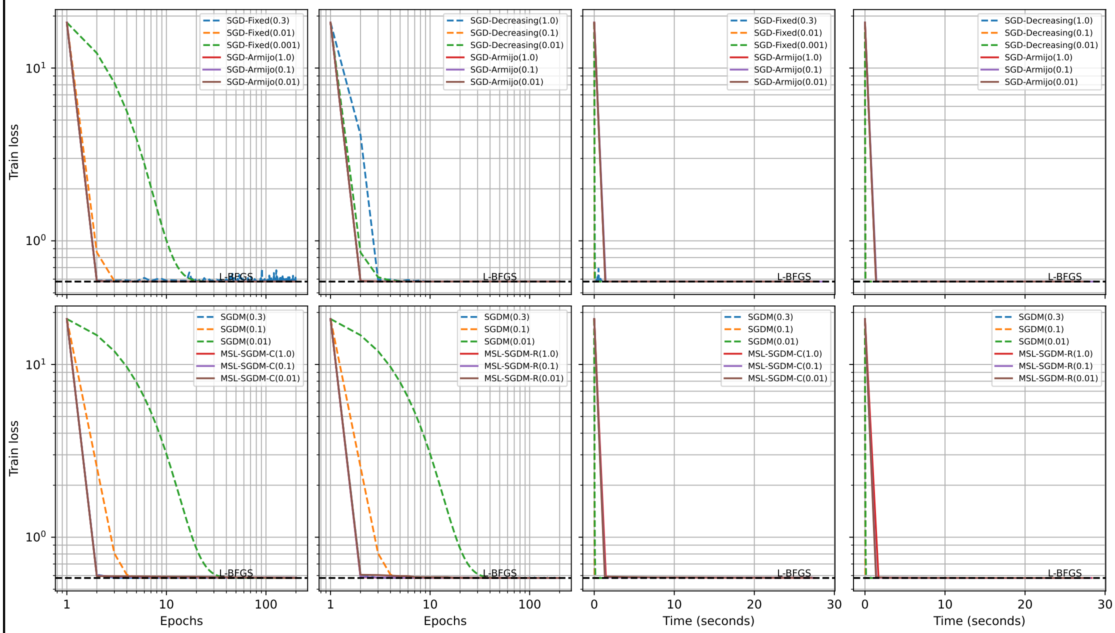


(a) *svmguide1 dataset*

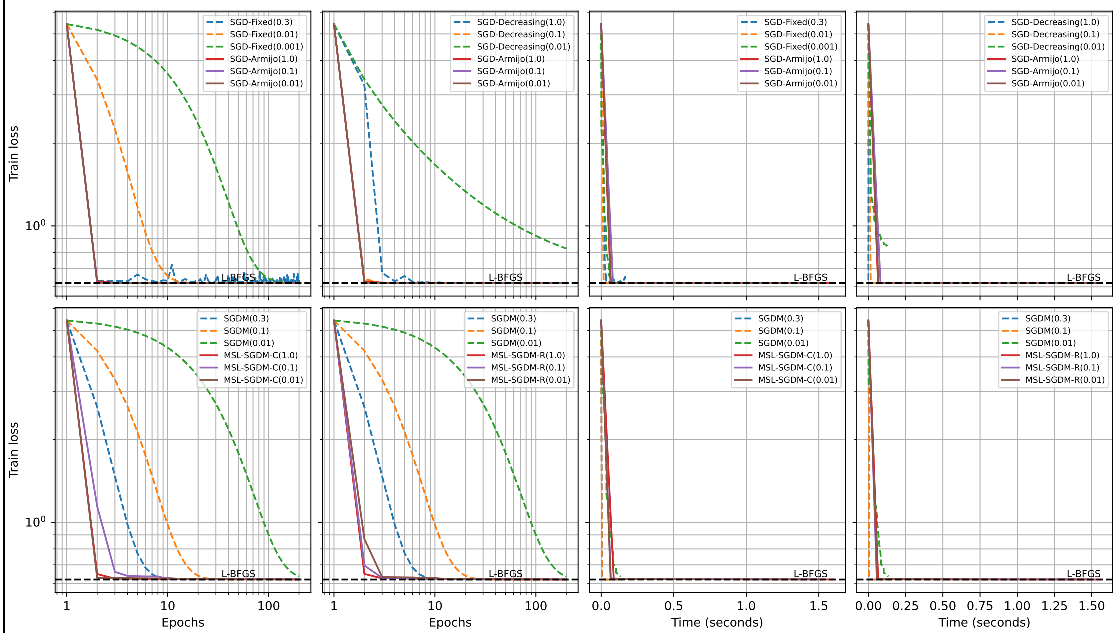


(b) *Australian dataset*

Figure 2: svmguide1 and Australian datasets



(a) *Mushrooms dataset*



(b) *German dataset*

Figure 3: Mushrooms and German datasets

Table 2: Breast cancer dataset

Solver	α_0	Epochs	Run-time	Loss	Test score
Newton-CG	NaN	7	NaN	0.492 561	0.8175
CG	NaN	8	NaN	0.492 561	0.8175
L-BFGS	NaN	7	NaN	0.492 561	0.8175
SGDM	0.050	76	0.0000	0.492 561	0.8175
SGD-Fixed	0.005	24	0.4926	0.492 561	0.8175
SGD-Decreasing	1.000	129	0.0000	0.492 561	0.8175
SGD-Armijo	1.000	81	0.0000	0.492 562	0.8175
MSL-SGDM-R	0.500	200	3.2298	0.492 572	0.8175
MSL-SGDM-C	0.500	200	3.2644	0.492 730	0.8175

