# Stochastic Gradient Descent with Momentum and Line Searches

David Nardi
MSc student in Artificial Intelligence, Univeristy of Florence

7th February 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 `bib1`. 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 `bib2`, where a strategy is proposed to guarantee the descent property with momentum.

## Contents

# 1 Introduction

This report summarizes the analysis performed in order to investigate the behaviour of the algorithms retrieved from the scientific literature. The optimization problem that we aim to solve is that of the Logistic Regression with $\ell_2$-regularization term added.

The implemented algorithms are

- Mini-batch Gradient Descent with fixed and decreasing step-size, algorithm 1 on page 6;
- Mini-batch Gradient Descent with Armijo-type line search, algorithm 3 on page 7;
- Mini-batch Gradient Descent with fixed step-size and momentum, algorithm 4 on page 8;
- Mini-batch Gradient Descent with with Armijo-type line search and momentum correction and restart, algorithms 5 and 6 on page 10.

After that, the efficiency of the algorithms is tested on different datasets.

In this section the Machine Learning (ML) problem and the relative optimization problem are shown shown, proving the existence and uniqueness of the optimal solution.

## 1.1 Classification task

Given a dataset as follows

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

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

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

where $L(w)$ is called *loss function* and $\Omega(w)$ it's the *regularization term* with its coefficient $\lambda$. There are three regularization possible choices, the $\ell_2$ 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*, where $\mathcal{Y} = \{-1, 1\}$ are the allowed values for the response variable, i.e. negative and positive class; the adopted machine learning model is Logistic Regression. Every ML model has its loss function, the logistic regression uses the *log-loss*, for a sample of the dataset the loss function is as follows

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

figure 1a on page 4 shows a plot of the loss function where $u = y^{(i)}$ and $v = w^T x^{(i)}$, so the resulting function $\ell(uv) = \log\big(1 + \exp(-uv)\big)$.

## 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

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

where $i = 1, \dots, N$ are the dataset samples, $\mathcal{X} \subseteq \mathbb{R}^{(p+1)}$ where $p+1$ means that there are $p$ features and the intercept. The $1/N$ term isn't always used, we choose to use that term for scaling issues. We define the matrix associated to the dataset and the model weights as follows

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

the constant column is added for the intercept, also known as *bias*, as the $b$ weight in vector $w$.

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

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

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

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

where $r \in \mathbb{R}^N$ is a vector of the same length as the total number of sample, whose elements are $r_i = -y^{(i)} \sigma(-y^{(i)} w^T x^{(i)})$, note that $\sigma(z)$ is the sigmoid function as shown in figure 1c on the next page, $D \in \mathbb{R}^{N \times N}$ 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 $p+1$.

The next proposition allows to address the optimization problem.

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

*Proof.* (*i*) *Existence* of a optimal solution. The problem is quadratic and the objective function is coercive, in fact $\forall \{w^k\}$ such that $\lim_{k\to\infty} \|w^k\| = \infty$ holds

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

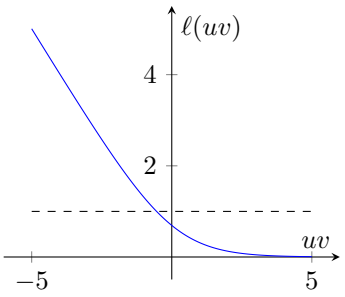
hence by a corollary of the Weirstrass theorem (see theorem 2 on page 12) the problem admits global minimum in $\mathbb{R}^{(p+1)}$.

(*ii*) *Unicity* of the optimal solution. We now prove that the hessian matrix (3c) is positive definite

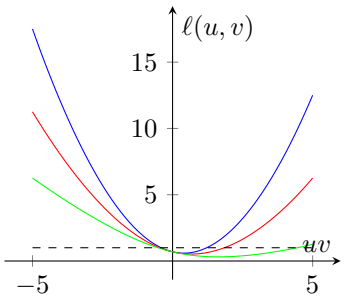
$$w^T \nabla^2 f(w) w = w^T X D X^T w + \lambda w^T I w = \underbrace{y^T D y}_{\geq 0} + \lambda \|w\|^2 \geq \lambda \|w\|^2 > 0 \quad \forall w$$

the hessian matrix positive definite implies that the objective function is strictly convex (see definition 1) and that implies that the global minimum, if exists, is unique (see proposition 3). Being in the convex case, the global minimum is a $w^* \in \mathbb{R}^{(p+1)}$ such that $\nabla f(w^*) = 0$ (see proposition 5). ■

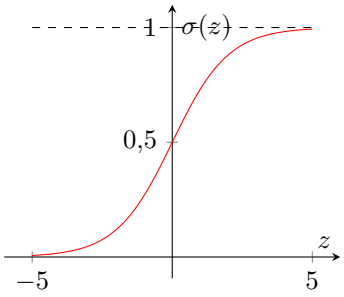
*Remark* 1. Since the log-loss is convex, the regularization term makes the objective function *strongly convex*, this should speed up the optimization process.




