Second-Order Trust-Region Optimization for Data-Limited Inference

Aditya Ranganath[1], Omar DeGuchy[2], Mukesh Singhal[1], and Roummel F. Marcia[2]
[1]*Electrical Engineering and Computer Science, University of California, Merced, Merced, CA USA*
[2]*Applied Mathematics, University of California, Merced, Merced, CA USA*

***Abstract*—Neural networks generally require large amounts of data to adequately model the domain space. In situations where the data are limited, the predictions from these models, which are typically obtained from stochastic gradient descent (SGD) minimization algorithms, can be poor. In these cases, the use of more sophisticated optimization approaches becomes even more crucial to increase the impact of each training iteration. In this paper, we propose an optimization algorithm that uses second-derivative information that exploits curvature information for avoiding saddle points, which can result in limitations on the learning process. In particular, we utilize a Hessian-free approach where we do not explicitly store the second-derivative matrix; rather, we apply a conjugate gradient method and require the Hessian matrix only to compute matrix-vector products. Our approach is based on trust-region methods, which do not require the Hessian to be positive definite and which differentiates our approach from existing Hessian-free methods. We present numerical experiments which demonstrate the improvement in classification accuracy using our proposed approach over a standard SGD approach.**

***Index Terms*—Second-order methods, Hessian-free, conjugate gradient methods, trust-region methods**

I. Introduction

Deep learning problems often involve the minimization of an objective function given by

$$\underset{w\in\Re^d}{\text{minimize}}\ f(w) \equiv \frac{1}{n}\sum_{j=1}^{n} f_j(w), \tag{1}$$

where f_j is a function that depends on the j^{th} observation in a training set $\{(x_j, y_j)\}_{j=1}^{n}$. These objective functions are generally large-scale (the dimension of w, d, and the number of data points, n, are typically in the order of millions), non-linear (the function f often involves nonlinear activation functions), and non-convex (f is a composition of functions that can result in non-convexity [1]).

The recent success in the application of neural networks to a variety of fields can be mostly attributed to two driving factors: improvements in network architecture and the optimization techniques used during the learning process [2]–[4]. Since the inception of the back-propagation algorithm made the optimization of the network parameters viable, methods using first-derivative information have dominated the field (see e.g., [5]–[11]). With an increased access to powerful computational resources, there is greater potential for the field to shift toward considering more sophisticated algorithms. This is particularly true for data-limited inference, where the availability of data is limited, i.e., n in (1) is relatively small. In the case of an image classifier, this translates to a limited number of training images for classification.

*This work is funded in part by National Science Foundation Grant IIS 1741490.

The novelty of this paper is as follows: **we exploit second-derivative information to improve the predictive capabilities of artificial neural networks for data-limited inference**. Our method combines the efficient computation of a true Hessian-vector product in a trust-region setting, thus allowing us to solve the trust-region subproblem using a conjugate based method.

The paper is organized as follows. In Sec. 2, we describe existing and commonly used algorithms and then discuss some of the recent techniques which approximate second-order information. In Sec. 3, we describe the novelty of our proposed approach, which is based on trust-region methods, which are alternatives to the more commonly-used line-search methods in optimization. We describe our numerical experiments in Sec. 4 and the main results in Sec. 5. Finally, we summarize our paper with concluding remarks in Sec. 6.

II. Related Methods

Stochastic Gradient Descent (SGD). First-derivative algorithms have emerged as the standard optimization techniques used for training deep neural networks. In particular, methods based on stochastic gradient descent (SGD) are preferred for their low computational cost and ease of implementation [5], [12]. This method differs from a classic gradient descent approach in that the gradient is computed based on a sample of the dataset. In the context of deep learning, the gradient is computed based on a sample batch. Specifically, at iteration k in SGD, a sample batch $S_k \subseteq \{1, 2, \ldots, n\}$ is randomly chosen and the current iterate w_k is updated using

$$w_{k+1} = w_k - \alpha_k \frac{1}{|S_k|}\sum_{j\in S_k} \nabla f_j(w_k),$$

where α_k known as the *learning rate*.

