Stochastic Gradient Descent with Momentum and Line Searches

David Nardi
MSc student in Artificial Intelligence, Univeristy of Florence

11th 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 ℓ_2-regularization term added.

Follows a list of the suggested algorithms from the literature, since those methods share many computations, we decided to implement just two algorithms

- Mini-batch Gradient Descent with fixed step-size and momentum, and decreasing step-size, algorithm 1 on page 7;
- Mini-batch Gradient Descent with Armijo line search and momentum restart or correction, algorithm 2 on page 8;

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 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*, 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

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

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 $\ell(uv) = \log\bigl(1 + \exp(-uv)\bigr)$.

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

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

where $i = 1, \dots, N$ are the dataset samples, $\mathcal{X} \subseteq \mathbb{R}^{(p+1)}$ where $p+1$ means that there are p features and the intercept. 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. A compact definition for the dataset matrix is $X = (x^{(1)}, x^{(2)}, \dots, x^{(N)})$.

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

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

$$\nabla f(w) = Xr + \lambda w \tag{3b}$$

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

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. 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 $\forall \{w^k\}$ such that $\lim_{k\to\infty} \|w^k\| = \infty$ holds

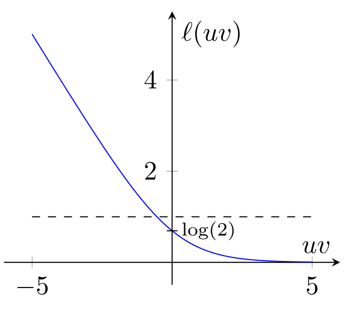
$$\lim_{k\to\infty} f(w^k) \geq \lim_{k\to\infty} \lambda \frac{1}{2} \|w^k\|^2 = \infty \Rightarrow \lim_{k\to\infty} f(w^k) = \infty$$

hence by a corollary of the Weirstrass theorem (see theorem 2 on page 10) 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

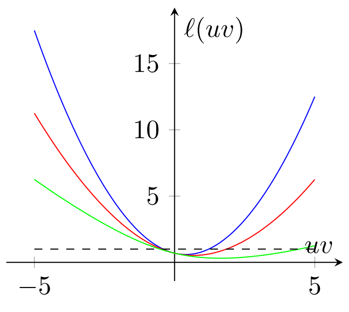
$$w^T \nabla^2 f(w) w = w^T XDX^T w + \lambda w^T I w = \underbrace{y^T D y}_{\geq 0} + \lambda \|w\|^2 \geq \lambda \|w\|^2 > 0 \quad \forall w$$

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). ■

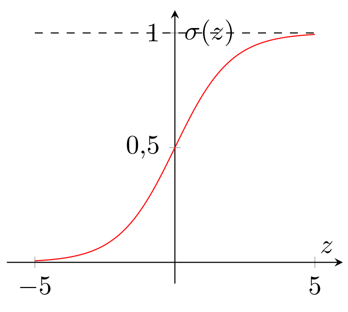
Remark 1. Since the log-loss is convex, the regularization term makes the objective function *strongly convex*, this should speed up the optimization process.



(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;* 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 $v \leq 0.5 \Rightarrow \hat{u} = -1$.

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, i.e. $1 < |B_t| = M \ll N$. For simplicity, we will call it SGD anyway.

The basic SGD performs steps of the form

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

where the direction d_k is equal to the *anti-gradient* evaluated on the considered sample (a random mini-batch extracted from the dataset), knowing that $\mathbb{E}\big[\nabla f_{i_k}(w^k)\big] = \nabla f(w)$, d_k is a *descent direction.* Also due to the global convergence, the algorithm can start from an arbitrary $w^0 \in \mathbb{R}^{(p+1)}$.

Using this step-size form, without a line search method for choosing the optimal step-size α_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.

Stopping criterion and failures

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 ε, the algorithm reaches a stationary point. Another choice could be

$$\|\nabla f(w^k)\| \leq \varepsilon\big(1 + |\underbrace{f(w^k)}_{\geq 0\,\forall w^k}|\big) = \varepsilon\big(1 + f(w^k)\big) \tag{6}$$

unlike the previous criterion (5), this one in independent from the scale of the objective function.

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 exceeding q^* iterations during the line search.

