CGD: Modifying the Loss Landscape by Gradient Regularization

Shikhar Saxena, Tejas Bodas, and Arti Yardi

International Institute of Information Technology (IIIT) Hyderabad, India
{shikhar.saxena@research.,tejas.bodas@,arti.yardi@}iiit.ac.in

Abstract. Line-search methods are commonly used to solve optimization problems. The simplest line search method is steepest descent where one always moves in the direction of the negative gradient. Newton's method on the other hand is a second-order method that uses the curvature information in the Hessian to pick the descent direction. In this work, we propose a new line-search method called Constrained Gradient Descent (CGD) that implicitly changes the landscape of the objective function for efficient optimization. CGD is formulated as a solution to the constrained version of the original problem where the constraint is on a function of the gradient. We optimize the corresponding Lagrangian function thereby favourably changing the landscape of the objective function. This results in a line search procedure where the Lagrangian penalty acts as a control over the descent direction and can therefore be used to iterate over points that have smaller gradient values, compared to iterates of vanilla steepest descent. We establish global linear convergence rates for CGD and provide numerical experiments on synthetic test functions to illustrate the performance of CGD. We also provide two practical variants of CGD, CGD-FD which is a Hessian free variant and CGD-QN, a quasi-Newton variant and demonstrate their effectiveness.

Keywords: Numerical Optimization · Gradient Regularization.

1 Introduction

Line-search methods such as gradient descent and Newton's method have been extremely popular in solving unconstrained optimization problems [15]. However, vanilla version of such algorithms typically produce unsatisfactory results when applied to large scale optimization problems such as those involving deep neural networks. In such problems, the objective/loss function has a very non-linear landscape that is embedded with several local maxima and minima making them difficult to optimize [11]. To tackle this problem, one of the approaches used is to modify the objective/loss function in such a way that the new function is easier to optimize. This is typically done by adding a regularization term that will make the loss landscape more smooth, and which would in turn reduce the chances of the optimizer reaching sub-optimal minima, thereby improving generalizability [5,18,1,17,8].

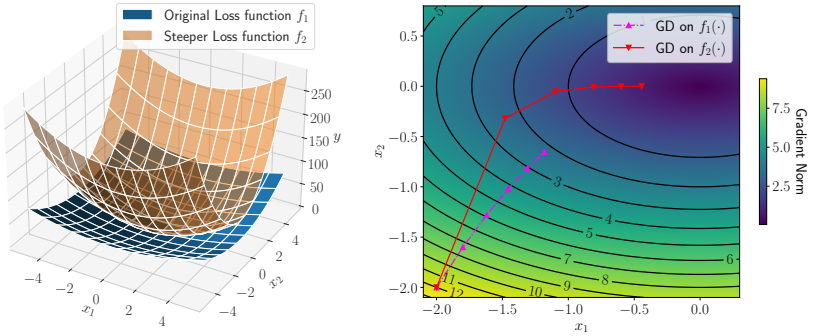


Fig. 1. *(Left)* Loss function $f_1(\mathbf{x}) = x_1^2 + 2x_2^2$ and a steeper loss function $f_2(\mathbf{x})$. *(Right)* 5 steps of GD on functions $f_1(\mathbf{x})$ and $f_2(\mathbf{x})$ with a fixed step-size $\alpha = 0.05$. The contours and the gradient norm heatmap are over the function $f_1(\mathbf{x})$.

In this work, we focus on this idea of modifying the objective function for effective numerical optimization and propose an algorithmic paradigm to achieve it. To illustrate the key idea, consider minimizing a loss function $f_1(\cdot)$ over $\mathbb{R}^2$ as shown in Fig. 1 *(Left)*. Additionally consider a steeper loss function $f_2(\cdot)$ such that for every $\mathbf{x} \in \mathbb{R}^2$ we have $f_1(\mathbf{x}) \leq f_2(\mathbf{x})$. Note that the functions are such that their minima coincides. Now perform gradient descent (GD) with a fixed step-size on both these functions. Their respective trajectories are plotted in Fig. 1 *(Right)* on the contours of f_1. It is clear from the figure that the iterates of GD on f_2 are much closer to the minima as compared to iterates of GD on f_1. We make this observation a focal point of this work and investigate if one can achieve the latter iterates (iterates from GD on f_2) on the former (f_1).

Towards this, we propose Constraint gradient descent (CGD), a variant of the GD algorithm which achieves this by constraining the gradient norm to be appropriately small. Instead of solving this constrained optimization problem, we consider the Lagrangian of this function and perform GD on it. The Lagrangian parameter λ controls the penalty on gradient norm and thereby controls the steepness of the modified function. It is on this modified function that we seek to apply GD to possibly attain iterates that are much closer to the local minima as compared to GD iterates. From Fig. 1 *(Right)* we additionally see that compared to GD, the CGD iterates (which is nothing but GD iterates on f_2) have a lower gradient norm. Since the path to minima is over points with lower gradient norm, this we believe is an attractive feature and even amounts to better generalization properties for high dimensional functions such as loss landscapes of neural networks. The idea of flat minima and their attractiveness goes long back to Hochreiter and Schmidhuber [7]: An optimal minimum is considered "flat" if the test error changes less in its neighbourhood. Keskar et al. [10] and Chaudhari et al. [4] observe better generalization results for neural networks at flat minima. Gradient regularization has only been recently explored, that too from a numer-

ical perspective for deep neural networks (DNNs) where it has been shown that fixed learning rates result in implicit gradient regularization [17,8,1,16].

A key aim of this work is to understand gradient regularization from the perspective of a numerical optimization algorithm and identify properties and features that may not have been obvious earlier. We believe CGD and its variants discussed in this work have the ability to find flatter minima, something which is useful while training neural networks. We summarize our contributions below:

- We propose a new line-search procedure called Constraint Gradient Descent (CGD) that performs gradient descent on a gradient regularized loss function. We also provide global linear convergence for CGD in Theorem 1.
- While CGD requires the Hessian information, we also propose a first-order variant of CGD using finite-difference approximation of the Hessian called *CGD-FD*. We define appropriate stopping criteria in CGD-FD for settings where gradient computation can be expensive.
- We conduct experiments over synthetic test functions to compare the performance of CGD and its variants compared to standard line-search procedures.
- Our work also provides new insights to existing gradient regularization based methods. In fact, we re-interpret and identify pitfalls in the Explicit Gradient Regularization (EGR) Method [1] using our formulation.

