Gradient-only line searches: An Alternative to Probabilistic Line Searches

Dominic Kafka¹ and Daniel Wilke¹

¹Centre for Asset and Integrity Management (C-AIM), Department of Mechanical and Aeronautical Engineering, University of Pretoria, Pretoria, South Africa
{dominic.kafka@gmail.com, wilkedn@gmail.com}

March 25, 2019

Abstract
Step sizes in neural network training are largely determined using predetermined rules such as fixed learning rates and learning rate schedules, which require user input to determine their functional form and associated hyperparameters. Global optimization strategies to resolve these hyperparameters are computationally expensive. Line searches are capable of adaptively resolving learning rate schedules. However, due to discontinuities induced by mini-batch sampling, they have largely fallen out of favor. Notwithstanding, probabilistic line searches have recently demonstrated viability in resolving learning rates for stochastic loss functions. This method creates surrogates with confidence intervals, where restrictions are placed on the rate at which the search domain can grow along a search direction.

This paper introduces an alternative paradigm, Gradient-Only Line Searches that are inexact (GOLS-I), as an alternative strategy to automatically resolve learning rates in stochastic cost functions over a range of 15 orders of magnitude without the use of surrogates. We show that GOLS-I is a competitive strategy to reliably resolve step sizes, adding high value in terms of performance, while being easy to implement. Considering mini-batch sampling, we open the discussion on how to split the effort to resolve quality search directions from quality step size estimates along a search direction.

Keywords: Artificial Neural Networks, Gradient-only, Line Searches, Learning Rates

1 Introduction
Selecting learning rate related parameters is still an active field of research in deep learning [Smith, 2015, Orabona and Tommasi, 2017, Wu et al., 2018], since they have been shown to be the most sensitive hyperparameters in training [Bergstra and Bengio, 2012]. In practice, these parameters are often selected a priori by the user. However, in mathematical programming, a common strategy is to resolve step sizes (learning rates) is the use of line searches [Arora, 2011]. Stochastic sub-sampling spoils the utility of conventional line searches in neural network training, since it introduces discontinuities into the cost functions and gradients, causing line searches that minimize along a descent direction to become stuck in false minima resulting from discontinuities [Wilson and Martinez, 2003, Schraudolph and Graepel, 2003, Schraudolph et al., 2007]. This has resulted in line searches being replaced by a priori rule based step size schedules typical of subgradient methods that includes stochastic gradient descent [Schraudolph, 1999, Boyd et al., 2003, Smith, 2015]. Consider for example Figure 1 that depicts both function values

1
Figure 1: Gradient information can be more stable than function values in the context of sub-sampling, making it useful for use in line searches. Though more importantly, the gradient-only optimum equivalent definition, the Non-Negative Associated Gradient Projection Point (NN-GPP), is much more robust in stochastic loss functions.

and directional derivatives\(^1\) of a simple neural network as applied to the famous Iris dataset [Fisher, 1936]. The top row of Figure 1 depicts the function values and directional derivatives when all the samples are used in the training data, where the bottom row depicts the function values and directional derivatives when a single training data point is randomly removed for every function and gradient evaluation.

Recently, Gaussian processes incorporating both function value and gradient information along search directions have successfully been used to construct line searches in stochastic environments via Bayesian optimization methods [Mahsereci and Hennig, 2017]. However, we postulate that a simpler and more accessible approach may be sufficient to construct line searches, using only gradient information. The premise for this postulate is that the severity of the discontinuities in the function values are more abrupt than the directional derivatives which are considerably more robust as is evident in Figure 1. In this paper we demonstrate how this characteristic in conjunction with estimating Non-Negative Associated Gradient Projection Point (NN-GPP) [Wilke et al., 2013; Snyman and Wilke, 2018], allows for the construction of gradient-only line searches to automatically resolve step sizes.

The NN-GPP merely presents an alternative solution to a function minimizer for discontinuous functions, i.e. instead of minimizing the discontinuous stochastic function directly, we use the associated derivative [Wilke et al., 2013; Snyman and Wilke, 2018] to filter out all discontinuities from the stochastic function. Essentially, when we only consider associated derivatives to make decisions during line searches we may interpret the discontinuous stochastic function presented in Figure 2(a) to be the stochastic continuous function presented in Figure 2(c), since both are consistent with the associated derivatives presented in Figure 2(b). It is clear that the function minimizer of the discontinuous stochastic function, depicted as a gray dot in Figure 2(a), is associated with a negative directional derivative in its neighborhood along the direction \(\alpha = +1\). This implies that the global function minimizer present in the discontinuous stochastic function is

\(^1\)Consider the search direction \(d\), where the \(d\) is zero at all entries apart from those corresponding to \([x_8, x_9]\), which have value \(\frac{1}{\sqrt{2}}\) resulting in a normalized direction. Select a starting point \(x_0\). We sample on a regular grid \(-50 \leq x_8 \leq 50\) and \(-50 \leq x_9 \leq 50\) and compute the function values and directional derivatives using \(d\).
Figure 2: (a) Discontinuous stochastic function with (b) associated derivatives and (c) an alternative interpretation of (a) that is consistent with the derivatives given in (b). The function minimizer of \( F(\alpha) \) (gray dot) and sign change from negative to positive (red dot) are indicated.

not representative of a local minimum according to the associated derivatives. \cite{Wilke2013, Snyman2018}. This is because a global or local minimum would be characterized by a directional derivative going from negative to positive along a descent direction. Traditionally, for \( C^1 \) smooth functions the derivative would be zero at the local minimum, indicative of a critical point \cite{Snyman2018}. Fortunately, since NN-GPP were developed for discontinuous functions, \cite{Wilke2013, Snyman2018}, it does not rely on the concept of a critical point as there is usually no point where the derivative is zero when discontinuous stochastic functions are considered. A NN-GPP only requires the directional derivative to change sign from negative to positive as one travels along a descent direction. As outlined by \cite{Wilke2012}, this way of characterizing solutions of discontinuous stochastic functions is also consistent with solutions that sub-gradient algorithms or stochastic gradient descent would find, i.e. using stochastic gradient descent to optimize Figure 2(a) would only result in converge around the NN-GPP (red dot), while the global function minimizer (gray dot) would be completely ignored.

In this study, based on empirical evidence, we argue that developing line searches that locate NN-GPPs offers two advantages: 1) it offers a more representative (and consistent) way to define candidate solutions of a discontinuous stochastic cost function, and 2) allows for solutions to be isolated more robustly and with lower variance by the line search as some solutions are filtered out.

2 Cost functions in Machine learning

Commonly, the objective functions used in machine learning training are of the form

\[
\mathcal{L}(\mathbf{x}) = \frac{1}{M} \sum_{k=1}^{M} \mathcal{L}(\mathbf{x}; t_k),
\]

where \( \{t_1, \ldots, t_M\} \) is a training dataset of size \( M \), \( \mathbf{x} \in \mathbb{R}^d \) is an \( n \)-dimensional vector of model parameters, and \( \mathcal{L}(\mathbf{x}; t) \) defines the loss quantifying the fitness of parameters \( \mathbf{x} \) with regards to training sample \( t \). Backpropagation \cite{Werbos1994} allows for the computation of the exact gradient w.r.t. \( \mathbf{x} \) as follows:

\[
\nabla \mathcal{L}(\mathbf{x}) = \frac{1}{M} \sum_{k=1}^{M} \nabla \mathcal{L}(\mathbf{x}; t_k).
\]
Figure 3: (a) Function values and (b) the directional derivatives of the cost function in dimensions $x_8$ and $x_9$ for a single hidden layer neural network applied to the Iris dataset problem [Fisher, 1936]. Directional derivatives are generated using a fixed search direction $\mathbf{d}$, where only entries corresponding to $x_8$ and $x_9$ have value $\frac{1}{\sqrt{2}}$. This direction is then evaluated in $\frac{dE_n}{d\alpha} = \mathbf{g}(\mathbf{x}_n) \cdot \mathbf{d}$, to produce the generated plots. Say we hold back some test and validation data, such that the training data available is $M = 76$. When using full batches both the function value and the directional derivatives evaluations produce smooth functions. (c) Function values and (d) the directional derivatives are discontinuous, when mini-batch samples of size $|B| = 10$ are implemented. The function value plot shape is not recognizable in comparison to (a), while directional derivatives still contain features of the original shape.