**(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.*




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




**(c)** *Sigmoid function*

# 2 Mini-batch gradient descent variants

In this section we tackle the algorithmic part, the SGD-type chosen is the Mini-batch Gradient Descent where the mini-batch size $M$ is greater than 1 and much less than the dataset size. For simplicity, we will call it SGD anyway.

The basic SGD perform steps of the form

$$w^{k+1} = w^k - \alpha_k \nabla f_{i_k}(w^k) \tag{4}$$

starting from an arbitrary $w^0 \in \mathbb{R}^{(p+1)}$ due to global convergence, $\nabla f_{i_k}(w^k)$ is the gradients evaluated on a random mini-batch extracted from the dataset. Using this form, without a line search method for choosing the optimal step-size $\alpha_k$, the objective function value doesn't decrease necessarily at each step, thus making the method *non-monotonous*.

In order to use the SGD 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 function $f$ in problem (2) has a *finite-sum structure*, that is the common machine learning setting;
- being a loss function plus a quadratic regularization term, $f$ is bounded below by some value $f^*$, we can also take a look at figure 1a;
- for some constant $G > 0$ the magnitude of all gradients samples are 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.

Regarding the implementation of the algorithm, it is essential to define a stopping criterion. The first choice is always

$$\|\nabla f(w^k)\| \leq \varepsilon, \quad \varepsilon > 0 \tag{5}$$

unless there is a small tolerance $\varepsilon$, the algorithm reaches a stationary point.

Other than this, 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 during the line search.

## 2.1 Basic SGD

Particularly the basic version has two possible step-size choices

- *constant step-size* $\alpha_k = \alpha$;
- *decreasing step-size* $\alpha_k = \frac{\alpha_0}{k+1}$.

the second choice has such form in order to ensure the convergence of the algorithm; this two version are shown in algorithm 1 on the next page. The iteration (4) sees the index $k$ changed to $t$, the former is the index of the *epochs* while the latter is the index of the mini-batches.

**Algorithm 1:** Mini-batch Gradient Descent with fixed or decreasing step-size

**Data:** $w^0 \in \mathbb{R}^{(p+1)}$, $M > 1$, $k^*$, $\varepsilon > 0$, $\{\alpha_k\}$
$k \leftarrow 0$;
**while** $\|\nabla f(w^k)\| \leq \varepsilon \wedge k < k^*$ **do**
  shuffle $\{1, \ldots, N\}$ and split $B_1, \ldots, B_{N/M}$ s.t. $1 < |B_t| = M \ll N$;
  $y_0 \leftarrow w^k$;
  **for** $t = 1$ **to** $N/M$ **do**
    get indices $i_t$ from $B_t$;
    $\nabla f_{i_t}(w) \leftarrow \frac{1}{M} \sum_{j \in B_t} \nabla f_j(y_{t-1})$;
    $d_t \leftarrow -\nabla f_{i_t}(y_{t-1})$;
    $y_t \leftarrow y_{t-1} + \alpha_k d_t$;
  **end**
  $w^{k+1} \leftarrow y_{N/M}$;
  $k \leftarrow k + 1$;
**end**

## 2.2 Stochastic line search

Now we move on to the approach by `bib1`. The proposed algorithm needs one more assumption, 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

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

The proposed approach applies the Armijo line search to the SGD algorithm, referring to the notation in (4), the *Armijo condition* is as follows

$$f_{i_k}(w^{k+1}) \leq f_{i_k}(w^k) + \gamma \alpha_k \nabla f_{i_k}(w^k)^T d_{i_k} = f_{i_k}(w^k) - \gamma \alpha_k \nabla \|f_{i_k}(w^k)\|^2$$

where the direction $d_{i_k}$ is equal to the *anti-gradient* evaluated on the considered sample. The constant $\gamma$ is an hyper-parameter set to $1/2$ for convergence properties in the strongly-convex case.

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

The authors also gave heuristics in order to avoid unnecessary function evaluations, see algorithm 2 on the following page,

See algorithm 3 on the next page.