The remainder of the paper is organized as follows. In the next section, we recall some preliminaries on line-search methods. We then discuss the CGD algorithm and propose its variants. We then illustrate the performance of our algorithm on several test functions and conclude with a discussion on future directions.

2 Notation and Preliminaries

The set of real numbers and non-negative real numbers is denoted by $\mathbb{R}$ and $\mathbb{R}^+$ respectively. We consider a vector $\mathbf{x} \in \mathbb{R}^n$ as a column vector given by $\mathbf{x} = \begin{bmatrix} x_1 & x_2 & \dots & x_n \end{bmatrix}^T$. We use a boldface letter to denote a vector and lowercase letters (with subscripts) to denote its components. L^p-norm of a vector $\mathbf{x}$ is defined as $\|\mathbf{x}\|_p = \left(\sum_i |x_i|^p\right)^{1/p}$. Setting $p = 2$ gives us L^2-norm: $\|\mathbf{x}\| \triangleq \|\mathbf{x}\|_2 = \sqrt{\sum_i x_i^2} = \sqrt{\mathbf{x}^T\mathbf{x}}$. $\lambda_{\max}(\cdot)$ and $\lambda_{\min}(\cdot)$ denote the maximum and minimum eigenvalue of a matrix respectively. Norm of a matrix is assumed to be the spectral norm *i.e.*, $\|A\| = \sqrt{\lambda_{\max}(A^TA)}$. I_n denotes the identity matrix of size $n \times n$. All zero vector of length n is denoted by $\mathbf{0}_n$.

2.1 Line-Search Methods

Let $f(\mathbf{x})$ be a twice differential function with domain $\mathcal{D} \subseteq \mathbb{R}^n$ and codomain $\mathbb{R}$, i.e., $f : \mathcal{D} \to \mathbb{R}$. Let $\nabla f(\mathbf{x}_k)$ and $H(\mathbf{x}_k)$ denote the gradient and Hessian of the function $f(\mathbf{x})$ evaluated at point $\mathbf{x}_k \in \mathcal{D}$. For the sake of simplicity, we will also use the notation ∇f_k and H_k to denote $\nabla f(\mathbf{x}_k)$ and $H(\mathbf{x}_k)$ respectively. For the

given function $f(\mathbf{x})$, we consider the problem of finding its minimizer, i.e., we wish to find $\mathbf{x}^* \in \mathcal{D}$ such that

$$\mathbf{x}^* = \arg\min_{\mathbf{x}\in\mathcal{D}} f(\mathbf{x}). \tag{1}$$

We focus on the situation where $\mathbf{x}^*$ is obtained using an iterative line-search procedure (for details refer [12, Ch. 3]). The steepest descent (*gradient descent*) method, Newton's method, and quasi-Newton's method are some examples of line-search methods. In an iterative algorithm, the key idea is to begin with an initial guess $\mathbf{x}_0$ for the minimizer and generate a sequence of vectors $\mathbf{x}_0, \mathbf{x}_1, \ldots$ until convergence (or until desired level of accuracy is achieved). In such algorithms, $\mathbf{x}_{k+1}$ is obtained using $\mathbf{x}_k$ using a pre-defined update rule.

For line-search methods, the update rule can be written in a general form as, $\mathbf{x}_{k+1} = \mathbf{x}_k + \alpha \mathbf{p}_k$ where $\alpha \in \mathbb{R}^+$ is the step-size (or learning rate) and $\mathbf{p}_k \in \mathbb{R}^n$ corresponds to the direction in the k^{th} iteration. For $\mathbf{p}_k$ to be a *descent* direction, the following condition must hold:

$$\nabla f_k^T \mathbf{p}_k < 0. \tag{2}$$

2.2 Quasi-Newton Methods

Quasi-Newton methods (QN methods) are line-search methods that maintain an approximation of the inverse of the Hessian to emulate Newton's direction. The iterates here are of the form,

$$\mathbf{x}_{k+1} = \mathbf{x}_k - \alpha G_k \nabla f_k \tag{3}$$

where G_k is a positive definite matrix that is updated at every step to approximate the inverse Hessian.

We consider DFP and BFGS algorithms that update G_k using symmetric rank-two updates at each descent step [3,6]. The update equations for DFP and BFGS are given as follows (for more details refer [12, Ch. 6]):

$$\textbf{(DFP)}\ G_{k+1} = G_k + \frac{\mathbf{s}_k \mathbf{s}_k^T}{\mathbf{y}_k^T \mathbf{s}_k} - \frac{G_k \mathbf{y}_k \mathbf{y}_k^T G_k}{\mathbf{y}_k^T G_k \mathbf{y}_k}$$

$$\textbf{(BFGS)}\ G_{k+1} = \left(I_n - \frac{\mathbf{s}_k \mathbf{y}_k^T}{\mathbf{y}_k^T \mathbf{s}_k}\right) G_k \left(I_n - \frac{\mathbf{y}_k \mathbf{s}_k^T}{\mathbf{y}_k^T \mathbf{s}_k}\right) + \frac{\mathbf{s}_k \mathbf{s}_k^T}{\mathbf{y}_k^T \mathbf{s}_k}$$

where $\mathbf{s}_k = \mathbf{x}_{k+1} - \mathbf{x}_k$ and $\mathbf{y}_k = \nabla f_{k+1} - \nabla f_k$.

We can derive the corresponding Hessian approximation $\tilde{G}_k$ from the update steps for G_k by applying the Sherman-Morrison-Woodbury formula [12, App. A]. These are given as follows:

$$\textbf{(DFP)}\ \tilde{G}_{k+1} = \left(I_n - \frac{\mathbf{y}_k \mathbf{s}_k^T}{\mathbf{y}_k^T \mathbf{s}_k}\right) \tilde{G}_k \left(I_n - \frac{\mathbf{s}_k \mathbf{y}_k^T}{\mathbf{y}_k^T \mathbf{s}_k}\right) + \frac{\mathbf{y}_k \mathbf{y}_k^T}{\mathbf{y}_k^T \mathbf{s}_k} \tag{4}$$

$$\textbf{(BFGS)}\ \tilde{G}_{k+1} = \tilde{G}_k + \frac{\mathbf{y}_k \mathbf{y}_k^T}{\mathbf{y}_k^T \mathbf{s}_k} - \frac{\tilde{G}_k \mathbf{s}_k \mathbf{s}_k^T \tilde{G}_k}{\mathbf{s}_k^T \tilde{G}_k \mathbf{s}_k} \tag{5}$$