In the limit case, where all the training data is used for both function and gradient evaluations, $L(\mathbf{x})$ and $\nabla L(\mathbf{x})$ are smooth. We demonstrate this in our test example in Figures 3(a) and (b). In smooth environments such as these, minimization line search methods are capable of locating local minima. However, the cost of computation is high, due to processing $M$ datapoints at every function evaluation. The minimization line search is also more likely to become "stuck" in a smooth local minimum within the multi-modal and non-convex cost function.

Using mini-batches of the data during training decreases the computational cost and increases the chance of an optimization algorithm overcoming local minima. This changes the form of the cost function as follows: Mini-batches, $B \subset \{1, \ldots, M\}$ of size $|B| \ll M$ are sampled from the training set of size $M$, resulting in an approximate loss function

$$L(\mathbf{x}) = \frac{1}{|B|} \sum_{k \in B} \ell(\mathbf{x}; \mathbf{t}_k), \quad (3)$$
and corresponding approximate stochastic gradient

$$g(x) = \frac{1}{|B|} \sum_{k \in B} \nabla \ell(x; t_k).$$

(4)

The approximate loss has expectation $E[L(x)] = L(x)$ and corresponding expected gradient $E[g(x)] = \nabla L(x)$ [Tong and Lin, 2005], but individual instances may vary significantly from the mean. This implies that the first order optimality criterion $g(x_{\text{min}}) = 0$ [Arora, 2011] may not exist for the instance of mini-batch $B$, even if it may exist for the full batch case, $\nabla L(x_{\text{min}}) = 0$.

For discontinuous functions, Wilke et al. [Wilke et al., 2013, Snyman and Wilke, 2018] proposed the *gradient-only* optimality criterion given by:

$$d \cdot g(x_{\text{nn-gpp}} + \epsilon d) \geq 0 \quad \forall d \in \mathbb{R}^n : \|d\| = 1,$$

(5)

as an alternative to the first order optimality criterion $g(x_{\text{min}}) = 0$ [Arora, 2011]. Candidate solutions of the *gradient-only* optimality criterion developed for discontinuous functions, are defined as *Non-Negative Associative Gradient Projection Points* (NN-GPPs) [Wilke et al., 2013, Snyman and Wilke, 2018].

For smooth functions, NN-GPP is equivalent to finding critical points $g(x_{\text{min}}) = 0$ that are semi-positive definite [Wilke et al., 2013, Snyman and Wilke, 2018]. Hence, NN-GPP incorporates second order information in the form of requiring that there are no descent directions from NN-GPP.

For notational convenience, we define a univariate function $F_n(\alpha)$ along a descent direction, $d_n \in \mathbb{R}^n$ from $x_n \in \mathbb{R}^n$:

$$F_n(\alpha) = f(x_n(\alpha)) = L(x_n + \alpha d_n),$$

(6)

with associated derivative

$$\frac{dF_n(\alpha)}{d\alpha} = d_n \cdot g(x_n + \alpha d_n).$$

(7)

### 3 Our Contribution

In this paper, we automatically resolve learning rates over a range of 15 orders of magnitude for stochastic loss functions using gradient-only line searches. We propose an *Inexact Gradient-Only Line Search* (GOLS-I) method that isolates Non-Negative Associate Gradient Projection Points (NN-GPP). As argued before, when considering univariate functions, a NN-GPP is merely a sign change from negative to positive in the univariate directional derivative along the descent direction.

Importantly, we select a new mini-batch sub-sample from the training data at every evaluation of the loss function within the line search. We stress again that we do not rely on the concept of a critical point as we do not require the derivative at a NN-GPP to be zero. For multi-dimensional functions this naturally requires that we search for a sign change from negative to positive in the directional derivative along a descent direction. Since we require a sign change from negative to positive along a descent direction, and not from positive to negative, we incorporate some second information, i.e. the requirement of a local minimum.

Commonly used learning rate schedules use step sizes ranging over 5 orders of magnitude [Senior et al., 2013], while the magnitudes of cyclical learning rate schedules typically range over 3 to 4 orders of magnitude [Smith, 2015, Loshchilov and Hutter, 2016]. Manually selected schedules can require a number of hyperparameters to be determined. Our proposed method, GOLS-I, can resolve step sizes over a range of 15 orders of magnitude.

The high range of available step sizes within the line search allow GOLS-I to effectively traverse flat planes or steep declines in discontinuous stochastic cost functions, while requiring no user intervention.
3.1 Empirical evidence that NN-GPP is more robust than minimizers

We present empirical evidence that indicates that NN-GPPs offers a more representative and consistent way to define candidate solutions for discontinuous stochastic cost functions, as well as, allowing solutions to be isolated more robustly and with lower variance by a line search as some sporadic minima are filtered out.

Consider the Iris test problem where we sample along the search direction with only non-zero elements $x_8$ and $x_9$ equal to $\frac{1}{\sqrt{2}}$. Along this direction in 100 increments of $\alpha$, we note the locations of all the minimizers and NN-GPP. We repeat this procedure 100 times for different sample sizes $|B|$ and construct the distributions determining the locations of minima and NN-GPP observed in Figure 4. The spatial distribution of local minima across the sampled domain approximate a uniform distribution. The location of the true minimum is identified by the full batch $|B| = M$. Conversely, the spatial location of NN-GPPs are constrained in what resembles a Gaussian distribution around the true minimum, with variance inversely proportional to the sample size $|B|$. The central message of these plots is that the spatial location of NN-GPP is restricted, making it a reliable metric to be implemented to resolve step sizes in stochastic cost functions. Additionally, the NN-GPP definition generalizes to the minimization definition in the limit case of using the full batch $|B| = M$.

```
\begin{figure}[h]
\centering
\includegraphics[width=\textwidth]{figure4.png}
\caption{(a) Function values and (b) directional derivatives along search direction $d$, the direction where only components $[x_8, x_9]$ have non-zero value $\frac{1}{\sqrt{2}}$. The cost function is obtained from the Iris classification problem of Figure 3 [Fisher, 1936]. The search direction is sampled by 100 points with sample sizes ranging from $|B| = 10$ to $|B| = M = 150$. This is repeated 100 times and the average number of minima and NN-GPP found at every point is plotted. Minima are spread across the entire domain for most sample sizes in (a). The full batch located identifies the true minimum. The spatial spread of NN-GPP is neatly localized around the true minimum with increasing spread for decreasing sample size. However, even with a small batch size $|B|$, the spatial location remains bounded.}
\end{figure}
```

4 Algorithmic details

We propose GOLS-I, the following inexact gradient-only line search method. Given an initial ($n = 1$) descent direction $d_n$, with initial ($i = 1$) step size $\alpha_{n,1}$ and real scaling parameter $\eta \in \{R^+ \mid \eta > 1\}$. First it is determined whether the update can be accepted without further refinement. Towards this we consider a modified strong Wolfe-condition

$$0 < \frac{dF_n(\alpha)}{d\alpha} \leq c_2|\frac{dF_n(0)}{d\alpha}|,$$  \hspace{1cm} (8)
Figure 5: Illustration of the method for Inexact Gradient-Only Line Search (GOLS-I): A new mini-batch is drawn for every evaluation of the loss, resulting in a discontinuous function. Step sizes are increased or decreased by a given factor, $\eta$ (in our case $\eta = 2$) from an initial guess until a sign change is found. If the initial guess satisfies Equation (8), it is immediately accepted, reducing the cost of the algorithm.

with $c_2 > 0$. Hence, the initial update step will be taken as is when the directional derivative is positive, but with a restricted magnitude w.r.t. the initial descent magnitude $|dF_n(0)|$ for the $n$th direction. This implies that we have stepped over the sign change in a controlled fashion. The reason why we consider this update is that it has been found to work better than the strong Wolfe condition [Arora, 2011]

$$|dF_n(\alpha_{n,i})| \leq c_2 |dF_n(0)|,$$

which also allows some restricted negative directional derivative to be acceptable. Hence, our studies have found that larger step sizes are preferred over smaller step sizes for computationally as well as generalization benefits for the architectures under consideration in this study. We note that it is of some importance to conduct a more comprehensive study for a wider group of architectures to properly understand this empirically observed asymmetry around a sign change.