Table 3: svmguide1 dataset

Solver	α_0	Epochs	Run-time	Loss	Test score
Newton-CG	NaN	5	NaN	0.673 302	0.5168
CG	NaN	8	NaN	0.673 302	0.5168
L-BFGS	NaN	5	NaN	0.673 302	0.5163
SGD-Fixed	0.010	24	0.0000	0.673 303	0.5173
SGDM	0.050	15	0.0000	0.673 303	0.5170
SGD-Decreasing	1.000	56	0.0000	0.673 303	0.5170
SGD-Armijo	0.500	21	0.0000	0.673 303	0.5155
MSL-SGDM-R	1.000	193	0.0000	0.673 304	0.5160
MSL-SGDM-C	0.100	200	14.7151	0.673 363	0.5085

Table 4: Australian dataset

Solver	α_0	Epochs	Run-time	Loss	Test score
Newton-CG	NaN	7	NaN	0.615 582	0.8768
L-BFGS	NaN	7	NaN	0.615 582	0.8768
CG	NaN	8	NaN	0.615 582	0.8768
SGD-Decreasing	0.050	18	0.0000	0.615 582	0.8768
SGDM	0.020	190	0.0000	0.615 582	0.8768
SGD-Fixed	0.001	108	0.0000	0.615 582	0.8768
SGD-Armijo	0.700	200	4.4424	0.615 585	0.8768
MSL-SGDM-R	1.000	200	4.4594	0.615 638	0.8841
MSL-SGDM-C	1.000	200	4.4995	0.615 716	0.8768

Table 5: Mushrooms dataset

Solver	α_0	Epochs	Run-time	Loss	Test score
Newton-CG	NaN	8	NaN	0.580 925	0.8862
L-BFGS	NaN	9	NaN	0.580 925	0.8862
CG	NaN	10	NaN	0.580 925	0.8862
SGD-Fixed	0.001	45	0.0000	0.580 925	0.8862
SGDM	0.010	92	2.0982	0.580 925	0.8862
SGD-Decreasing	0.100	54	0.0000	0.580 925	0.8862
SGD-Armijo	1.000	200	58.3441	0.580 938	0.8862
MSL-SGDM-R	1.000	200	58.1482	0.581 116	0.8849
MSL-SGDM-C	1.000	200	58.1920	0.581 724	0.8874

Table 6: German dataset

Solver	α_0	Epochs	Run-time	Loss	Test score
Newton-CG	NaN	7	NaN	0.619 120	0.7000
L-BFGS	NaN	10	NaN	0.619 120	0.7000
CG	NaN	9	NaN	0.619 120	0.7000
SGD-Decreasing	1.000	161	0.0000	0.619 120	0.7000
SGD-Fixed	0.005	63	0.0000	0.619 121	0.7000
SGD-Armijo	1.000	200	1.5964	0.619 132	0.7000
SGDM	0.025	200	0.1654	0.619 138	0.7000
MSL-SGDM-R	1.000	200	1.6023	0.619 192	0.7000
MSL-SGDM-C	1.000	200	1.5971	0.619 393	0.7000

4 Mathematical background

Definition 1 (Convex function)**.** Let $S \subseteq \mathbb{R}^n$ be a convex set, a function $f\colon S \to \mathbb{R}$ is said to be convex if the hessian matrix is semi-positive-defined. If the hessian matrix is positive-defined, then the function is strictly convex.

Theorem 1 (Weirstrass theorem)**.** *Let $f\colon \mathbb{R}^n \to \mathbb{R}$ be a continuous function and $S \subseteq \mathbb{R}^n$ a compact set. Then function f admits global minimum in S.*

Corollary 2 (Sufficient condition)**.** *If function $f\colon \mathbb{R}^n \to \mathbb{R}$ is a continuous and coercive function, then f admits global minimum in $\mathbb{R}^n$.*

Proposition 2 (Coercivity of a quadratic function)**.** *A quadratic function $f(x) = \frac{1}{2}x^TQx - c^Tx$ is said to be coercive if and only if the symmetric matrix $Q \in \mathbb{R}^{n\times n}$ is positive-defined.*

Proposition 3 (Unique global minimum)**.** *Let $S \subseteq \mathbb{R}^n$ be a convex set, let $f\colon S \to \mathbb{R}$ be a strictly convex function. Then the global minimum, if exists, is unique.*

Proposition 4 (First order optimality condition)**.** *$\bar{x}$ is a local minimum for $f\colon \mathbb{R}^n \to \mathbb{R}$ of class $f \in C^1(\mathbb{R}^n)$ if and only if $\nabla f(\bar{x}) = 0$.*

Proposition 5 (Second order optimality condition)**.** *$\bar{x} \in \mathbb{R}^n$ is a local minimum for $f\colon \mathbb{R}^n \to \mathbb{R}$ of class $f \in C^2(\mathbb{R}^n)$ if and only if*

$$\nabla f(\bar{x}) = 0 \quad \wedge \quad \nabla^2 f(\bar{x}) \text{ positive semi-definite}$$

References

[1] S. Vaswani, A. Mishkin, I. Laradji, M. Schmidt, G. Gidel and S. Lacoste-Julien, 'Painless stochastic gradient: Interpolation, line-search, and convergence rates,' presented at the Advances in Neural Information Processing Systems, ISSN: 1049-5258, vol. 32, 2019 (cit. on pp. 1, 6).

[2] C. Fan, S. Vaswani, C. Thrampoulidis and M. Schmidt, 'MSL: An adaptive momentem-based stochastic line-search framework,' presented at the OPT 2023: Optimization for Machine Learning, 2023 (cit. on pp. 1, 6).