3 Constrained Gradient descent

In this section we propose *Constrained Gradient Descent* (CGD), an iterative line-search method to find a minimizer $\mathbf{x}^*$ of the given function $f(\mathbf{x})$ (see Equation 1). The key idea in our approach is to focus on the set of $\mathbf{x} \in \mathcal{D}$ such that $\nabla f(\mathbf{x})$ is close to zero. For this consider a general constrained optimization problem:

$$\mathbf{x}^* = \underset{\substack{\mathbf{x}\in\mathcal{D} \\ h(\nabla f(\mathbf{x}))\leq\epsilon}}{\arg\min} \; f(\mathbf{x}) \tag{6}$$

where the constraint $h(\cdot)$ is defined on the gradient and ϵ is a small positive real number. The unconstrained optimization problem corresponding to Equation 6 is obtained by penalizing the constraint with a Lagrange multiplier $\lambda > 0$:

$$\mathbf{x}^\star = \underset{\mathbf{x}\in\mathcal{D}}{\arg\min} \Big[f(\mathbf{x}) + \lambda \; h\left(\nabla f(\mathbf{x})\right) \Big] \tag{7}$$

Note that ϵ doesn't affect the optimization and hence ignored from the objective. Also, observe that such a formulation provides us with a *modified* loss function to optimize over. A suitable constraint $h(\cdot)$ will make the objective steeper while also keeping the stationary points intact. We call this *penalization* or using a *gradient penalty* since the objective is penalized at points based on its gradient value. For CGD, we consider $h(\cdot)$ to be the square of L^2-norm of the gradient.

$$\mathbf{x}^\star = \underset{\mathbf{x}\in\mathcal{D}}{\arg\min}\, g(\mathbf{x}) \triangleq f(\mathbf{x}) + \lambda \|\nabla f(\mathbf{x})\|^2 = f(\mathbf{x}) + \lambda \left(\nabla f(\mathbf{x})^T \nabla f(\mathbf{x})\right) \tag{8}$$

Steepest descent iterates over $g(\cdot)$ in Equation 8 are given as

$$\begin{aligned} \mathbf{x}_{k+1} &= \mathbf{x}_k - \alpha \nabla g_k \\ &= \mathbf{x}_k - \alpha \left(\nabla f_k + 2\lambda H_k \nabla f_k\right) \\ &= \mathbf{x}_k - \alpha B_k \nabla f_k \end{aligned} \tag{9}$$

where $B_k \triangleq I_n + 2\lambda H_k$. Thus, $\mathbf{p}_k = -B_k \nabla f_k$ which is the direction taken at the k^{th} iteration by CGD. Revisiting the example in Fig. 1, the function $f_1(\mathbf{x}) = x_1^2 + 2x_2^2$ was penalized with the square of L^2-norm of gradient as given in Equation 8. Thus, the modified loss function $f_2(\mathbf{x}) = x_1^2 + 2x_2^2 + \lambda\left((2x_1)^2 + (4x_2)^2\right) = (1 + 4\lambda)x_1^2 + 2(1 + 8\lambda)x_2^2$ where λ was chosen to be 0.4.

We now provide a lemma that investigates if the penalized objective function has stationary points which are different from the original function and if so characterizes them.

Lemma 1. *Let $\mathcal{S}_{\hat{\mathbf{x}}}$ and $\mathcal{S}_{\mathbf{x}^\star}$ be the stationary points of $f(\mathbf{x})$ and $g(\mathbf{x})$ as defined in Equation 8. Then, $\mathcal{S}_{\hat{\mathbf{x}}} \subseteq \mathcal{S}_{\mathbf{x}^*}$. Furthermore, for any $\mathbf{x}^* \in \mathcal{S}_{\mathbf{x}^*}$ one of the following is true: (a) $\mathbf{x}^* \in \mathcal{S}_{\hat{\mathbf{x}}}$ or (b) $\nabla f(\mathbf{x}^*)$ is an eigenvector of $H(\mathbf{x}^*)$ with the eigenvalue $-\frac{1}{2\lambda}$.*

Proof. From Equation 9, for any $\hat{\mathbf{x}} \in \mathcal{S}_{\hat{\mathbf{x}}}$, $\nabla f(\hat{\mathbf{x}}) = \mathbf{0}_n \Rightarrow \nabla g(\hat{\mathbf{x}}) = \mathbf{0}_n$. So, $\mathcal{S}_{\hat{\mathbf{x}}} \subseteq \mathcal{S}_{\mathbf{x}^*}$ holds trivially. Now for any $\mathbf{x}^* \in \mathcal{S}_{\mathbf{x}^*}$, $\nabla g(\mathbf{x}^*) = \mathbf{0}_n \Rightarrow (I_n + 2\lambda H(\mathbf{x}^*))\nabla f(\mathbf{x}^*) = \mathbf{0}_n$. If $\nabla f(\mathbf{x}^*) \neq \mathbf{0}_n$, we have $H(\mathbf{x}^*)\nabla f(\mathbf{x}^*) = -\frac{1}{2\lambda}\nabla f(\mathbf{x}^*)$ which proves the result. □

The above lemma illustrates that additional stationary points could possibly be introduced and some of these points could also be local minima. The lemma also characterizes conditions under which this is true and therefore such points can easily be detected. A perturbation from the current λ in that case results in the iterate to descend further, possibly moving towards a better minimum.

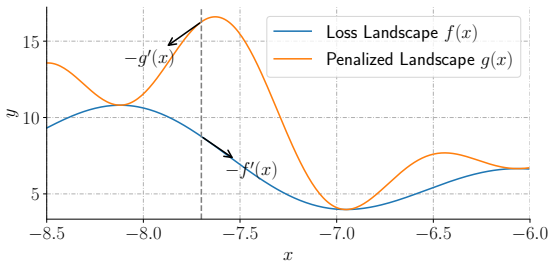


Fig. 2. Penalization introducing new stationary points at $x \approx -6.5$ and $x \approx -7.6$, and changing the local maximum at $x \approx -8.1$ to a local minimum.