**Algorithm 2:** Procedure for resetting the step-size in the line search setting

**Data:** $\alpha$, $\alpha_0$, $a$, $M$, $N$, $t$, $\texttt{opt}$
**if** $t = 1$ **then**
  **return** $\alpha_0$
**else if** $\texttt{opt} = 0$ **then**
  $\alpha \leftarrow \alpha$
**else if** $\texttt{opt} = 1$ **then**
  $\alpha \leftarrow \alpha_0$
**else if** $\texttt{opt} = 2$ **then**
  $\alpha \leftarrow \alpha a^{M/N}$
**end**
**return** $\alpha$

**Algorithm 3:** Mini-batch Gradient Descent with Armijo line search

**Data:** $w^0 \in \mathbb{R}^{(p+1)}$, $M > 1$, $k^*$, $\varepsilon > 0$, $\alpha_0 \in \mathbb{R}^+$
**Data:** $\gamma \in (0,1)$, $\delta \in (0,1)$, $a \in \mathbb{R}^+$, $\texttt{opt} \in \{0,1,2\}$
$k \leftarrow 0$;
**while** $\|\nabla f(w^k)\| \leq \varepsilon \wedge k < k^*$ **do**
  shuffle $\{1,\ldots,N\}$ and split $B_1,\ldots,B_{N/M}$ s.t. $1 < |B_t| = M \ll N$;
  $y_0 \leftarrow w^k$;
  **for** $t = 1$ **to** $N/M$ **do**
    get indices $i_t$ from $B_t$;
    $\nabla f_{i_t}(w) \leftarrow \frac{1}{M}\sum_{j\in B_t} \nabla f_j(y_{t-1})$;
    $d_t \leftarrow -\nabla f_{i_t}(y_{t-1})$;
    $\alpha \leftarrow \texttt{reset}(\alpha_{t-1}, \alpha_0, a, M, N, t, \texttt{opt})$;
    $q \leftarrow 0$;
    **while** $f_{i_t}(y_t) > f_{i_t}(y_{t-1}) + \gamma\alpha\nabla f_{i_t}(y_{t-1})^T d_t \wedge q < q^*$ **do**
      $\alpha \leftarrow \delta\alpha$;
      $y_t \leftarrow y_{t-1} + \alpha d_t$;
      $q \leftarrow q + 1$;
    **end**
    $\alpha_t \leftarrow \alpha$;
    $y_t \leftarrow y_{t-1} + \alpha_t d_t$
  **end**
  $w^{k+1} \leftarrow y_{N/M}$;
  $k \leftarrow k + 1$;
**end**

### 2.3 Adding momentum term

**Algorithm 4:** Mini-batch Gradient Descent with fixed Momentum term and step-size

**Data:** $w^0 \in \mathbb{R}^{(p+1)}$, $M > 1$, $k^*$, $\varepsilon > 0$, $\{\alpha_k\}$, $\{\beta_k\}$
$k \leftarrow 0$;
**while** $\|\nabla f(w^k)\| \leq \varepsilon \wedge k < k^*$ **do**
  shuffle $\{1, \ldots, N\}$ and split $B_1, \ldots, B_{N/M}$ s.t. $1 < |B_t| = M \ll N$;
  $y_0 \leftarrow w^k$;
  **for** $t = 1$ **to** $N/M$ **do**
    get indices $i_t$ from $B_t$;
    $\nabla f_{i_t}(w) \leftarrow \frac{1}{M} \sum_{j \in B_t} \nabla f_j(y_{t-1})$;
    $d_t \leftarrow -\big((1-\beta)\nabla f_{i_t}(y_{t-1}) + \beta d_{t-1}\big)$;
    $y_t \leftarrow y_{t-1} + \alpha_k d_t$;
  **end**
  $w^{k+1} \leftarrow y_{N/M}$;
  $k \leftarrow k + 1$;
**end**