Should the initial step not be acceptable, the following decisions are made, based on the sign of the directional derivative $dF_n(\alpha_{n,i})$ at initial guess, $\alpha_{n,1}$: If $dF_n(\alpha_{n,i}) < 0$, then $\alpha_{n,i+1} = \eta \alpha_{n,i}$ where-after $i := i + 1$ until $dF_n(\alpha_{n,i}) > 0$. Alternatively, if $dF_n(\alpha_{n,i}) > 0$, then $\alpha_{n,i+1} = \frac{\alpha_{n,i}}{\eta}$ where-after $i := i + 1$ until $dF_n(\alpha_{n,i}) < 0$. The $\alpha_{n,i}$ at which either conditions terminates is used as the acceptable update $\alpha_{n,I_n}$ and the next search direction $d_{n+1}$ is computed. For the $n$th search direction the update domains are illustrated in Figure 5.

Depending on the nature of the problem (loss function, architecture, activation function etc.), for small mini-batch sizes it is possible to obtain divergent behavior where no sign change is located along a search direction for many consecutive $\alpha$ updates. We therefore introduce a maximum step $\alpha_{n,i} \leq \alpha_{max}$ size to protect the line search from divergent steps. Inspired by the Lipschitz condition for convergent fixed step sizes, we choose the maximum step size conservatively as

$$\alpha_{max} = \min\left(\frac{1}{\|d_n\|_2}, 10^7\right).$$

The Euclidean norm of the descent direction limits the line search towards more
conservative updates for steep search directions, but allows larger update steps for faster progress over flat planes. The upper bound restricts divergent behaviour from unreliable directions in flat planes.

Step sizes are restricted to a minimum to avoid expensive line searches that may reduce step sizes to approach 0, in cases where a computed descent direction is statistically unlikely, which is given by

$$\alpha_{\text{min}} = 10^{-8}. \quad (11)$$

As a result of these bounds, the line search can resolve an iteration specific step size over 15 orders of magnitude. We do not set a cap on the number of gradient evaluations allowed per iteration, which is common practice in other line search approaches used in machine learning training [Mahsereci and Hennig 2017].

For the first search direction of GOLS-I, i.e. \( n = 1 \) along \( d_1 \), a conservative initial guess of \( \alpha_{1,1} = \alpha_{\text{min}} \) is chosen. This is an overly conservative assumption based on gradients being steep in the beginning of optimization and the length scale of the problem not being initially known. GOLS-I is then grows the step size until the length scale of the first sign change is determined. In practice the initial guess can be increased, but having a small initial guess in our investigations also demonstrates that the method is capable of automatically adjusting the step size magnitude in a single iteration. In subsequent iterations, \( n > 1 \), the initial guess along the next search direction \( d_{n+1} \) is that of the previous iteration, \( \alpha_{n,1} = \alpha_{n-1,1} \). A conceptual summary of GOLS-I is given in Algorithm 1, while a detailed pseudo code can be reviewed in the Appendix under listing Algorithm 2.

### Algorithm 1: GOLS-I: Inexact Gradient-Only Line Search, a conceptual outline.

**Input:** \( F_n(\alpha), d_n, \alpha_{n,1} \)  
**Output:** \( \alpha_{n,I} \)

1. Define constants: \( \alpha_{\text{min}} = 10^{-8}, \text{flag} = 1, \eta = 2, c_2 = 0.9 \)
2. Evaluate \( F_n'(0) \) (this can also be inherited from previous iteration, \( n - 1 \))
3. Evaluate positive Wolfe condition \( W = |c_2 F_n'(0)| \)
4. Define upper limit and enforce on \( \alpha_{n,1} \) if necessary: \( \alpha_{\text{max}} = \min\left( \frac{1}{||d_n||^2}, 10^7 \right) \)
5. Evaluate \( F_n'(\alpha_{n,1}) \)
6. if \( F_n'(\alpha_{n,1}) > 0 \) then
   7.   flag = 1, decrease step size
8. if \( F_n'(\alpha_{n,1}) < 0 \) then
   9.   flag = 2, increase step size
10. if \( F_n'(\alpha_{n,1}) > 0 \) and \( F_n'(\alpha_{n,1}) < W \) then
    11.   flag = 0, immediate accept clause
12. while \( \text{flag} > 0 \) and \( \alpha_{n,i} > \alpha_{\text{min}} \) and \( \alpha_{n,i} < \alpha_{\text{max}} \) do
    13.     if \( \text{flag} = 2 \) then
    14.         \( \alpha_{n,i+1} = \alpha_{n,i} \cdot \eta \) until \( F_n'(\alpha_{n,i+1}) > 0 \), then \( \text{flag} = 0 \)
    15.     if \( \text{flag} = 1 \) then
    16.         \( \alpha_{n,i+1} = \frac{\alpha_{n,i}}{\epsilon} \) until \( F_n'(\alpha_{n,i}) < 0 \), then \( \text{flag} = 0 \)

4.1 Proof of Global Convergence for Full Batch Sampling

Suppose that the loss function \( \mathcal{L} \) obtained from full batch sampling is smooth, coercive with a unique minimizer \( \mathbf{x}^* \). Any Lipschitz function \( \hat{\mathcal{L}} \) can be regularized to be coercive using Tikhonov regularization with a sufficient large regularization coefficient.

The step updates of an optimization algorithm can be considered as a dynamical
system in discrete time:

\[ x_{k+1} = D(x_k), \quad D : \mathbb{R}^n \rightarrow \mathbb{R}^n. \] (12)

It follows from Lyapunov's global stability theorem [Aleksandr M., 1992] in discrete time that any Lyapunov function \( \Gamma(x) \) defined by positivity, coercive and expected strict decrease:

1. Positivity: \( \Gamma(0) = 0 \) and \( \Gamma(x) > 0 \), \( \forall x \neq 0 \)
2. Coercive: \( \Gamma(x) \rightarrow \infty \) as \( x \rightarrow \infty \)
3. Strict descent: \( \Gamma(D(x)) < \Gamma(x) \), \( \forall x \neq 0 \)

results in \( x_k \rightarrow 0 \) as \( k \rightarrow \infty \), \( \forall x_0 \in \mathbb{R}^n \).

**Theorem 4.1** Let \( f(x) \) be any smooth coercive function with a unique global minimum \( x^* \), for \( x_{k+1} = D(x_k), \forall x_k \neq x^*_k \) restricted such that \( f(\alpha x_{k+1} + (1 - \alpha)x_k) < f(x_k) \), \( \forall \alpha \in (0,1] \). Then \( D \) will result in updates that are globally convergent.

Let the error at step \( k \) be given by \( e_k := x_k - x^* \) for which we can construct the Lyapunov function \( \Gamma(e) = f(e + x^*) - f(x^*) \). It follows that \( \Gamma(0) = 0 \) and that \( \Gamma(e) > 0 \), \( \forall e \neq 0 \), since \( x^* \) is a unique global minimum of \( f \).

At every iteration our line search update locates a NN-GPP along the descent direction \( d_k \), by locating a sign change from negative to positive along \( d_k \). Wilke et al. [2013] proved this to be equivalent to minimizing along \( d_k \) when \( f(x_k + \alpha d_k) \) is smooth and the sign of the directional derivative \( \nabla f(x_k + \alpha d_k) d_k \), is negative \( \forall \alpha \in (0, \alpha_k^* \) along \( d_k \). Here, \( x_k^* \) defines the step length to the first minimum along the search direction \( d_k \). It is therefore guaranteed that \( f(x_{k+1}) < f(x_k) \) at every iteration \( k \). In addition, \( f(\alpha x_{k+1} + (1 - \alpha)x_k) < f(x_k), \forall \alpha \in (0,1] \) ensures that for our choice of discrete dynamical update \( D \), we can always make progress unless \( x_k = x^* \). Hence, for any \( e_k \neq 0 \) it follows that

\[ \Gamma(e_{k+1}) = f(x_{k+1} - x^* + x^*) - f(x^*) < f(x_k - x^* + x^*) - f(x^*) = \mathcal{L}(e_k). \]

It then follows from Lyapunov’s global stability theorem that \( e_k \rightarrow 0 \) as \( k \rightarrow \infty \). Hence \( \forall x_0 \) we have that \( x_k \rightarrow x^* \), which proves that finding NN-GPP at every iteration \( k \) results in a globally convergent strategy.