Quasi-Newton Methods. Quasi-Newton methods have also been explored as a possible alternative to gradient descent based algorithms for training neural networks [13]–[16]. These methods use previous computed gradients to approximate the second derivative to improve the search direction at each iteration.

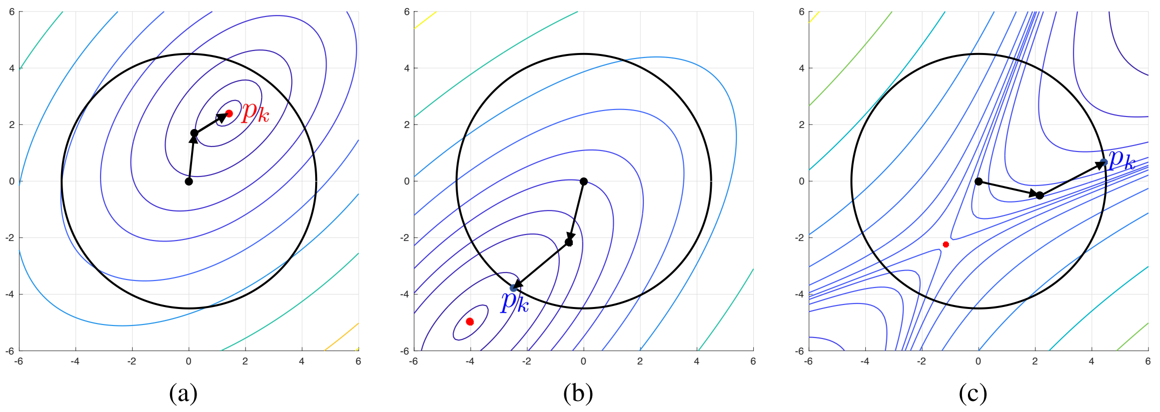

Fig. 1. Illustration of the CG-Steihaug approach in two dimensions. (a) When $\mathcal{Q}_k(p)$ is convex and its unconstrained minimizer lies within the trust-region radius, then the CG iterates will converge to the unconstrained minimizer. (b) When $\mathcal{Q}_k(p)$ is convex but its unconstrained minimizer is outside the trust region, then the minimizer p_k is defined where the CG iterate crosses the boundary. (c) When $\mathcal{Q}_k(p)$ is not convex, i.e., H_k is not positive definite, then the CG-Steihaug method terminates when a direction of curvature is detected and the minimizer p_k is defined where $\mathcal{Q}_k(p)$ is minimized along the last computed CG iterate.

Hessian-Free Methods. Like quasi-Newton methods, Hessian-free methods look to improve on gradient-descent methods by using higher-order information. To minimize the cost of storing second-derivative (Hessian) matrices (which can be potentially too large to store in memory), Hessian-free methods only require them for matrix-vector multiplication. The approach by Martens [17] uses the finite difference approximation of the matrix vector product:

$$Hd = \lim_{\epsilon \to 0} \frac{\nabla f(x + \epsilon d) - \nabla f(x)}{\epsilon}, \tag{2}$$

where the operation is used in a conjugate gradient (CG) setting in order to provide the descent direction to the next iterate. Pearlmutter [18] offers an alternative approach which computes the actual Hessian-vector product as

$$Hd = \frac{\partial}{\partial \epsilon} \nabla f(x + \epsilon d) \bigg|_{\epsilon = 0}. \tag{3}$$

For details on implementation, see [19]–[22]. A related approach utilizes Gauss-Newton approximations [23]. For example, the approach in [24] extends the approximation for use with a cross-entropy loss and a framework similar to [18].

III. Proposed Approach

In the methods described above, the Hessian H is often required to be positive definite to guarantee that the computed search direction is a descent direction. For non-convex problems, this requirement is not always satisfied. In [17], the eigenvalues of H are shifted by adding λI to H, where $\lambda > 0$ is obtained heuristically. Here, we propose using a trust-region approach that does not require modifying the eigenvalues of H. As they are stated, the matrix-vector products in (2) and (3) do not provide any guarantees that the matrix H is positive definite. One commonly used technique to guarantee to avoid the problems associated with an indefinite matrix is to shift or "dampen" the eigenvalues of H as $B = H + \lambda I$, where $\lambda \geq 0$. The Hessian-vector product is now expressed as $Bd = Hd + \lambda d$, where Hd is evaluated using the previously described techniques.