Mini-batch gradient

Now we spend a few words about the computation of the mini-batch gradient evaluated on a point using the pseudo-code notation. Given a mini-batch B_t at iteration t with indices i_t and the previous weights y_t, we want to compute

$$\begin{aligned}\nabla f_{i_t}(y_t) &= \frac{1}{M}\sum_{j\in B_t}\nabla f_j(y_t) = \frac{1}{M}\sum_{j\in B_t}\big(x^{(j)}r_j + \lambda y\big) = \frac{1}{M}\bigg(\sum_{j\in B_t}x^{(j)}r_j + M\lambda y\bigg)\\ &= \frac{1}{M}\big(\underbrace{Xr}_{i_t\in B_t} + \lambda' y\big)\end{aligned}$$

so we use the same expression as the full gradient except that the dataset matrix contains just the mini-batch samples (and so the r vector), and the regularization coefficient is redefined as $\lambda' = M\lambda$ where M is the size of the considered mini-batch.

2.1 Stochastic gradient descent

The SGD-type variants differs by the selection of the step-size. Particularly the basic version has two possible choices

- *constant step-size* $\alpha_k = \alpha_0$;
- *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 page 7. The iteration (4) sees index k changed to t, the former is the index of the *epochs* while the latter is the index of the mini-batches.

2.1.1 Stochastic line search

Now we move on to the approach by `bib1`. Other than $\mathcal{L}_0 = \{w \in \mathbb{R}^{(p+1)} \mid f(w) \leq f(w^0)\}$ being a *compact set*, since the function is coercive (see...), 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

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

The proposed approach applies the Armijo line search to the SGD algorithm, specializing the condition of sufficient reduction in the context of finite-sum problems. Referring to the notation in (4), the *Armijo condition* has the following form

$$f(w^{k+1}) \leq f(w^k) + \gamma\alpha_k \nabla f(w^k)^T d_k \Rightarrow f_{i_k}(w^{k+1}) \leq f_{i_k}(w^k) + \gamma\alpha_k \nabla f_{i_k}(w^k)^T d_k$$

so, when it comes to SGD we obtain

$$f_{i_k}\big(w^k - \alpha_k \nabla f_{i_k}(w^k)\big) \leq f_{i_k}(w^k) - \gamma\alpha_k \nabla \|f_{i_k}(w^k)\|^2 \tag{7}$$

as already said, d_k is a *descent direction.* The constant γ 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 α_0 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, see algorithm 3 on page 8.[*]

The SGD with Stochastic Line Search is shown in algorithm 2 on page 8.

2.2 Adding momentum term

The iteration performed is still $w^{k+1} = w^k + \alpha_k d_k$ what differs from the basic versions is the direction

$$d_k = -\big((1-\beta)\nabla f_{i_k}(w^k + \beta d_{k-1})\big)$$

in a finite-sum problem the momentum term must be selected from a specific range $\beta \in (0, 1)$, the algorithm is called SGDM, the resulting iteration

$$w^{k+1} = w^k - \alpha_k\big((1-\beta)\nabla f_{i_k}(w^k + \beta d_{k-1})\big) \tag{8}$$

which is applied in algorithm 1 on the following page.

As the paper `bib2` says, when using the momentum term together with a line search, β complicates the selection of a suitable step-size; using the algorithm 2, the approach is not robust to the choice of the momentum term.

Like the stochastic line search approach in section ..., the Armijo condition added to the SGDM algorithm has the form

$$f_{i_k}(w^k + \alpha_k d_k) \leq f_{i_k}(w^k) - \gamma\alpha_k \nabla f_{i_k}(w^k)^T \big((1-\beta)\nabla f_{i_k}(w^k) + \beta d_{k-1}\big) \tag{9}$$

placing $d_{i_k} = -\nabla f_{i_k}(w^k)$ we obtain the equation (7).

The problem is that $\nabla f_{i_k}(w^k)^T d_{i_k} < 0$ isn't always guaranteed, i.e. the direction is not descent, therefore the line search doesn't converge. There are thus two situations that can be resolved as follows

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

[*] *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.

in algorithmic terms, while the direction is not descent, damp the momentum term by a factor δ_m usually set to 1/2. Using this procedure, a descent direction d_{i_k} is guaranteed, and is called *momentum correction*, see algorithm 2 on the next page.