### 4.2 Proof of Global Convergence for Mini-Batch Sampling

Consider the discontinuous loss function \( L \) obtained from mini-batch sampling with smooth expected response \( E(L) \) and unique expected minimizer \( x^* \). Assume that the function \( L \) is directional derivative coercive (see Wilke et al. [2013]) around a ball \( x \in B_r(x) = \{ q \mid \| q - x^* \| > r \} \) of given radius \( r \in \mathbb{R} > 0 \) that is centered around the expected minimizer \( x^* \). This implies that for given radius \( r \) and for any point outside the ball \( x_1 \in B_r(x_1) \) and any point inside the ball \( x_2 \in B_r(x_2) = \{ q \mid \| q - x^* \| < r \} \) with \( u = \| x_2 - x_1 \| \) the following must hold:

\[ \nabla f(x_2)^T u > 0. \] (13)

As before, the step updates of an optimization algorithm can be considered as a dynamical system in discrete time:

\[ x_{k+1} = D(x_k), \quad D : \mathbb{R}^n \rightarrow \mathbb{R}^n. \] (14)

We relax Lyapunov’s global stability theorem in discrete time for mini-batched sub-sampled discontinuous functions that any smooth expected Lyapunov function \( E(\Gamma(x)) \) defined by expected positivity, coercive and expected strict decrease around a ball \( x \in B_r(x) = \{ q \mid \| q - x^* \| > r \} \) of given radius \( r \in \mathbb{R} > 0 \):
1. Expected positivity: $E(\Gamma(0)) = 0$ and $E(\Gamma(x)) > 0$, $\forall x \neq 0$.

2. Coercive: $\Gamma(x) \to \infty$ as $x \to \infty$.

3. Directional derivative coercive for any point $x \in \mathcal{B}_r(x) = \{ q \mid \|q - x^*\| > r \}$ of radius $r \in \mathcal{R} > 0$.

4. Expected strict descent: $E(\Gamma(D(x))) < E(\Gamma(x))$, $\forall x \neq 0$.

results in $x_k \in \mathcal{B}_r(x_k) = \{ q \mid \|q - x^*\| < r \}$ as $k \to \infty$, $\forall x_0 \in \mathcal{R}^n$.

**Theorem 4.2** Let $f(x)$ be any smooth expected coercive function with a unique expected global minimum $x^*$ that is directional derivative coercive around a ball $\mathcal{B}_r(x) = \{ q \mid \|q - x^*\| > r \}$ of radius $r \in \mathcal{R} > 0$. Then $x_{k+1} = D(x_k)$, $\forall x_k \notin \mathcal{B}_r(x_k)$ restricted such that $\nabla^2 f(x_k + \alpha d_k) d_k < 0$, $\forall \alpha \in [0, \alpha^*_k)$ along descent direction $d_k$. Then $D$ will result in updates that globally converges to the ball $\mathcal{B}_r(x) = \{ q \mid \|q - x^*\| < r \}$ of radius $r \in \mathcal{R} > 0$ centered around $x^*$.

Let the error at step $k$ be given by $e_k := x_k - x^*$ for which we can construct the Lyapunov function $\Gamma(e) = f(e + x^*) - f(x^*)$ and expected Lyapunov function $E(\Gamma(e)) = E(f(e + x^*)) - E(f(x^*))$. It follows that $E(\Gamma(0)) = 0$ and that $E(\Gamma(e)) > 0$, $\forall e \neq 0$, since $x^*$ is a unique expected global minimum of $f$.

At every iteration our line search update locates a NN-GPP along the descent direction $d_k$, by locating a sign change from negative to positive along $d_k$. Since the function is smooth expected coercive and directional derivative coercive around a ball $\mathcal{B}_r(x)$, expected descent follows $E(\Gamma(D(x))) < E(\Gamma(x))$, $\forall x \notin \mathcal{B}_r(x) = \{ q \mid \|q - x^*\| < r \}$ of radius $r \in \mathcal{R} > 0$. It is therefore guaranteed that $E(f(x_{k+1})) < E(f(x_k))$ at every iteration $k$. In addition, $E(f(\alpha x_{k+1} + (1 - \alpha) x_k)) < E(f(x_k))$, $\forall \alpha \in (0,1]$ ensures that for our choice of discrete dynamical update $D$, we can always make progress unless $x_k \in \mathcal{B}_r(x) = \{ q \mid \|q - x^*\| < r \}$. In addition, since the function is directional derivative coercive around the ball $\mathcal{B}_r(x_k)$, any point $x \in \mathcal{B}_r(x_k)$ remains in $\mathcal{B}_r(x_k)$ due to the update requirement of a sign change from negative to positive along the descent direction. Hence, for any $e_k$ such that $\|e_k\| > r$ it follows that

$$E(\mathcal{L}(e_{k+1})) = E(f(x_{k+1} - x^* + x^*)) - E(f(x^*)) < E(f(x_k - x^* + x^*)) - E(f(x^*)) = E(\Gamma(e_k)).$$

It then follows from Lyapunov’s relaxed global stability theorem that $e_k \in \mathcal{B}_r(x)$ as $k \to \infty$. Hence $\forall x_0$ we have that $x_k \in \mathcal{B}_r(x_k)$ as $k \to \infty$, which proves that finding NN-GPP at every iteration $k$ results in a globally converges to the ball $\mathcal{B}_r(x_k)$.

## 5 Numerical Studies

The architectures and problems for the numerical studies conducted in this study are taken from [Mahsereci and Hennig, 2017] for their probabilistic line search strategy research in which they compared to stochastic gradient descent using constant step sizes. This allows for a direct comparison of our obtained results to at least the stochastic gradient descent using constant step sizes that they reported. The problems we consider are:

- Breast Cancer Wisconsin Diagnostic (BCWD) Dataset [Street et al., 1993], a binary classification problem, distinguishing between “benign” and “malignant” tumors, using 30 different features;
- MNIST Dataset [Lecun et al., 1998], a multi-class classification problem with images of handwritten digits from 0 to 9 in grey-scale with a resolution of 28x28 pixels; and
• CIFAR10 [Krizhevsky and Hinton 2009], a multi-class classification problem with images of 10 natural objects such as deer, cats, dogs, ships, etc.; the colour images have a resolution of 32x32.

Further details about the datasets, and the various parameters governing their implementation are given in Table 1. These details are used as given by [Mahsereci and Hennig 2017] (“the authors”), where the dataset problems are trained with different network architectures, fixed step size and line search methods, and different corresponding batch sizes. Our implementation was done using PyTorch 1.0. All datasets were pre-processed using the standard transformation (Z-transform).

| Dataset   | Training obs | Test obs | Input dim. | Output dim. | Net structure | Max. F.E. | [8] for training |
|-----------|--------------|----------|------------|-------------|---------------|-----------|-----------------|
| BCWD      | 400          | 169      | 30         | 2           | Log. Regression | 100000    | 10,50,100,400   |
| MNIST     | 60000        | 10000    | 784        | 10          | NetI, NetII   | 40000     | 10,100,200,1000 |
| CIFAR     | 10000 (Batch1) | 10000    | 3072       | 10          | NetI, NetII   | 10000     | 10,100,200,1000 |

Table 1: Relevant parameters related to the datasets used for numerical experiments. Training occurs over a fixed number of function evaluations.

Following [Mahsereci and Hennig 2017], both MNIST and CIFAR10 are implemented using two different network architectures, NetI and NetII. Including the logistic regression for the BCWD Dataset, this constitutes a total of 5 architectures to be used in the numerical study. The parameters concerning the implementations of the different architectures are summarized in Table 2. All networks are fully connected, and the detail given concerning the hidden layers of the network excludes the biases, although they are included. [Mahsereci and Hennig 2017] have stated that a normal distribution was used to initialize all networks. However, we found that anything resembling comparable results could not be obtained for NetII (with with constant step sizes or otherwise), unless Xavier initialization [Glorot and Bengio 2010] was used.

| Network          | Hidden layer architecture | Activation func. | Initialization | Loss func.       | Fixed step sizes |
|------------------|----------------------------|------------------|----------------|------------------|------------------|
| log. Regression  | N/A                        | Sigmoid          | N(0,1)         | Binary cross entropy | 1,10,100        |
| NetI (MNIST)     | 800                        | Sigmoid          | N(0,1)         | Cross entropy    | 1e-2,1e-1,1     |
| NetII (CIFAR10)  | 1000,500,250               | Tanh             | Xavier         | Mean Squared Error | 1e-1,1.1,4      |

Table 2: Parameters and settings governing the implemented network architectures and their training.