Further note that within the current scheme, the nature of stationary points of $f(\mathbf{x})$ might change in $g(\mathbf{x})$. Particularly, the local maxima and saddle points of $f(\mathbf{x})$ might become local minima in $g(\mathbf{x})$ depending on λ. This happens because penalization does not change the function value at the stationary point while spiking up the function values at points in the neighbourhood.

As a result, descent on the penalized function might actually cause ascent over the original loss function. For example in Fig. 2, observe that at $x = -7.7$ (marked with the dashed line), steepest descent along $-\nabla g(\mathbf{x})$ direction would actually cause an ascent over the original loss function.

To fix this behaviour, we ensure that we only move along the direction $-\nabla g(\mathbf{x})$ when it is a descent direction. Therefore, we only move along $-\nabla g(\mathbf{x})$, if $\nabla f(\mathbf{x})^T \nabla g(\mathbf{x}) > 0$ (from Equation 2). Otherwise, we set $\lambda = 0$ at this point and move along the $-\nabla f(\mathbf{x})$ direction. Note that this simple check also helps us to avoid stopping at artificially introduced stationary points. We summarize CGD in Algorithm 1.

3.1 Global Linear Convergence of CGD

We begin with the following definition followed by a theorem on convergence guarantees for CGD.

Algorithm 1 Constrained Gradient Descent (CGD)

Input: Objective function $f : \mathcal{D} \to \mathbb{R}$, initial point $\mathbf{x}_0$, max iterations T, step-size α, regularization coefficient λ.
Output: Final point $\mathbf{x}_T$.

for iteration $k = 0, \ldots, T-1$ **do**
  $\mathbf{p}_k \leftarrow -\left(I_n + 2\lambda_k H_k\right) \nabla f_k$
  if $\nabla f_k^T \mathbf{p}_k < 0$ **then** ▷ *Check if* $\mathbf{p}_k$ *is a Descent Direction*
    $\mathbf{x}_{k+1} \leftarrow \mathbf{x}_k + \alpha \mathbf{p}_k$
  else
    $\mathbf{x}_{k+1} \leftarrow \mathbf{x}_k - \alpha \nabla f_k$
return $\mathbf{x}_T$

Definition 1 (PL Inequality [14]). *A function $f : \mathcal{D} \to \mathbb{R}$ satisfies the PL inequality if for some $\mu > 0$,*

$$\frac{1}{2}\|\nabla f(x)\|^2 \geq \mu(f(x) - f^*), \ \forall x \in \mathcal{D}. \tag{10}$$

where f^* is the optimal function value.

Theorem 1. *Let f be a convex, L-smooth function on $\mathcal{D} \subseteq \mathbb{R}^n$. Let $\{\mathbf{x}_k\}_{k\geq 0}$ be the sequence generated by the CGD method as described in Equation 9. Then for a constant step-size $\alpha \in \left(0, \frac{2}{L(1+2\lambda L)^2}\right)$, the following holds true [2, Theorem 4.25]:*

1. *The sequence $\{f_k\}_{k\geq 0}$ is nonincreasing. In addition, for any $k \geq 0$, $f_{k+1} < f_k$ unless $\nabla f_k = \mathbf{0}$.*
2. *$\nabla f_k \to 0$ as $k \to \infty$.*

Furthermore, if f satisfies the PL inequality (Equation 10) then the CGD method with a step-size of $\frac{1}{L(1+2\lambda L)^2}$, has a global linear convergence rate,

$$f_k - f^* \leq \left(1 - \frac{\mu}{L(1+2\lambda L)^2}\right)^k (f_0 - f^*).$$

Proof. Since f is convex and L-smooth we have,

- $\mathbf{0} \preceq H \preceq LI \Leftrightarrow 0 \leq \mathbf{v}^T H \mathbf{v} \leq L\|\mathbf{v}\|^2 \Leftrightarrow 0 \leq \|H\| \leq L$
- $f(\mathbf{y}) \leq f(\mathbf{x}) + \nabla f(\mathbf{x})^T(\mathbf{y} - \mathbf{x}) + \frac{L}{2}\|\mathbf{y} - \mathbf{x}\|^2$

Substituting $\mathbf{x}_{k+1}$ and $\mathbf{x}_k$ in place of $\mathbf{y}$ and $\mathbf{x}$. And $\mathbf{x}_{k+1} - \mathbf{x}_k = -\alpha \nabla g_k$,

$$f_{k+1} \leq f_k - \alpha \nabla f_k^T \nabla g_k + \frac{L}{2}\alpha^2 \|\nabla g_k\|^2 \tag{11}$$

Bounds on $\|\nabla g_k\|^2$ and $\nabla f_k^T \nabla g_k$ are found as follows:

$$\begin{aligned}
\|\nabla g_k\|^2 &= \|\nabla f_k + 2\lambda H_k \nabla f_k\|^2 \\
&= \|\nabla f_k\|^2 + 4\lambda \nabla f_k^T H_k \nabla f_k + 4\lambda^2 \|H_k \nabla f_k\|^2 \\
&\leq \|\nabla f_k\|^2 + 4\lambda \nabla f_k^T H_k \nabla f_k + 4\lambda^2 \|H_k\|^2 \|\nabla f_k\|^2 \\
&\leq \|\nabla f_k\|^2 \left(1 + 4\lambda L + 4\lambda^2 L^2\right) = (1 + 2\lambda L)^2 \|\nabla f_k\|^2
\end{aligned}$$

and $\nabla f_k^T \nabla g_k = \|\nabla f_k\|^2 + 2\lambda \nabla f_k^T H_k \nabla f_k \geq \|\nabla f_k\|^2$. Substituting in Equation 11,

$$\begin{aligned} f_{k+1} - f_k &\leq -\alpha \|\nabla f_k\|^2 + \frac{L}{2}\alpha^2 (1+2\lambda L)^2 \|\nabla f_k\|^2 \\ &= -\alpha \left(1 - \frac{L(1+2\lambda L)^2 \alpha}{2}\right) \|\nabla f_k\|^2 \end{aligned} \tag{12}$$