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

$$\begin{tikzcd}
& \nabla f(x^k)^T d_k \arrow[dl, "< 0"'] \arrow[dr, "\geq 0"] & \\
\texttt{evaluate}\ (9) & & d_{k-1} = d_0 \arrow[d, "d_k = -\big((1-\beta)\nabla f_{i_k}(w^k) + \beta d_0\big)"] \\
& & \texttt{evaluate}\ (9)
\end{tikzcd}$$

so if $d_0 = 0$ the direction will be $d_k = -(1-\beta)\nabla f_{i_k}(w^k)$, see algorithm 2 on the following page.

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 $B_0, \dots, B_{N/M-1}$;
    $y_0 \leftarrow w^k$;
    $d_{-1} \leftarrow 0$;
    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: $w^0 \in \mathbb{R}^{(p+1)}$, $M > 1$, $k^*$, $\varepsilon > 0$, $\alpha_0$, $\beta_0$,
$\gamma \in (0,1)$, $\delta \in (0,1)$, $a \in \mathbb{R}^+$, opt $\in \{0,1,2\}$
$k \leftarrow 0$;
while $\|\nabla f(w^k)\| > \varepsilon$ and $k < k^*$ do
    create mini-batches $B_0, \dots, B_{N/M-1}$;
    $y_0 \leftarrow w^k$;
    $d_{-1} \leftarrow 0$;
    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
            $d_t \leftarrow$ correction$(\beta_0, \nabla f_{i_t}(y_t), d_{t-1})$ see 5;
        else if MSL-SGDM-R then
            $d_t \leftarrow$ restart$(\beta_0, \nabla f_{i_t}(y_t), d_{t-1})$ see 4;
        end
        $\alpha \leftarrow$ reset$(\alpha_{t-1}, \alpha_0, a, M, N, t, \text{opt})/\beta$ see 3;
        $q \leftarrow 0$;
        repeat
            $\alpha \leftarrow \delta\alpha$;
            $y_{t+1} \leftarrow y_t + \alpha d_t$;
            $q \leftarrow q + 1$;
        until $f_{i_t}(y_{t+1}) \leq f_{i_t}(y_t) + \gamma\alpha\nabla f_{i_t}(y_{t-1})^T d_t$ or $q \geq q^*$;
        $\alpha_t \leftarrow \alpha$;
        $y_{t+1} \leftarrow y_t + \alpha_t d_t$
    end
    $w^{k+1} \leftarrow y_{N/M}$;
    $k \leftarrow k + 1$;
end
```

Algorithm 3: `reset`

```
Input: $\alpha$, $\alpha_0$, $a$, $M$, $N$, $t$, opt
if $t = 0$ then
    return $\alpha_0$
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: $\alpha$
```

Algorithm 4: `restart`

```
Input: $\beta_0$, $\nabla f_{i_t}(y_t)$, $d_{t-1}$
q ← 0;
$d_t \leftarrow -\big((1-\beta)\nabla f_{i_t}(y_t) + \beta d_{t-1}\big)$;
if $\nabla f_{i_t}(y_{t-1})^T d_t \geq 0$ then
    $d_{t-1} \leftarrow d_0$;
    $d_t \leftarrow -(1-\beta)\nabla f_{i_t}(y_t)$;
end
Output: $d_t$
```

Algorithm 5: `correction`

```
Input: $\beta_0$, $\nabla f_{i_t}(y_t)$, $d_{t-1}$
$\beta \leftarrow \beta_0$;
$q \leftarrow 0$;
repeat
    $\beta \leftarrow \delta\beta$;
    $d_t \leftarrow -\big((1-\beta)\nabla f_{i_t}(y_t) + \beta d_{t-1}\big)$;
    $q \leftarrow q + 1$;
until $\nabla f_{i_t}(y_t)^T d_t < 0$ or $q \geq q^*$;
$\beta_t \leftarrow \beta$;
$d_t \leftarrow -\big((1-\beta_t)\nabla f_{i_t}(y_t) + \beta_t d_{t-1}\big)$;
Output: $d_t$
```

3 Experiments and results discussion

$\beta = 0.9$

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*

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