We conducted an extensive study using numerous fixed step sizes for the different architectures and problems. In our analyses we chose three constant step sizes, each one order of magnitude apart, ensuring that the full training performance modality is captured. This means that step sizes selected within the 3 orders of magnitude encapsulates a potential optimal constant step size. Thus we select a small, a medium and a large constant step size, along the following guidelines:

• Small: Resembles a slow and overly conservative learning rate that leads to wasted gradient computations during training.

• Medium: Resembles an effective and efficient learning rate with desired convergence performance.

• Large: Resembles a learning rate that is aggressive and usually leads to detrimental performance.

The training algorithm used in this study is Stochastic Gradient Descent (SGD) [Robbins and Monro 1951]. We apply both our line search methods GOLS-I, as well as the 3 fixed step sizes assigned, to every network architecture shown in Table 2. For each of
the 4 different step size schemes (GOLS-I and 3 constant leaning rates), 10 runs were conducted, using the same initial seeds.

The training and test classification errors (as evaluated on the full respective training and test datasets) are evaluated and noted during training. The resolution of these plots is therefore limited by the size of the respective datasets.

For the benefit of reproducible science, we highlight results that could not be exactly recovered according to the information supplied by Mahsereci and Hennig [2017]: It was not possible to obtain the same refinement for error plots given the number of data points in the BCWD dataset. Additionally, the test errors obtained for CIFAR10 in NetI did not approach the same values as those shown by Mahsereci and Hennig [2017]. This is true even for the ”best” fixed step analyses as indicated in their work. Though we believe that a reasonable investigation into the source of these discrepancies has been conducted, we are open to the possibility that there are unidentified inconsistencies between the implementations of Mahsereci and Hennig [2017] and ours or - perish the thought - between PyTorch and Matlab which they used in their study. For these reasons we cannot directly compare with their results but we can compare against our implementations of the constant step sizes they considered in their studies. We therefore use their results as guidelines, but not absolutes. For this reason we include fixed steps as relative comparisons and wish to demonstrate that GOLS-I is more effective strategy than seeking for an effective fixed step. Nevertheless, in order to aid comparison where possible, we use similar ranges on the axes and match the layout of our plots to those of the authors.

6 Results

6.1 Breast Cancer Wisconsin Diagnostic (BCWD) Dataset

We plot the log of the training error, log of the training loss, log of the the test error, and log of the step size for the BCWD using mini-batch sizes of $|\mathcal{B}| \in \{10, 50, 100, 400\}$, in Figure 6. Note $|\mathcal{B}| = 400$ is indicative of a full batch and is representative of a smooth loss function. Since we do not cap the cost of the line search, the number of gradient evaluations per iteration varied from 1 (immediate accept) to 17. On average, the number of gradient evaluations per iteration is in the low 2’s. Hence, all results are listed in terms of the number of gradient evaluations, as it quantifies the value added by the line search when compared to the equivalent computational cost of a fixed step size method. To avoid unfortunate scaling of figures in the log domain, the minimum training error was clipped to $10^{-4}$, as indicated by the lowest training error in Figure 6. However, in the interest of unrestricted convergence comparison, no clipping was applied in the log of the training loss plot.

Let us first consider the performance of the constant step size line searches. As expected, the small constant step size exhibits slow convergence, the medium step size performs well, and the large step size often leads to divergence. As the batch size increases, the performance of the large constant step size performs better for isolated instances as is evident for $|\mathcal{B}| = 100$ and $|\mathcal{B}| = 400$.

The unclipped log of the training loss for this problem gives a better perspective of the convergence behaviour of GOLS-I. For $(|\mathcal{B}| = 10)$ the variance in the computed gradient between batches is high, which hinders the performance of GOLS-I. However, GOLS-I remains competitive, performing better than the small fixed step size, but worse than the medium constant step size. As the batch size increases to $|\mathcal{B}| = 50$ and beyond, the quality of the computed gradient improves sufficiently that GOLS-I trains faster than any of the constant step size methods. The constant step sizes continue to converge linearly towards the optimum, while GOLS-I converges exponentially.

It seems the aggressive training performance of GOLS-I and the medium constant step length suffers from overfitting in this problem. In fact even the small step length seems to overfit within the first 1000 function evaluations for this problem. We speculate
that during training areas associated with generalization of the cost function were either missed or possibly overstepped, indicative of an architecture that is much more flexible than required by the data set. Interestingly, overfitting is not present in the work done by Mahsereci and Hennig [2017] for this problem, neither for their line search, nor for their constant step implementations. Their implementations also seem to train more slowly, requiring a larger number of function evaluations.

Resolved step sizes are plotted to allow comparison between the magnitudes of step sizes obtained by GOLS-I, to the chosen constant step sizes. The large range of step sizes available to GOLS-I is immediately evident. Recall that we do not limit the number of gradient computations per iteration, which allows the line search to vary its magnitude significantly between iterations. Another consequence of this is that the variance in the resolved step size can be used as an indication for the variance in the computed gradient information. As the batch size increases, the range in magnitude of the step sizes begins to narrow, and a slowly increasing step size trend as training progresses begins to emerge. For $|B| = 100$, this increase is slow and still considerably noisy, whereas for $M = 400$ the step size magnitude increases rapidly in a narrow band, as the gradient magnitude drops and the method approaches an optimum. Presumably this occurs to compensate for the decreasing magnitude in the gradient vector, thus requiring a larger step size for an equivalent magnitude in update to the weights. In a ball around this optimum, the line search "bounces" around in high dimensional space. Since the gradient norm is small, the step size magnitudes are large, which corresponds to the flat error region in the corresponding log Training Loss plot for $M = 400$. Here the variance in step size is only due to the inexact line search, since there is no variance in the data for the full batch. This example therefore confirmations that GOLS-I generalizes naturally to smooth loss
6.2 MNIST Dataset

The results for training MNIST with the NetI network architecture using mini-batch sizes of \(|B| \in \{10, 100, 200, 1000\}\), are shown in Figure 7. Again, GOLS-I is hindered by the inconsistent information offered by the smallest mini-batch size \(|B| = 10\). However, as the mini-batch size increases GOLS-I remains competitive. The convergence performance of GOLS-I is better than that of the medium fixed step size from \(|B| = 100\) and larger. In the case of \(|B| = 1000\) training is particularly aggressive in comparison to the constant step sizes. The automatically resolved step sizes of GOLS-I increases with an increase in batch size as well as the training progresses. The superior training performance of GOLS-I in this problem also translates to better test classification errors.

In comparison to the BCWD problem the resolved step sizes of this problem are relatively consistent, having low variance while also showing a slight growing trend during the course of training. Recall, that the initial guess for GOLS-I is \(\alpha_{1,1} = 10^{-8}\). The plots show magnitudes that are quickly within the range of the fixed step sizes. This shows that GOLS-I is capable of recovering an effective step size from the given problem within a few gradient computations.

In this analysis GOLS-I has different convergence characteristics to those of the work of Mahsereci and Hennig [2017]. We cannot comment on the absolute error obtained, due to possible differences in implementation. However, concerning the shape of the convergence rates: The authors’ method tends to progress quickly, then stagnate. Instead, GOLS-I follows a consistent linear convergence rate, which does not stagnate (not counting \(|B| = 10\), where this is not evident) up to the number of function values used for the analyses.

The MNIST results for the NetII architecture are depicted in Figure 8 which exhibit a less competitive view of GOLS-I when compared to the to the NetI results. Firstly, the overall performance of GOLS-I is less competitive; and secondly, the variance in the error
curves is much lower. As expected the architecture significantly effects the training, which is evident in that the three equivalent constant step sizes that had to be chosen one order of magnitude lower than for NetI training. In contrast, GOLS-I remained unchanged. This demonstrates that GOLS-I is able to automatically recover step size within the range of the carefully selected constant step sizes. It is interesting, that in this case GOLS-I tends to decrease the step size slightly as training progresses. We suspect that this behaviour is due to narrow ravines in the cost function, as observed by Goodfellow et al. [2015] (for an additional visual example, refer back to Figure 1), which is due to the NetII architecture. The consequence is that smaller step sizes are being resolved, whereas the medium constant step size could potentially step over these ravines, instead of traversing along them. We would also like to remind the reader, that this would not be a shortcoming of the line search, but of the directions obtained using SGD. This may offer an explanation as to why the convergence of GOLS-I slows down. Although GOLS-I is not as efficient as the medium step size it automatically identified and resolved step size updates in the range of the medium step size without any intervention or tuning required.