Thus, we have, $f_k - f_{k+1} \geq M \|\nabla f_k\|^2 \geq 0$ where $M = \alpha \left(1 - \frac{L(1+2\lambda L)^2 \alpha}{2}\right) > 0$ for constant step-size $\alpha \in \left(0, \frac{2}{L(1+2\lambda L)^2}\right)$. So, the equality $f_{k+1} = f_k$ only holds when $\nabla f_k = \mathbf{0}$. Furthermore, since the sequence $\{f_k\}_{k \geq 0}$ is non-increasing and bounded below, it converges. So, $\nabla f_k \to 0$ as $k \to \infty$. This proves the first two parts. For the third part, consider the μ-PL assumption; from Equations 10 and 12 at step-size $\alpha = \frac{1}{L(1+2\lambda L)^2}$, we have

$$f_{k+1} - f_k \leq -\frac{1}{2L(1+2\lambda L)^2} \|\nabla f_k\|^2 \leq -\frac{\mu}{L(1+2\lambda L)^2}(f_k - f^*)$$

On subtracting f^* from both sides,

$$\therefore f_{k+1} - f^* \leq \left(1 - \frac{\mu}{L(1+2\lambda L)^2}\right)(f_k - f^*)$$

Applying this inequality recursively gives us the result. □

Note that the PL inequality assumption is not necessary for functions that are strongly convex or strictly convex functions (over compact sets) since the inequality is already satisfied. Karimi et al. [9] further shows that this convergence result can be extended to functions of the form $h(A\mathbf{x})$ where h is a strongly (or strictly) convex function composed with a linear function *e.g.*, least-squares problem and logistic regression.

3.2 CGD-FD: Finite difference approximation of the Hessian

Note that CGD is a second-order optimization algorithm as it requires the Hessian information to compute each iterate. We improve over this complexity by restricting CGD to be a first-order line search method wherein the Hessian is approximated using a finite difference [13]. Using Taylor series, we know that

$$\nabla f(\mathbf{x}_k + \Delta\mathbf{x}) = \nabla f_k + H_k \Delta\mathbf{x} + \mathcal{O}(\|\Delta\mathbf{x}\|^2).$$

Now let $\Delta\mathbf{x} = r\mathbf{v}$ where r is arbitrarily small. Then, we can rewrite the above expression as:

$$H\mathbf{v} = \frac{\nabla f(\mathbf{x}_k + r\mathbf{v}) - \nabla f_k}{r} + \mathcal{O}(\|r\|).$$

Substituting $\mathbf{v} = \nabla f_k$ and using this approximation in Equation 9 gives us

$$\begin{aligned} \nabla g(\mathbf{x}_k) &\approx \nabla f_k + 2\lambda \frac{\nabla f(\mathbf{x}_k + r\nabla f_k) - \nabla f_k}{r} \\ &= (1-\nu)\nabla f_k + \nu \nabla f(\mathbf{x}_k + r\nabla f_k) \end{aligned} \tag{13}$$

where $\nu = 2\lambda/r$. We call this version *CGD with Finite Differences* (CGD-FD).

Note that this requires two gradient calls in every iteration, which might be prohibitive. Towards this it is natural to consider a stopping criterion to revert back to using steepest direction whenever the improvement through CGD-FD iterates is low. This is done to avoid the extra gradient evaluations after the optimizer has moved to a sufficiently optimal point in the domain. Such criteria are particularly helpful in loss functions where gradient evaluations are costly and therefore, budgeted. This also gives us the freedom to play around with which direction to choose (steepest or CGD-FD) and quantify when to switch to the other. For instance, we initially only move in the directions suggested by CGD-FD and switch to steepest once we make a good-enough drop in the function value from the initial point.

In our experiments, we only use CGD-FD for the first b iterations (out of the total budget T) to strive for a good-drop in function values initially. While doing so, we also ensure that we only move in the direction $\mathbf{p}_k$ (CGD-FD, Equation 13) if it is a descent direction. Otherwise, we stop using CGD-FD and use the steepest direction henceforth. We summarize our CGD-FD method in Algorithm 2.

Algorithm 2 Constrained Gradient Descent using Finite Differences (CGD-FD)

Input: Objective function $f : \mathcal{D} \to \mathbb{R}$, initial point $\mathbf{x}_0$, max iterations T, step-size α, regularization coefficients λ, stopping threshold b.
Output: Final point $\mathbf{x}_T$.

$\nu \leftarrow 2\lambda/r$, USE_CGD $\leftarrow$ TRUE
Gradient Evaluations $c \leftarrow 0$ ▷ *Can be interpreted as cost*
for $k = 0, \ldots, T-1$ **do**
  if $c = T$ **then return** $\mathbf{x}_k$ ▷ *if budget has been exhausted*
  if USE_CGD **then**
    $\mathbf{p}_k \leftarrow -(1-\nu)\nabla f_k - \nu \nabla f(\mathbf{x}_k + r\nabla f_k)$
    $c \leftarrow c + 2$
    if $\nabla f_k^T \mathbf{p}_k < 0$ **then** ▷ *Check if* $\mathbf{p}_k$ *is a Descent Direction*
      $\mathbf{x}_{k+1} \leftarrow \mathbf{x}_k + \alpha \mathbf{p}_k$
    else
      $\mathbf{x}_{k+1} \leftarrow \mathbf{x}_k - \alpha \nabla f_k$
      USE_CGD $\leftarrow$ FALSE
  else
    $\mathbf{x}_{k+1} \leftarrow \mathbf{x}_k - \alpha \nabla f_k$
    $c \leftarrow c + 1$
  if $k \geq b$ **then**
    USE_CGD $\leftarrow$ FALSE
return $\mathbf{x}_T$

3.3 CGD-QN: Quasi-Newton Variants of CGD

We consider quasi-Newton variants of CGD (CGD-QN) where a positive-definite approximation of the Hessian is maintained. The update step for CGD-QN is

given as: $\mathbf{x}_{k+1} = \mathbf{x}_k - \alpha \left(I_n + 2\lambda_k \tilde{G}_k\right) \nabla f_k$ where $\tilde{G}_k$ is the Hessian approximation at step k, which is updated after every step. Particularly Equations 4 and 5 correspond to the updates of DFP and BFGS variants of CGD respectively.

Algorithm 3 Constrained Gradient Descent: Quasi Newton (CGD-QN)

Input: Objective function $f : \mathcal{D} \to \mathbb{R}$, initial point $\mathbf{x}_0$, max iterations T, step-size α, regularization coefficient λ.
Output: Final point $\mathbf{x}_T$.