The proposed method presents a novel optimization routine designed to minimize (1). Unlike the previous methods described, the goal is computing fast Hessian-vector products within a trust-region setting. This allows us to approximately solve the trust-region subproblem using CG while allowing for negative curvature. By incorporating second-derivative information, we improve the impact of each iteration and avoid certain local minima and saddle points. The increase in the quality of the optimization routine will increase the value of each data point and allow us to reach better optima with fewer training instances. In the following subsections we describe the proposed approach in more detail.

Fast Exact Hessian-Vector Products. As stated in the previous section, computing the second-derivative or Hessian can be computationally intensive. Furthermore, in the context of neural networks, storing the Hessian can be infeasible. Using exact second derivative information allows us to create an accurate localized model of the true objective function which is used in the trust-region setting described in the next section. At the same time we require this information to be available for use in the CG method in Section 3.3. In both cases, a Hessian-vector multiplication is required, and so we chose to use Pearlmutter's algorithm, commonly referred to as Rop [18]. While we are not the first to use Rop in a CG setting, our approach is novel in that we no longer require dampening of the Hessian to guarantee positive definiteness.

The motivation for avoiding damped Hessian approximations comes in two parts. The first is that it requires the choice of another hyperparameter, λ. The choice of λ can greatly affect the convergence of the optimization routine and should be chosen for each update of the Hessian vector product. The second motivation is that the perturbation to the true Hessian imposed by λ results in an approximation of the second derivative and thus less accurate curvature information. *The proposed algorithm relaxes the requirement of the Hessian*

Algorithm 1 CG-Steihaug

```
Given: Tolerance $\epsilon_k > 0$
Set: $p_0 = 0$, $r_0 = g_k, d_0 = -r_0 = -g_k$
if $\|r_0\|_2 < \epsilon_k$ then
    return $p_k = z_0 = 0$
end if
for $j \in$ 0,1,2,3,4.. do
    if $d_j^\top H_k d_j \leq 0$ then
        Find $\tau$ such that $p_k = z_j + \tau d_j$ minimizes
                $\mathcal{Q}_k(p)$ in (4) with $\|p_k\|_2 = \Delta_k$
        return $p_k$
    end if
    Set $\alpha_j$ = $r_j^\top r_j/\ d_j^\top H_k d_j$
    Set $z_{j+1}$ = $z_j + \alpha_j d_j$
    if $\|z_{j+1}\| \geq \Delta_k$ then
        Find $\tau \geq 0$ such that $p_k = z_j + \tau d_j$ satisfies
                $\|p_k\|_2 = \Delta_k$
        return $p_k$
    end if
    Set $r_{j+1} = r_j + \alpha_j H_k d_j$
    if $\|r_{j+1}\|_2 < \epsilon_k$ then
        return $p_k = z_{j+1}$
    end if
    Set $\beta_{j+1} = r_{j+1}^\top r_{j+1}/r_j^\top r_j$
    Set $d_{j+1} = -r_{j+1} + \beta_{j+1} d_j$
end for
```

to be positive-definite and compensates for the possibility of negative curvature by using a trust-region setting.

Trust-Region Methods. Trust-region methods [25] are alternative approaches to line-search methods for solving optimization problems. While line-search methods first compute a search direction and then determine a step length along that direction at each iteration, trust-region methods determine a quadratic model to the true objective function and a corresponding region over which the quadratic model can be trusted to be accurate. Specifically, trust-region methods solve a sequence of quadratic subproblems with a single constraint of the following form:

$$\begin{aligned} p_k \quad = \quad & \underset{p \in \Re^m}{\arg\min} \quad \mathcal{Q}_k(p) \equiv \nabla f(w_k)^\top p + \frac{1}{2} p^\top \nabla^2 f(w_k) p \\ & \text{subject to} \quad \|p\|_2 \leq \Delta_k \end{aligned} \tag{4}$$

where Δ_k is a scalar parameter referred to as the *trust-region radius*. The trust-region subproblem solution p_k is used to compute the next iterate given by $w_{k+1} = w_k + p_k$, provided p_k satisfies a certain property discussed below.