This analysis is an example where reproduction of the work of Mahsereci and Hennig [2017] was difficult, as even their chosen step sizes did not perform in our implementation as in theirs. However, a notable positive in our case is that GOLS-I is more stable with $|B| = 10$ than their probabilistic line search, which diverges at this batch size in their implementation.

### 6.3 CIFAR10

The results for CIFAR10 with NetI using mini-batch sizes of $|B| \in \{10, 100, 200, 1000\}$ are shown in Figure 9. It is evident that as the large constant step size improves dramatically as the batch size increases. Similarly, the automatically resolved step sizes of GOLS-I increases with an increase in batch size as well as during the training progress. As before, for the smallest batch size $|B| = 10$ GOLS-I struggles the most to reduce the
training error, where as GOLS-I improves in performance, as the batch size increases. As expected, the medium step size consistently performs well, setting a competitive baseline. For batch sizes $|B| = 100$ and above, GOLS-I outperforms the medium constant step size in training. However, since training only occurs on Batch1 similar to Mahsereci and Hennig [2017], it is difficult to make statements about generality from the test error, irrespective of the step size method used.

Apart from the test error, our results are very similar to those obtained by Mahsereci and Hennig [2017]. Again, this is true for both constant step sizes, and GOLS-I, therefore not being due to the line search. The data combination given by the authors is plausible, since their results are well replicated for NetII. However, we were unable to replicate their test results for CIFAR10 on NetI. Irrespective thereof, the training plots represent the effectiveness of the training methods. In this regard GOLS-I again proves itself to be a capable method, performing well on this example.

For NetI, the performance of GOLS-I on the training data with a mini-batch size of $|B| = 200$ performed the best. As noted before, the resolved step size not only increases progressively during training but also as the batch size gets larger. A trend which is repeated from the MNIST analysis with the same architecture. This might suggest, that the trends of optimal step sizes over training may be linked to network architecture. For this example the step sizes has low variance.

Lastly, the training plots for CIFAR10 with NetII are given in Figure 10. For this example, the training and test errors we obtained were the closest match to those reported.
by Mahsereci and Hennig [2017]. Instead of choosing the medium and large step sizes an order apart we selected the medium constant step size to be 1, and the large constant step size 4. This highlights that the difference between a ”good” and ineffective constant training step size can be small. For training using \(|B| = 100\) and larger, GOLS-I is able to recover competitive step sizes effectively without user intervention, even though the step size sensitivity is high for this architecture and problem.

Comparing the step sizes of this analysis to those of MNIST with the same NetII architecture in Figure 8 it is evident that in both cases GOLS-I overestimates the resolved step sizes for the smallest mini-batch size of \(|B| = 10\). For larger batch sizes the resolved step size trend decreases as training progresses, similar to NetII on MNIST. As expected, it that the architecture might dominate the influence on step size evolution during training.

Interestingly, GOLS-I does not perform as well for \(|B| = 1000\) as for \(|B| = 100\) or \(|B| = 200\). To confirm that this was not an anomaly, we conducted further analyses using \(|B| = 500\) and \(|B| = 2000\), which confirmed these trends. This indicates that the quality of the search directions may not be effective for the given problem, since the precision of a NN-GPP along the direction can only improve with increasing mini-batch size. To substantiate this intuitive speculation, we dedicate an additional numerical investigation on the influence of mini-batch size on the descent direction quality versus its influence on identifying NN-GPP along a descent direction.

7 Uncoupling search direction from directional information quality

In this section we highlight the difference in contribution between the quality of the search direction, and the quality of the information contained along that search direction, in the context of stochastic line searches.

There are undoubtedly two aspects of a line search that utilize information. Firstly,
the search direction, and secondly, estimating a step size along the search direction. In the case of full batch training, both of these utilize maximum available information. However, in mini-batch sampling the contribution of information to the search direction and information along a search direction may be affected differently by mini-batch sampling. We therefore investigate the sensitivity of the descent direction and the sensitivity of locating a sign change along a descent direction with respect to batch size by investigating the performance of GOLS-I.

![Graphs showing training loss for different batch sizes](image)

**Figure 11:** Training loss for the BCWD problem with different batch sizes for the generation of the search direction and evaluation along the search direction. We denote a direction generated with a given batch size $|B|$ as $d_{|B|}^{n}$, and the batch size used during resolution of the NN-GPP in the given direction as $F_{|B|}^{n}$. The quality of both is important: Poor search directions slow down training progress, while high variance in the directional derivative information along a search direction causes large variance in the resolution of the step sizes. The diagonals dominate, where direction quality is matched to that of directional resolution. However, there is a slight bias towards using better search directions, than spending more computational resources on directional resolution.

We conduct an experiment, by which we separate the sampling related to generating the direction, from the sampling that occurs along the search direction. Since we are using SGD, this amounts to evaluating the gradient of the cost function used to decide the descent direction (superscript of $d_{n}$ indicates the batch size) with a different batch size to the gradient computations (superscript of $F_{n}$ indicates the batch size) that is used to evaluate the directional derivative along the descent direction. To this end we
use the BCWD dataset, as its small size allows us to easily use the full dataset during evaluation. We use the same batch sizes as used in the previous section for this dataset, namely: \(|B| = 10, |B| = 50, |B| = 100\) and \(M = 400\). In the investigation each batch size used for the descent direction is paired with each batch size used to resolve the step size along the descent direction, resulting in the full combinatorial range. One can consider the constant step size method with different batch sizes to be SGD with a set constant radius, but varying quality in search direction. Therefore, we include the constant step size results for a given batch size with the corresponding GOLS-I training run with the same search direction batch size. The end result is a 16 plot grid of loss curves relative to function evaluation, shown in Figure 11.

Since the magnitude remains fixed between iterations, constant step sizes are only sensitive to the search direction. Hence, small step sizes are affected less by the variance in direction, as the algorithm never moves particularly far in a given direction and generally moves along the expected direction due to the relatively large number of gradient evaluations within the same local neighbourhood of weight estimates. Conversely, large step sizes perform significantly better when larger batch sizes are used for search directions as opposed to directional derivatives along a search direction. Compare Figures 11(c,d)) to Figures 11(i,m)). It is evident that the medium step size has more uniform improvement when search directions are resolved with higher accuracy, compare again Figures 11(c,d)) to Figures 11(i,m)) but this time in view of the medium step size.

Considering GOLS-I in terms of direction quality, a poorly resolved search direction results in poor training, regardless of the quality to which the NN-GPP along that direction is resolved. This makes intuitive sense, as the line search can make significantly large step updates along the search direction under the immediate accept condition. This is evident when comparing Figures 11(a)-(d) to Figures 11(i), (e), (i) and (m) in view of GOLS-I.

Interestingly, good search directions and inferior resolution along them also do not result in competitive training (see Figures 11(m)-(o)). If one compares this to the use of competitive stochastic directions with \(|B| = 50\) (see Figures 11(e)-(g))) and \(|B| = 100\) (see Figures 11(i)-(k)), full batch directions show severely slower convergence indicating that the additional computational cost to compute better search directions are not capitalized on when the step size is poorly resolved along the descent direction. However, it is expected that improvements should be observed albeit for significantly more gradient computations.