$\tilde{G}_0 \leftarrow I_n$
for iteration $k = 0, \ldots, T-1$ **do**
  $\mathbf{p}_k \leftarrow -\left(I_n + 2\lambda_k \tilde{G}_k\right) \nabla f_k$
  if $\nabla f_k^T \mathbf{p}_k < 0$ **then** ▷ *Check if* $\mathbf{p}_k$ *is a Descent Direction*
    $\mathbf{x}_{k+1} \leftarrow \mathbf{x}_k + \alpha \mathbf{p}_k$
  else
    $\mathbf{x}_{k+1} \leftarrow \mathbf{x}_k - \alpha \nabla f_k$
  Update $\tilde{G}$ according to the QN method used. For CGD-DFP and CGD-BFGS follow the updates in Equations 4 and 5 respectively.
return $\mathbf{x}_T$

4 Numerical Experiments

We test CGD-FD and CGD-QN on synthetic test functions from *Virtual Library of Simulation Experiments: Test Functions and Datasets* [1].

For our experiments we chose total budget $T = 40$ and the stopping threshold b as $T/4$. For the hyperparameter λ, we empirically tested different schedules and strategies for different kinds of functions. We observed that using constant λ works well with convex functions. For some non-convex functions, an increasing schedule is preferred. We denote this as $\mathtt{L}(a, b)$ which is an increasing linear schedule of T values going from a to b. For the step-size α, a constant-value was found to be suitable in all our experiments.

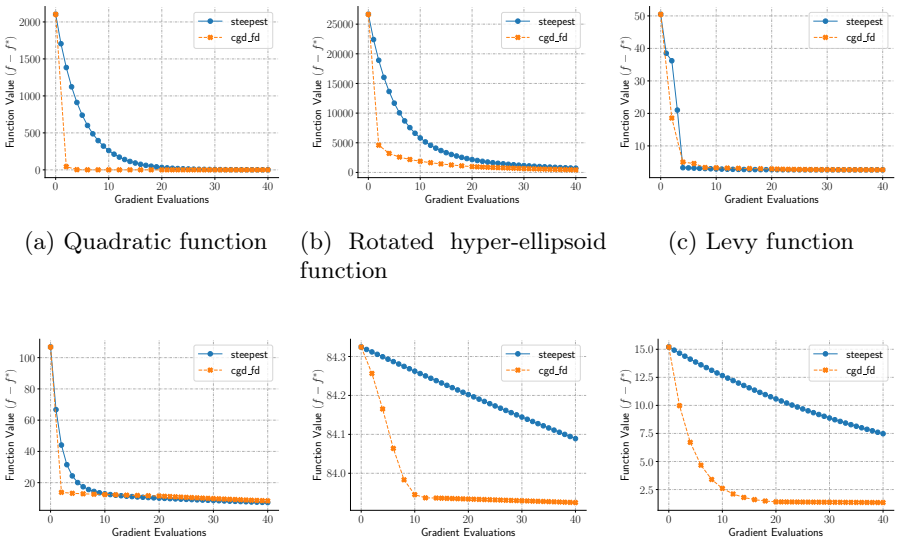
To measure the initial drop in function value, we compare the *Improvement* for the first step of CGD-FD vs the first step of steepest descent. Improvement (in %) is given as $\frac{f(\mathbf{x}_0) - f(\mathbf{x}_1)}{f(\mathbf{x}_0)} * 100$.

Table 1 summarizes our findings and Fig. 3 visualizes function values vs gradient evaluations for the chosen test functions for experiments on CGD-FD. Observe that in the first step of optimizing Quadratic function (Fig. 3a), CGD-FD has a 97.91% decrease in value compared to that of GD which only had an initial improvement of about 18.89%. Similar pattern can also be seen for other functions. This gives us the insight that CGD-FD penalizes the original function well enough to concentrate most of the decrease in value within the

[1] http://www.sfu.ca/~ssurjano

Table 1. Initial Improvements (in %) over test functions (Dimensions=n) for choices of α and λ with a budget $T = 40$ and stopping threshold $b = T/4$.

Test Function	n	λ	α	**GD**	**CGD-FD**
Quadratic function	10	0.4	0.01	18.89	**97.91**
Rotated hyper-ellipsoid function	5	0.5	0.01	15.94	**82.76**
Levy function	2	L(0.01, 0.1)	0.05	23.73	**63.21**
Branin function	2	0.07	0.01	37.53	**87.07**
Griewank function	2	40.0	0.01	0.01	**0.08**
Matyas function	2	10.0	0.01	1.83	**34.40**



(a) Quadratic function (b) Rotated hyper-ellipsoid function (c) Levy function

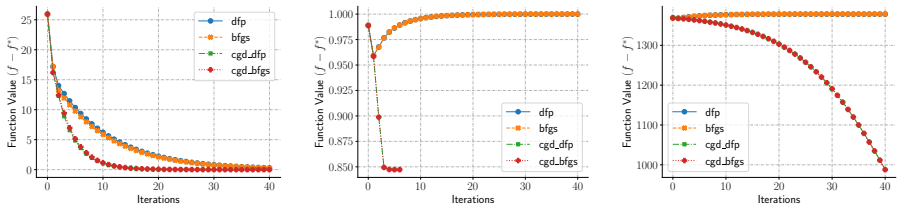
(d) Branin function (e) Griewank function (f) Matyas function

Fig. 3. Experiments for CGD-FD vs GD: Function value $f(\cdot) - f^*$ vs Gradient Evaluations. *Note*: The x-axis of each plot is not iterations but number of gradients evaluated.

first few steps of the trajectory itself. Also, notice that in Griewank function (Fig. 3e) even though the initial step improvement is low (= 0.08%), we still see a much rapid decrease in value within the first 10 steps compared to that of the GD trajectory. Thus, using CGD-FD with appropriate penalization can provide for great boosts in function value well within the starting iterates of the optimization procedure.

Experiments for CGD-QN : We ran CGD-BFGS and CGD-DFP methods against BFGS and DFP for $T = 40$. The function values vs iterations are visualized in Fig. 4. We observe that for most suitable choices of λ, CGD-DFP and CGD-BFGS exhibit similar behaviour and hence, their trajectories coincide. Another

thing to observe is that in CGD-QN methods, most improvements aren't concentrated in the initial steps since Hessian approximation is still not very great here. Instead the improvements are more gradual and appear over further steps. For example, in Zakharov function (Fig. 4a), we can observe that all methods follow the same decrease in function value for the first 1–2 steps and then the CGD-QN methods start to drop more in value. In EggHolder function (Fig. 4c) this effect is most pronounced where the improvements are very gradual but providing with better results than their vanilla counterparts.