Conjugate Gradient (CG)-Steihaug Approach. An important component of the algorithm facilitates solving the trust-region subproblem in (4) to provide the directional step p_k to the next iteration. In a similar approach to [17], we use a conjugate gradient method with the fast exact Hessian-vector product in (3). Our algorithm uses a modified CG method known as the CG-Steihaug approach [26], or as

Algorithm 2 Proposed Second-Order Trust-Region Method

```
Given: For some $\Delta_{\max} > 0, \Delta_0 \in (0, \Delta_{\max}), \eta \in [0, \frac{1}{4})$,
and $\epsilon > 0$
while $\|\nabla f(w_k)\|_2 > \epsilon$ do
    Obtain $p_k$ from CG-Steihaug in [26]
    Perform line search using Wolfe conditions
    Evaluate the ratio given by
        $\rho_k = (f(w_k) - f(w_{k+1}))/(\mathcal{Q}_k(0) - \mathcal{Q}_k(p_k))$
    if $\rho_k < \frac{1}{4}$ then
        $\Delta_{k+1} = \frac{1}{4}\Delta_k$
    else
        if $\rho_k > \frac{3}{4}$ and $\|p_k\|_2 = \Delta_k$ then
            $\Delta_{k+1} = \min(2\Delta_k, \Delta_{\max})$
        else
            $\Delta_{k+1} = \Delta_k$
        end if
    end if
    if $\rho > \eta$: then
        $w_{k+1} = w_k + p_k$
    else
        $w_{k+1} = w_k$
    end if
end while
```

the Steihaug-Toint truncated conjugate gradient method [25]. which we outline in Algorithm 1. If the minimizer lies within the trust-region radius, then the CG iterates converge to the unconstrained minimizer (see Fig. 1(a)). If the minimizer is outside of the trust region, then the CG iteration terminates where the iterate crosses the boundary 1(b)). Finally, if the Hessian is not positive definite, then CG-Steihaug terminates the algorithm and the minimizer is detected at the boundary of the trust region in the direction of the last computed CG iterate (see Fig. 1(c)). We re-iterate that the proposed method does not require the Hessian to be positive definite. Therefore, we do not have to use a dampening scalar λ, and consequently, the method allows directions of negative curvature to be detected to avoid saddle points. We describe our proposed approach in Algorithm 2. We note that the final iterate p_k in Algorithm 2 satisfies the following decrease in the quadratic model:

$$\mathcal{Q}_k(0) - \mathcal{Q}_k(p_k) \geq c_1 \|g_k\|_2 \min\left(\Delta_k, \frac{\|g_k\|_2}{\|H_k\|_2}\right), \tag{5}$$

where c_1 is a constant with $c_1 \in (0, 1]$. Also, we note that the direction p_k satisfies the trust-region constraint, i.e., $\|p_k\|_2 \leq \Delta_k$, which will be used for convergence results.

Summary of Proposed Approach. In summary, the proposed approach has three major components: (i) a trust-region method (outlined in Algorithm 2, which allows for indefinite Hessians that avoid saddle points; (ii) a fast and exact Hessian-vector product (3) which efficiently provides true second-derivative information at the current iterate; and (iii) the CG-Steihaug approach, which solves the trust-region subproblem without storing the Hessian matrix. To guarantee that the search direction provided by the CG-Steihaug method sufficiently decreases the value of the quadratic subproblem, we implement a Wolfe line search in the direction of p_k [26]. Furthermore, we evaluate the accuracy of the quadratic model of the objective function within the trust region using the ratio ρ_k in Algorithm 2 to determine whether the update is acceptable or not. The result is an algorithm that considers second derivative information and allows for the possibility of negative curvature. As such, the impact of each iteration is more valuable when compared to those of first-order methods.

Convergence. We conclude with the following convergence guarantee for our proposed method. We first define the level set $S = \{w : f(w) \leq f(w_0)\}$, where w_0 is the initial point, and an open neighborhood of S by $S(R_0) = \{w : \|w - y\| < R_0 \text{ for some } y \in S\}$, where R_0 is a positive constant. Then we have the following result.