If we consider sample accuracy along a search direction, a low quality in spatial resolution of the NN-GPP is ineffective regardless of the quality of the search direction (see Figure 11(a), (c), (i) and (m)). In this case the variance of 1D the location of the NN-GPP is too high to result in meaningful progress. The other extreme is using very high quality and good spatial resolution to find NN-GPP that are in sub-optimal descent directions (see Figures 11(d), (h), (l) and (p)). This results in a high computational cost in order to resolve solutions along poor descent directions. It is important to note, that apart from the added computational cost of the larger batch size per function evaluation, the line search itself also uses more gradient evaluations per update step, as the higher resolution allows for more accuracy, prompting the algorithm to expend more iterations to find a sign change. It is not uncommon for full batch analyses to use on average 17 gradient evaluations per update step. This is in contrast to the other stochastic examples, where average evaluations per iteration are typically between 2 and 3. In general, computationally sensible strategies should match the quality of the information used to determine an appropriate search direction to that of the quality of information used to determine the solution along a search direction as indicated by the diagonal of Figure 11. The asymmetry in Figure 11 gives a relative indication that a slightly better resolved search direction is better than better resolved direction derivatives along a search direction.
The best training error was obtained using a mini-batch size of $|\mathcal{B}| = 100$, which is one order lower than the full-batch size of $|\mathcal{B}| = 400$. This indicates that mini-batch sampling acts as a regularizer during training. The empirical evidence suggests that to keep the batch sizes the same for both direction estimation and sampling along a search direction. However, slight improvements in performance may be obtained by choosing descent directions with slightly larger sample sizes than the sampling along a search direction. We demonstrate this empirical assertion in Section 7.1.

7.1 Direction Sensitivity to Batch Size

**Figure 12:** (a-h) Mean angle, $\theta_{dg}$, denoting the angle between the fixed search direction $d_n^M$ and $g(x)^{\mathcal{B}}$. The full batch is used for the descent direction, setting a consistent optimization path, while $g(x)^{\mathcal{B}}$ is evaluated using $|\mathcal{B}|$, the angle between the two indicates potential deviation from the "true" path. (i-p) The log norm of descent directions $d_n^M$ plotted against function evaluations. The maximum angles remain relatively consistent, but the minimum angle decreases with increasing batch size, showing closer approximation to the "true" descent direction. In all cases the algorithm converges, since the full batch direction was used for optimization. This causes the norm of the direction to approach zero.

In this section we conduct a short numerical investigation to identify the variance evolution of mini-batch computed descent directions over training. As before, we use the BCWD problem for this analysis. We sample the true descent direction (with $M = 400$) at the solution of every iteration, $x_n$, and sample an additional 20 other descent directions...
using various mini-batch sample sizes $|B| \in \{10, 40, 100, 200, 300, 350, 399, 400\}$, where 400 indicates the full batch. We then calculate the angle between the full batch "true" descent direction and the estimated mini-batched sampled descent directions by noting the average angle. Hence, the mean angle between the true descent direction and estimated mini-batched sampled descent directions are plotted as the optimizer updates using the full batch "true" descent with full batch directional derivatives along the descent direction in Figure 12.

It is evident that between batches there are significant changes in mean angle. While later iterations exhibit larger variance for larger batch sizes, the mean angle decreases as the batch size increases. The analysis shows that there is a significant "ramp-up" period in the first roughly 1000 function evaluations. During this period the mean angle increases from a minimum to a maximum value, after which it seems to settle around a constant mean until convergence, where the mean changes again.

Important features include the starting point, and the behavior towards the end of training. As the sample size increases, the mean angle at the beginning of the analysis decreases. This indicates consistency in information contained in the directions. Considering, Figure 11 it is evident that only the direction sampled using the smallest mini-batch $|B| = 10$ failed to converge in general using GOLS-I. This may indicate that for this problem an initial directional deviation of around 50 degrees in the descent direction is too severe, leading to a different solution (recall Figure 11(a-d)). Deviations of around 20 degrees seem to generate similar solutions to the full batch true descent directions for this problem, as these analyses converged (recall Figure 6(b)). At later stages of convergence larger variance in the mean angles still leads to convergence, with many mean angles being around 80 degrees. This means that at later stages in training, individual samples contribute more towards the direction, though they contribute less towards the error. Since the BCWD dataset is a classification problem, an analogy might be that each sample has a different contribution towards which way the decision boundary needs to move to improve the sample’s error. In the beginning, most data-samples contain similar information in terms of where the decision boundary needs to move to reduce the classification error. However, as more of the common information in data-samples is incorporated into the model, more individual differences between the data points become highlighted. We observe this particularly clearly when $|B| = 399$. This means that overall the error decreases, but the differences between the directions increases for a constant batch size.

8 Conclusion

For discontinuous stochastic optimization objective functions, we proposed Inexact Gradient-Only Line Search (GOLS-I) as a computationally efficient strategy to automatically resolve learning rates. Instead of minimizing along a descent direction or finding critical points along descent directions we locate Non-Negative Associated Gradient Projection Points (NN-GPP). Along a 1-D descent direction NN-GPP are indicated by sign changes from negative (indicative of descent) to positive (indicative ascent) in the directional derivative. Hence, NN-GPP incorporates second order information indicative of a minimum.

We demonstrate on three classical machine learning problems (Breast Cancer Wisconsin Diagnostic, MNIST with two neural net architectures and CIFAR10 with two neural net architectures) that learning rates can be efficiently resolved for SGD using GOLS-I.

Our method has been demonstrated to be competitive in training without requiring any manual tuning, which reduces active human hours required to successfully train a neural net. GOLS-I allows for dynamic step sizes that can vary over 15 orders of magnitude, i.e. from $10^{-8}$ to $10^7$. Lastly, GOLS-I allows for an intuitive line search implementation, which shows a great deal of potential for further development and in-
Integration into other traditional mathematical programming methods. Towards this aim, we conducted a small empirical investigation regarding the information required to resolve descent directions versus directional derivatives along a descent direction for only the Breast Cancer Wisconsin Diagnostic Dataset. For this problem it was found that keeping the batch sizes for evaluation of search directions in SGD the same results in a reliable initial selection strategy as long as the search direction is sufficiently resolved. For SGD, there seems to be some potential computational benefit in using slightly less gradient computations to resolve the directional derivatives along a descent direction. However, to obtain conclusive results may require more representative datasets as well as additional optimizers where we use GOLS-I to resolve the step sizes dynamically.

This initial study will hopefully stimulate the possibility of successfully using line searches in stochastic neural network optimization, which may also present alternative opportunities to incorporate second order information with strategies like Quasi-Newton and conjugate gradient methods [Arora, 2011, Le et al., 2011].

Acknowledgements

This work was supported by the Centre for Asset and Integrity Management (C-AIM), Department of Mechanical and Aeronautical Engineering, University of Pretoria, Pretoria, South Africa. NVIDIA for sponsoring the Titan X Pascal GPU used in this study.
Appendix A.

Algorithm 2: GOLS-I: Inexact Gradient-Only Line Search

Input: $F_n(\alpha), d_n, \alpha_{n,1}$

Output: $\alpha_{n,I_n}, k$

1. Define constants: $\alpha_{\min} = 10^{-8}$, flag = 1, $k = 0$, $\eta = 2$, $c_2 = 0.9$
2. $\alpha_{\max} = \min\left(\frac{1}{||d_n||_2}, 10^7\right)$
3. Evaluate $F_n'(0)$, increment $k$
4. if $(\alpha_{n,1}) > \alpha_{\max}$ then
   5. $\alpha_{n,1} = \alpha_{\max}$
5. if $(\alpha_{n,1}) < \alpha_{\min}$ then
   6. $\alpha_{n,1} = \alpha_{\min}$
7. Evaluate $F_n'(\alpha_{n,1})$, increment $k$
8. Define $tol_{dd} = |c_2F_n'(0)|$
9. if $F_n'(\alpha_{n,1}) > 0$ and $(\alpha_{n,1}) < \alpha_{\max}$ then
   10. flag = 1, decrease step size
11. if $F_n'(\alpha_{n,1}) < 0$ and $(\alpha_{n,1}) > \alpha_{\min}$ then
   12. flag = 2, increase step size
13. if $F_n'(\alpha_{n,1}) > 0$ and $F_n'(\alpha_{n,1}) < tol_{dd}$ then
   14. flag = 0, immediate accept condition
15. while flag > 0 do
   16. if flag = 2 then
   17. $\alpha_{n,i+1} = \alpha_{n,i} \cdot \eta$
   18. Evaluate $F_n'(\alpha_{n,i+1})$, increment $k$
   19. if $F_n'(\alpha_{n,i+1}) \geq 0$ then
   20. flag = 0
   21. if $\alpha_{n,i+1} > \frac{\alpha_{\max}}{\eta}$ then
   22. flag = 0
   23. if flag = 1 then
   24. $\alpha_{n,i+1} = \frac{\alpha_{n,i}}{\eta}$
   25. Evaluate $F_n'(\alpha_{n,i+1})$, increment $k$
   26. if $F_n'(\alpha_{n,i+1}) < 0$ then
   27. flag = 0
   28. if $\alpha_{n,i+1} < \alpha_{\min} \cdot \eta$ then
   29. flag = 0
30. $\alpha_{n,I_n} = \alpha_{n,i+1}$
31. $\alpha_{n,I_n} = \alpha_{n,i+1}$
References

Lyapunov Aleksandr. M. The general problem of the stability of motion. *International Journal of Control, 55*(3):531–534, 1992. doi: 10.1080/00207179208934253.