(a) Zakharov function (b) Drop-Wave function (c) EggHolder function

Fig. 4. Experiments for CGD-QN vs QN methods: Function value $f(\cdot) - f^*$ vs iterations.

5 Reinterpretation of Explicit Gradient Regularization (EGR)

Barrett and Dherin [1] proposed **Explicit Gradient Regularization** (EGR) where the original loss function is regularized with the square of L^2-norm of the gradient. This regularized objective is then optimized with the intention that the model's parameters will converge to a *flat*-minima and thus, be more generalizable. However, an understanding of why this happens is missing.

We explain EGR through the example visualized in Fig. 5 *(Left)*. Based on our formulation explained in Section 3, we can interpret the gradient-regularized function as a steeper version of the original loss function wherein the minima remain same. Due to the gradient penalty, a steep minimum (in the original loss function) would turn steeper while the increase in steepness wouldn't be this high at a flatter minimum. Thus, at a constant step-size, the optimizer will likely overshoot over the sharper minimum due to the extremely high gradient value, while still being able to converge to the less-steeper and preferred flat minimum point. In Fig. 5 *(Left)* we see how GD gets stuck at a suboptimal point of the function while CGD (GD over the penalized loss function) is able to avoid this point and converges to a better (more optimal) minimum point.

EGR has the drawbacks of having fictitious minima identified in Lemma 1. Since, the loss landscape of neural networks is highly uneven [11], it's likely that

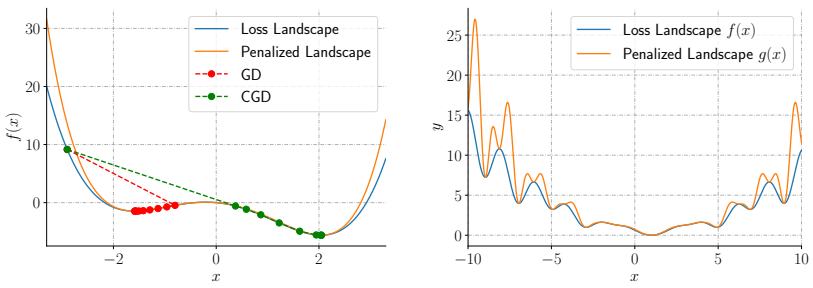


Fig. 5. *(Left)* CGD is able to move to a better local minimum by moving along the negative gradients over the modified loss function. *(Right)* Local maxima of the original loss function turning into local minima of the penalized loss function.

artificial stationary points are introduced and the optimizer might converge at local-maxima points since their nature changes with regularization. For example, in Fig. 5 *(Right)* we observe artificial stationary points being introduced between local-minima and local-maxima of the original loss function while some local-maxima also turn into local-minima of the penalized loss function. Therefore, one needs to ensure that there are suitable fixes for these scenarios for better performance. Specifically, the direction along of the penalized loss function, $-\nabla g(\cdot)$ should be a descent direction and that we shouldn't stop at points where $\nabla g(\cdot) = \mathbf{0}$ but $\nabla f(\cdot) \neq \mathbf{0}$ as discussed in Lemma 1.

Another downfall with EGR is that it requires hessian evaluation for each mini-batch while training. This might become a costly operation for bigger models and hence some approximation of the hessian should be used. Another thing we observe from the experimental results for CGD-FD, is that the improvement through optimizing the regularized function is mostly only during the initial steps. Hence, one should only use EGR for some initial steps to reach a good-enough starting point while not exhausting the budget for gradient evaluations.

6 Future Works

In this work, we considered a new line search method that penalizes the norm of the gradient and provides iterates that have lower gradient norm compared to vanilla gradient descent. We identify properties of this algorithm, provide a variant that does not require Hessian and illustrate connections to the widely popular explicit gradient regularization literature.

There are several future directions arising from this work. We would like to investigate in greater detail the role of different penalty functions $h(\cdot)$. We would also like to investigate applications of this method in more diverse settings like reinforcement learning and even Bayesian optimization. We hope this work sparks more discussions on gradient regularization helping in neural network generalization.

Acknowledgments. We thank IHub-Data, IIIT Hyderabad for the research fellowship supporting this work. This work was also supported by the SERB MATRICS project (Grant No. MTR/2023/000042) from the Science and Engineering Research Board (SERB).

Disclosure of Interests. The authors have no competing interests to declare that are relevant to the content of this article.

References

1. Barrett, D.G.T., Dherin, B.: Implicit gradient regularization (2022)
2. Beck, A.: Introduction to nonlinear optimization: Theory, algorithms, and applications with MATLAB. SIAM (2014)
3. Broyden, C.G.: Quasi-newton methods and their application to function minimisation. Mathematics of Computation **21**(99), 368–381 (1967)
4. Chaudhari, P., Choromanska, A., Soatto, S., LeCun, Y., Baldassi, C., Borgs, C., Chayes, J., Sagun, L., Zecchina, R.: Entropy-sgd: Biasing gradient descent into wide valleys. Journal of Statistical Mechanics: Theory and Experiment **2019**(12), 124018 (2019)
5. Du, J., Zhou, D., Feng, J., Tan, V., Zhou, J.T.: Sharpness-aware training for free. In: Koyejo, S., Mohamed, S., Agarwal, A., Belgrave, D., Cho, K., Oh, A. (eds.) Advances in Neural Information Processing Systems. vol. 35, pp. 23439–23451. Curran Associates, Inc. (2022)
6. Fletcher, R.: A new approach to variable metric algorithms. The Computer Journal **13**(3), 317–322 (01 1970)
7. Hochreiter, S., Schmidhuber, J.: Flat Minima. Neural Computation **9**(1), 1–42 (01 1997)
8. Karakida, R., Takase, T., Hayase, T., Osawa, K.: Understanding gradient regularization in deep learning: Efficient finite-difference computation and implicit bias. In: Krause, A., Brunskill, E., Cho, K., Engelhardt, B., Sabato, S., Scarlett, J. (eds.) Proceedings of the 40th International Conference on Machine Learning. Proceedings of Machine Learning Research, vol. 202, pp. 15809–15827. PMLR (23–29 Jul 2023)
9. Karimi, H., Nutini, J., Schmidt, M.: Linear convergence of gradient and proximal-gradient methods under the polyak-łojasiewicz condition (2020)
10. Keskar, N.S., Mudigere, D., Nocedal, J., Smelyanskiy, M., Tang, P.T.P.: On large-batch training for deep learning: Generalization gap and sharp minima (Feb 2017)
11. Li, H., Xu, Z., Taylor, G., Studer, C., Goldstein, T.: Visualizing the loss landscape of neural nets. In: Bengio, S., Wallach, H., Larochelle, H., Grauman, K., Cesa-Bianchi, N., Garnett, R. (eds.) Advances in Neural Information Processing Systems. vol. 31. Curran Associates, Inc. (2018)
12. Nocedal, J., Wright, S.J.: Numerical Optimization. Springer, 2nd edn. (2006)
13. Pearlmutter, B.A.: Fast Exact Multiplication by the Hessian. Neural Computation **6**(1), 147–160 (01 1994)
14. Polyak, B.: Gradient methods for the minimisation of functionals. USSR Computational Mathematics and Mathematical Physics **3**(4), 864–878 (1963)
15. Ruder, S.: An overview of gradient descent optimization algorithms (2017)
16. Smith, S.L., Dherin, B., Barrett, D.G.T., De, S.: On the origin of implicit regularization in stochastic gradient descent (2021)