Theorem 1. Let $\eta = 0$ in Algorithm 2. Suppose that $\|H_k\| \leq \beta$ for some constant $\beta > 0$, that $f(w)$ is bounded below on the level set L and Lipschitz continuously differentiable in the neighborhood $L(R_0)$ for some $R_0 > 0$, and that all feasible solutions p_k of (4) satisfy (5). Then we have

$$\liminf_{k \to \infty} \|g_k\|_2 = 0.$$

The proof follows directly from Theorem 4.5 in [26].

IV. Limited-Data Experimental Setup

To validate the effectiveness of the proposed algorithm, we implemented Algorithm 1 with the intent to train a neural network to perform a well established classification task. However, we imposed a limitation on the amount of data available for training to simulate data-starvation. Here, we describe our experimental setup.

Neural Network Architecture. In each of these experiments, we used a type of architecture known as a Multi-Layer Perceptron (MLP), which are a class of feed-forward artificial neural networks with the ability to separate data with a non-linear decision boundary [27]. The MLP used in our experiments was implemented using the deep learning package Theano and consisted of three layers of nodes/neurons. The input layer consisted of the vectorized version of the input image (784 nodes), where each node corresponds to each pixel in the sample image. The hidden layer contained 500 neurons and was followed by the non-linear hyperbolic tangent function (tanh) used as a non-linear activation on the output of the layer. Finally, the output layer consisted of 10 neurons corresponding to the 10 classes present in the the dataset. This layer was also followed by the same activation function as the previous layer. The softmax activation function was applied to the output of the neural network in order to provide a probability distribution of the possible classes. The probability distribution was compared to the true one-hot encoding of the target class using a cross entropy loss function. The architecture was kept simple to demonstrate the effectiveness of the optimization method on the intended task rather than the complexity of the MLP.

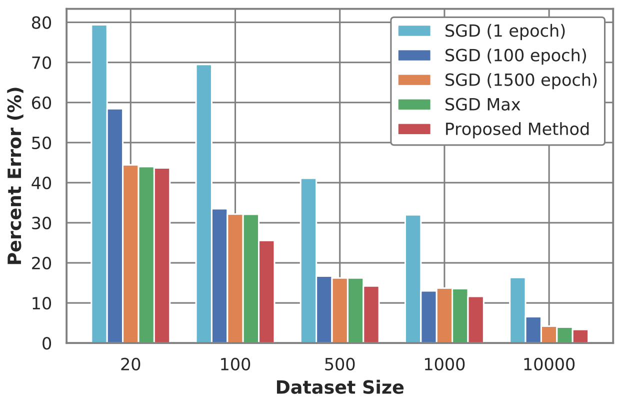

Fig. 2. A comparison of our method to Stochastic Gradient Descent (SGD) with limited datasets. Here we report the accuracy error for datasets of 20, 100, 500, 1000 and 10000 images. Results are also shown for various numbers of epochs. The proposed approach was only trained for 1 epoch while SGD was allowed to run for 1, 100, and 1500 epochs and SGD Max, which refers to the number of epochs of SGD allowed to run within the time our proposed method runs 1 epoch.

Dataset. The dataset used for these experiments is known as the MNIST dataset [28], which consists of rasterized 28×28 pixel images of handwritten digits from 0-9 (10 classes). The collection of images is partitioned in a training set of 60,000 images and a test set of 10,000 images.

Hardware. Experiments were carried out on GPUs housed in a high-performance cluster. The cluster consists of 95 computer nodes with a total of 2116 cores and 2301 Mhz processing power. The GPUs include 4 NVIDIA Tesla K20 graphics cards and 2 Nvidia p100s with a total operating capacity of 62 TFLOPS.