Jasbir Arora. *Introduction to Optimum Design, Third Edition.* Academic Press Inc, 2011. ISBN 0123813751.

James Bergstra and Yoshua Bengio. Random Search for Hyper-Parameter Optimization. *Journal of Machine Learning Research, 13*(Feb):281–305, 2012. ISSN 1532-4435. doi: 10.1162/153244303322533223. URL [http://www.jmlr.org/papers/v13/bergstra12a.html](http://www.jmlr.org/papers/v13/bergstra12a.html).

Stephen Boyd, Lin Xiao, and Almir Mutapcic. Subgradient methods. *lecture notes of EE392o, Stanford . . ., 1*(May):1–21, 2003.

R. A. Fisher. The use of Multiple Measurements in Taxonomic Problems. *Annals of Eugenics, 7*(2):179–188, sep 1936. ISSN 20501420. doi: 10.1111/j.1469-1809.1936.tb02137.x.

Xavier Glorot and Yoshua Bengio. Understanding the difficulty of training deep feedforward neural networks. In *Proceedings of Machine Learning Research*, pages 1–8, 2010. ISBN 0-7803-1421-2. doi: 10.1109/IJCNN.1993.716981.

Ian J. Goodfellow, Oriol Vinyals, and Andrew M. Saxe. Qualitatively Characterizing Neural Network Optimization Problems. *ICLR*, pages 1–11, 2015. URL [http://arxiv.org/abs/1412.6544](http://arxiv.org/abs/1412.6544).

Alex Krizhevsky and Geoffrey E. Hinton. Learning Multiple Layers of Features from Tiny Images. 2009. URL [https://www.cs.toronto.edu/~kriz/cifar.html](https://www.cs.toronto.edu/~kriz/cifar.html).

Quoc V Le, Adam Coates, Bobby Prochnow, and Andrew Y Ng. On Optimization Methods for Deep Learning. *Proceedings of The 28th International Conference on Machine Learning (ICML)*, pages 265–272, 2011. ISBN 9781450306195. doi: 10.1.1.220.8705.

Y Lecun, L Bottou, Y Bengio, and P Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE, 86*(11):2278–2324, nov 1998. ISSN 0018-9219. doi: 10.1109/5.726791.

Ilya Loshchilov and Frank Hutter. SGDR: Stochastic Gradient Descent with Warm Restarts. pages 1–16, 2016. ISSN 15826163. doi: 10.1002/fut. URL [http://arxiv.org/abs/1608.03983](http://arxiv.org/abs/1608.03983).

Maren Mahsereci and Philipp Hennig. Probabilistic Line Searches for Stochastic Optimization. pages 1–12, 2017. ISSN 10495258. doi: 10.1016/j.physa.2015.02.029. URL [http://arxiv.org/abs/1703.10034](http://arxiv.org/abs/1703.10034).

Francesco Orabona and Tatiana Tommasi. Training Deep Networks without Learning Rates Through Coin Betting. pages 1–14, 2017. ISSN 10495258. URL [http://arxiv.org/abs/1705.07795](http://arxiv.org/abs/1705.07795).

Herbert Robbins and Sutton Monro. A Stochastic Approximation Method. *The Annals of Mathematical Statistics, 22*(3):400–407, sep 1951. ISSN 0003-4851. doi: 10.1214/aoms/1177729586. URL [http://projecteuclid.org/euclid.aoms/1177729586](http://projecteuclid.org/euclid.aoms/1177729586).

Nicol N Schraudolph, Jin Yu, and Simon Günter. A Stochastic Quasi-Newton Method for Online Convex Optimization. *International Conference on Artificial Intelligence and Statistics*, pages 436—-443, 2007. ISSN 15324435. doi: 10.1137/140954362. URL [http://eprints.pascal-network.org/archive/00003992/](http://eprints.pascal-network.org/archive/00003992/).
N.N. Schraudolph. Local gain adaptation in stochastic gradient descent. *9th International Conference on Artificial Neural Networks: ICANN ’99*, 1999:569–574, 1999. ISSN 0537-9989. doi: 10.1049/cp:19991170. URL [http://digital-library.theiet.org/content/conferences/10.1049/cp{._19991170}](http://digital-library.theiet.org/content/conferences/10.1049/cp{._19991170})

Nn Schraudolph and T Graepel. Combining conjugate direction methods with stochastic approximation of gradients. *Proceedings of the Ninth International Workshop on Artificial Intelligence and Statistics, AISTATS 2003*, pages 2–7, 2003. URL [http://www.schraudolph.org/pubs/SchGra03.pdf](http://www.schraudolph.org/pubs/SchGra03.pdf).

Andrew Senior, Georg Heigold, Marc’Aurelio Ranzato, and Ke Yang. An empirical study of learning rates in deep neural networks for speech recognition. *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing, ICASSP 2013*, pages 6724–6728, 2013. ISSN 1520-6149. doi: 10.1109/ICASSP.2013.6638963. URL [http://ieeexplore.ieee.org/xpl/articleDetails.jsp?tp={(}&arnumber=6638963](http://ieeexplore.ieee.org/xpl/articleDetails.jsp?tp={(}&arnumber=6638963)

Leslie N. Smith. Cyclical Learning Rates for Training Neural Networks. (April), 2015. doi: 10.1109/WACV.2017.58. URL [http://arxiv.org/abs/1506.01186](http://arxiv.org/abs/1506.01186).

Jan A Snyman and Daniel N Wilke. *Practical Mathematical Optimization*, volume 133 of *Springer Optimization and Its Applications*. Springer International Publishing, Cham, 2018. ISBN 978-3-319-77585-2. doi: 10.1007/978-3-319-77586-9. URL [http://link.springer.com/10.1007/978-3-319-77586-9](http://link.springer.com/10.1007/978-3-319-77586-9).

W.N Street, W.H. Wolberg, and O.L. Mangasarian. Nuclear Feature Extraction For Breast Tumor Diagnosis. 1993.

Fei Tong and Xila Liu. Samples Selection for Artificial Neural Network Training in Preliminary Structural Design. *Tsinghua Science & Technology*, 10(2):233–239, apr 2005. ISSN 1007-0214. doi: 10.1016/S1007-0214(05)70060-2. URL [https://www.sciencedirect.com/science/article/pii/S1007021405000602](https://www.sciencedirect.com/science/article/pii/S1007021405000602).

Paul John Werbos. *The Roots of Backpropagation: From Ordered Derivatives to Neural Networks and Political Forecasting*. Wiley-Interscience, New York, NY, USA, 1994. ISBN 0-471-59897-6.

Daniel Nicolas Wilke, Schalk Kok, Johannes Arnoldus Snyman, and Albert A. Groenwold. Gradient-only approaches to avoid spurious local minima in unconstrained optimization. *Optimization and Engineering*, 14(2):275–304, June 2013. ISSN 1389-4420. doi: 10.1007/s11081-011-9178-7. URL [http://link.springer.com/10.1007/s11081-011-9178-7](http://link.springer.com/10.1007/s11081-011-9178-7).

D.N. Wilke. Structural shape optimization using Shor’s r-algorithm. In *Third International Conference on Engineering Optimization*, 2012. ISBN 978-85-76503-43-9.

D Randall Wilson and Tony R Martinez. The general inefficiency of batch training for gradient descent learning. *Neural Networks*, 16(10):1429–1451, 2003. ISSN 08936080. doi: 10.1016/S0893-6080(03)00138-2.

Xiaoxia Wu, Rachel Ward, and Léon Bottou. WNGrad: Learn the Learning Rate in Gradient Descent. pages 1–16, 2018. URL [http://arxiv.org/abs/1803.02865](http://arxiv.org/abs/1803.02865).