17. Zhao, Y., Zhang, H., Hu, X.: Penalizing gradient norm for efficiently improving generalization in deep learning. In: Chaudhuri, K., Jegelka, S., Song, L., Szepesvari, C., Niu, G., Sabato, S. (eds.) Proceedings of the 39th International Conference on Machine Learning. Proceedings of Machine Learning Research, vol. 162, pp. 26982–26992. PMLR (17–23 Jul 2022)
18. Zhuang, J., Gong, B., Yuan, L., Cui, Y., Adam, H., Dvornek, N., Tatikonda, S., Duncan, J., Liu, T.: Surrogate gap minimization improves sharpness-aware training (2022)

A How CGD affects the Condition Number of Quadratic Function

Let the quadratic function be $f(\mathbf{x}) = \frac{1}{2}\mathbf{x}^T Q\mathbf{x} - \mathbf{b}^T\mathbf{x}$ where $Q \succ 0$. Let $l \triangleq \lambda_{\min}(Q)$ and $L \triangleq \lambda_{\max}(Q)$.

From Equation 9 we know $\mathbf{x}_{k+1} = \mathbf{x}_k - \alpha B_k \nabla f_k$ where $B_k = I + 2\lambda Q$ and $\nabla f(\mathbf{x}_k) = Q\mathbf{x}_k - \mathbf{b}$. Thus, the error at each iterate can be given as:

$$\begin{aligned}
\|\mathbf{e}_{k+1}\| &= \|\mathbf{x}_{k+1} - \mathbf{x}^*\| = \|\mathbf{x}_k - \alpha B_k \nabla f_k - \mathbf{x}^*\| \\
&= \|\mathbf{x}_k - \alpha B_k (Q\mathbf{x}_k - \mathbf{b}) - \mathbf{x}^*\| \\
&= \|(I - \alpha B_k Q)\mathbf{x}_k + \alpha B_k \mathbf{b} - \mathbf{x}^*\| \\
&= \|(I - \alpha B_k Q)\mathbf{x}_k + \alpha B_k \mathbf{b} - \mathbf{x}^* + (\alpha B_k Q\mathbf{x}^* - \alpha B_k Q\mathbf{x}^*)\| \\
&= \|(I - \alpha B_k Q)(\mathbf{x}_k - \mathbf{x}^*) + \alpha B_k (\mathbf{b} - Q\mathbf{x}^*)\| \qquad \text{(on rearrangement)} \\
&= \|(I - \alpha B_k Q)\mathbf{e}_k\| \qquad \text{(Gradient at } \mathbf{x}^* = \mathbf{0}) \\
&\leq \|I - \alpha B_k Q\| \, \|\mathbf{e}_k\| \qquad (14)
\end{aligned}$$

Since I and Q commute, we have $(1+2\lambda l)I \preccurlyeq B_k \preccurlyeq (1+2\lambda L)I$. Also since both Q and B_k are symmetric positive-definite matrices, $\|B_k Q\|$ has the following bounds:

$$\lambda_{\min}(B_k)\,\lambda_{\min}(Q) \leq \|B_k Q\| \leq \lambda_{\max}(B_k)\,\lambda_{\max}(Q) \qquad (15)$$

$$\therefore \|I - \alpha B_k Q\| \leq (1 - \alpha\lambda_{\min}(B_k)\,\lambda_{\min}(Q)) = 1 - \alpha(1+2\lambda l)l$$

So in Equation 14,

$$\begin{aligned}
\|\mathbf{x}_{k+1} - \mathbf{x}^*\| &\leq (1 - \alpha(1+2\lambda l)l)\,\|\mathbf{x}_k - \mathbf{x}^*\| \\
&\leq (1 - \alpha(1+2\lambda l)l)^k\,\|\mathbf{x}_0 - \mathbf{x}^*\|
\end{aligned}$$

From Equation 15, substitute α as:

$$\alpha = \frac{2}{\lambda_{\max}(B_k)\lambda_{\max}(Q) + \lambda_{\min}(B_k)\lambda_{\min}(Q)} = \frac{2}{(1+2\lambda L)L + (1+2\lambda l)l}$$

$$\begin{aligned}
\Rightarrow \|\mathbf{x}_{k+1} - \mathbf{x}^*\| &\leq \left(\frac{(1+2\lambda L)L - (1+2\lambda l)l}{(1+2\lambda L)L + (1+2\lambda l)l}\right)^k \|\mathbf{x}_0 - \mathbf{x}^*\| \\
&= \left(\frac{\lambda_{\max}(B_k)\lambda_{\max}(Q) - \lambda_{\min}(B_k)\lambda_{\min}(Q)}{\lambda_{\max}(B_k)\lambda_{\max}(Q) + \lambda_{\min}(B_k)\lambda_{\min}(Q)}\right)^k \|\mathbf{x}_0 - \mathbf{x}^*\| \\
&= \left(\frac{\kappa - 1}{\kappa + 1}\right)^k \|\mathbf{x}_0 - \mathbf{x}^*\|
\end{aligned}$$

where $\kappa = \kappa(B_k)\kappa(Q)$ where $\kappa(A) \triangleq \frac{\lambda_{\max}(A)}{\lambda_{\min}(A)}$ is the condition number of matrix A.