Testing Procedure. The goal of our experiments is to show that the proposed approach can improve on SGD when training a neural network with **limited data**. The 70,000 images in the MNIST dataset were partitioned in the following manner: 50,000 images in the training set, 10,000 images in the validation set and 10,000 images in the testing set. From the training set, we randomly sampled subsets with sizes of 20, 100, 500, 1000 and 10,000 images. When implementing our algorithm, the entire subset is used in a single batch over a single epoch. The algorithm terminates when the norm of the gradient of the objective function reaches a sufficient tolerance (in our case, this was set to 10^{-5}). When using SGD, we approached training using the standard practice of using mini-batches. We chose a mini-batch size of 20 images for all data subsets

V. Main Results

The results of our experiments are presented in Fig. 2 and Table 1, which presents the average of 10 runs each with each run using a different initial point. When compared to SGD for various training times and various dataset sizes, the proposed method improves on the accuracy of the classification task. In particular, after a single pass (1 epoch) through the available

Dataset size	Percent Error				
	SGD 1 epoch	**SGD 100 epochs**	**SGD 1500 epochs**	**SGD Max**	**Proposed Method**
20	79.39	58.45	44.45	44.01	**43.68**
100	69.49	33.50	32.17	32.12	**25.58**
500	41.10	16.70	16.23	16.23	**14.23**
1000	31.98	13.01	13.71	13.57	**11.64**
10000	16.35	6.57	4.21	3.97	**3.38**

TABLE I

ERROR TABLE CORRESPONDING TO THE TESTING ERROR/LOSS OF THE NEURAL NETWORK FOR OUR PROPOSED METHOD IN COMPARISON TO SGD OVER VARIOUS EPOCHS AND FOR DIFFERENT DATASET SIZES. FOR OUR PROPOSED METHOD, THE DATASET IS FED AS A BATCH TO THE NETWORK. FOR SGD, THE DATA ARE FED IN MINI-BATCHES OF 20 IMAGES. SGD MAX CORRESPONDS TO SGD TRAINED OVER THE SAME GPU RUN TIME AS OUR PROPOSED ALGORITHM.

data the proposed method is almost twice as accurate as SGD. This confirms the notion that the iterations of the proposed method have a greater impact than that of the first-order method. As we continued to allow the network to train we can see that after almost 1500 epochs of SGD (which may be considered overtraining) we do not see a great improvement in the level of accuracy. **In fact, we even allowed the SGD algorithm to run in the same GPU time as our method, we still improve on the performance of SGD.** The results in Table 1 show that the improvement on SGD in the case of a 20 image dataset is less than 2%. In this case, 20 images might approach the lower limit on the minimum size of the data set you need in order for the network to learn. We can see that as we increase the number of images in the dataset, SGD still underperforms in comparison to our proposed method.

VI. CONCLUDING REMARKS

In this paper we proposed a novel algorithm for training neural networks **using second-order information for data-limited inference**. In particular, our algorithm improves on first-order methods by allowing the use of curvature information to improve the quality of each iteration in the optimization method. This is accomplished by using a fast exact Hessian operation, which allows us to efficiently compute matrix-vector multiplication with the exact second-derivative matrix. Unlike previous algorithms that have used this type of algorithm in a conjugate gradient setting, by using the CG-Steihaug approach in a trust-region setting it allows us to relax the requirement that the Hessian be positive definite. This allows the algorithm to detect negative curvature information and avoid saddle points. We provided numerical results in a standard implementation of an MLP classification problem where the training dataset was limited. In all cases, the proposed method improves upon a standard implementation of Stochastic Gradient Descent trained over various epochs.

REFERENCES

[1] S. Boyd and L. Vandenberghe, *Convex optimization*. Cambridge University Press, 2004.

[2] C. Angermueller, T. Pärnamaa, L. Parts, and O. Stegle, "Deep learning for computational biology," *Molecular systems biology*, vol. 12, no. 7, p. 878, 2016.

[3] D. Cireşan, A. Giusti, L. Gambardella, and J. Schmidhuber, "Mitosis detection in breast cancer histology images with deep neural networks," in *Intl. Conf. on Medical Image Computing and Computer-Assisted Intervention*. Springer, 2013, pp. 411–418.

[4] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *Nature*, vol. 521, no. 7553, p. 436, 2015.

[5] L. Bottou, "Large-scale machine learning with stochastic gradient descent," in *Proceedings of COMPSTAT'2010*. Springer, 2010, pp. 177–186.

[6] L. Bottou and Y. LeCun, "Large scale online learning," in *Advances in Neural Information Processing Systems*, 2004, pp. 217–224.