**Algorithm 5:** Mini-batch Gradient Descent with Armijo line search and Momentum correction

**Data:** $w^0 \in \mathbb{R}^{(p+1)}$, $M > 1$, $k^*$, $\varepsilon > 0$, $\alpha_0 \in \mathbb{R}^+$, $\beta_0 \in (0,1)$
**Data:** $\gamma \in (0,1)$, $\delta_a \in (0,1)$, $\delta_b \in (0,1)$, $a \in \mathbb{R}^+$, $\texttt{opt} \in \{0,1,2\}$
$k \leftarrow 0$;
**while** $\|\nabla f(w^k)\| \leq \varepsilon \wedge k < k^*$ **do**
  shuffle $\{1,\dots,N\}$ and split $B_1,\dots,B_{N/M}$ s.t. $1 < |B_t| = M \ll N$;
  $y_0 \leftarrow w^k$;
  **for** $t = 1$ **to** $N/M$ **do**
    get indices $i_t$ from $B_t$;
    $\nabla f_{i_t}(w) \leftarrow \frac{1}{M}\sum_{j\in B_t} \nabla f_j(y_{t-1})$;
    $\beta \leftarrow \beta_0$;
    $q_m \leftarrow 0$;
    **while** $\nabla f_{i_t}(y_{t-1})^T d_t \geq 0 \wedge q_m < q_m^*$ **do**
      $\beta \leftarrow \delta_m \beta$;
      $d_t \leftarrow -\big((1-\beta)\nabla f_{i_t}(y_{t-1}) + \beta d_{t-1}\big)$;
      $q_m \leftarrow q_m + 1$;
    **end**
    $\beta_t \leftarrow \beta$;
    $d_t \leftarrow -\big((1-\beta_t)\nabla f_{i_t}(y_{t-1}) + \beta_t d_{t-1}\big)$;
    $\alpha \leftarrow \texttt{reset}(\alpha_{t-1}, \alpha_0, a, M, N, t, \texttt{opt})$;
    $q_a \leftarrow 0$;
    **while** $f_{i_t}(y_t) > f_{i_t}(y_{t-1}) + \gamma\alpha\nabla f_{i_t}(y_{t-1})^T d_t \wedge q_a < q_a^*$ **do**
      $\alpha \leftarrow \delta_m \alpha$;
      $y_t \leftarrow y_{t-1} + \alpha d_t$;
      $q_a \leftarrow q_a + 1$;
    **end**
    $\alpha_t \leftarrow \alpha$;
    $y_t \leftarrow y_{t-1} + \alpha_t d_t$
  **end**
  $w^{k+1} \leftarrow y_{N/M}$;
  $k \leftarrow k + 1$;
**end**

**Algorithm 6:** Mini-batch Gradient Descent with Armijo line search and Momentum restart

**Data:** $w^0 \in \mathbb{R}^{(p+1)}$, $M > 1$, $k^*$, $\varepsilon > 0$, $\alpha_0 \in \mathbb{R}^+$, $\beta \in (0,1)$
**Data:** $\gamma \in (0,1)$, $\delta_a \in (0,1)$, $a \in \mathbb{R}^+$, $\texttt{opt} \in \{0,1,2\}$
$k \leftarrow 0$;
$d_0 \leftarrow 0$;
**while** $\|\nabla f(w^k)\| \leq \varepsilon \wedge k < k^*$ **do**
  shuffle $\{1,\dots,N\}$ and split $B_1,\dots,B_{N/M}$ s.t. $1 < |B_t| = M \ll N$;
  $y_0 \leftarrow w^k$;
  **for** $t = 1$ **to** $N/M$ **do**
    get indices $i_t$ from $B_t$;
    $\nabla f_{i_t}(w) \leftarrow \frac{1}{M}\sum_{j\in B_t} \nabla f_j(y_{t-1})$;
    $d_t \leftarrow -\big((1-\beta)\nabla f_{i_t}(y_{t-1}) + \beta d_{t-1}\big)$;
    **if** $\nabla f_{i_t}(y_{t-1})^T d_t \geq 0$ **then**
      $d_t \leftarrow d_0$;
    **end**
    $\alpha \leftarrow \texttt{reset}(\alpha_{t-1}, \alpha_0, a, M, N, t, \texttt{opt})$;
    $q_a \leftarrow 0$;
    **while** $f_{i_t}(y_t) > f_{i_t}(y_{t-1}) + \gamma\alpha\nabla f_{i_t}(y_{t-1})^T d_t \wedge q_a < q_a^*$ **do**
      $\alpha \leftarrow \delta_m \alpha$;
      $y_t \leftarrow y_{t-1} + \alpha d_t$;
      $q_a \leftarrow q_a + 1$;
    **end**
    $\alpha_t \leftarrow \alpha$;
    $y_t \leftarrow y_{t-1} + \alpha_t d_t$
  **end**
  $w^{k+1} \leftarrow y_{N/M}$;
  $k \leftarrow k + 1$;
**end**

# 3 Experiments

# 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 $f(x) = \frac{1}{2}x^T Qx - c^T x$ 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*

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