[7] J. Duchi, E. Hazan, and Y. Singer, "Adaptive subgradient methods for online learning and stochastic optimization," *Journal of Machine Learning Research*, vol. 12, pp. 2121–2159, 2011.

[8] Y. LeCun, D. Touresky, G. Hinton, and T. Sejnowski, "A theoretical framework for back-propagation," in *Proceedings of the 1988 connectionist models summer school*, vol. 1. CMU, Pittsburgh, Pa: Morgan Kaufmann, 1988, pp. 21–28.

[9] B. Recht, C. Re, S. Wright, and F. Niu, "Hogwild: A lock-free approach to parallelizing stochastic gradient descent," in *Advances in Neural Information Processing Systems*, 2011, pp. 693–701.

[10] H. Robbins and S. Monro, "A stochastic approximation method," *The annals of mathematical statistics*, pp. 400–407, 1951.

[11] M. Zinkevich, M. Weimer, L. Li, and A. Smola, "Parallelized stochastic gradient descent," in *Advances in Neural Information Processing Systems*, 2010, pp. 2595–2603.

[12] T. Zhang, "Solving large scale linear prediction problems using stochastic gradient descent algorithms," in *Proceedings of the 21st International Conference on Machine Learning*. ACM, 2004, p. 116.

[13] R. Byrd, S. Hansen, J. Nocedal, and Y. Singer, "A stochastic quasi-Newton method for large-scale optimization," *SIAM Journal on Optimization*, vol. 26, no. 2, pp. 1008–1031, 2016.

[14] Q. Le, J. Ngiam, A. Coates, A. Lahiri, B. Prochnow, and A. Ng, "On optimization methods for deep learning," in *Proceedings of the 28th International Conference on Machine Learning*. Omnipress, 2011, pp. 265–272.

[15] F. Curtis, "A self-correcting variable-metric algorithm for stochastic optimization," in *Proceedings of the 33rd International Conference on Machine Learning*, 2016, pp. 632–641.

[16] L. Bottou, F. Curtis, and J. Nocedal, "Optimization methods for large-scale machine learning," *SIAM Review*, vol. 60, no. 2, pp. 223–311, 2018.

[17] J. Martens, "Deep learning via Hessian-free optimization," in *Proceedings of the 27th International Conference on Machine Learning*, vol. 27, 2010, pp. 735–742.

[18] B. Pearlmutter, "Fast exact multiplication by the Hessian," *Neural computation*, vol. 6, no. 1, pp. 147–160, 1994.

[19] P. Werbos, "Backpropagation: Past and future," in *Proceedings of the Second International Conference on Neural Network*, vol. 1. IEEE, 1988, pp. 343–353.

[20] M. Møller, "Exact calculation of the product of the hessian matrix of feed-forward network error functions and a vector in $O(n)$ time," *DAIMI Report Series*, no. 432, 1993.

[21] F. Pineda, "Generalization of back-propagation to recurrent neural networks," *Physical Review Letters*, vol. 59, no. 19, p. 2229, 1987.

[22] L. Almeida, "A learning rule for asynchronous perceptrons with feedback in a combinatorial environment," in *Artificial neural networks: concept learning*. IEEE Press, 1990, pp. 102–111.

[23] J. M. Ortega and W. C. Rheinboldt, *Iterative solution of nonlinear equations in several variables*. SIAM, 2000.

[24] N. N. Schraudolph, "Fast curvature matrix-vector products for second-order gradient descent," *Neural computation*, vol. 14, no. 7, pp. 1723–1738, 2002.

[25] A. Conn, N. Gould, and P. Toint, *Trust-Region Methods*. Philadelphia, PA: Society for Industrial and Applied Mathematics (SIAM), 2000.

[26] J. Nocedal and S. Wright, *Numerical optimization*. Springer Science & Business Media, 2006.

[27] I. Goodfellow, Y. Bengio, and A. Courville, *Deep learning*. MIT press, 2016.

[28] Y. LeCun, C. Cortes, and C. Burges, "Mnist handwritten digit database," *AT&T Labs [Online]. Available: http://yann. lecun. com/exdb/mnist*, vol. 2, p. 18, 2010.