OPT2024: 16th Annual Workshop on Optimization for Machine Learning

Don't Be So Positive: Negative Step Sizes in Second-Order Methods

Betty Shea SHEAWS@CS.UBC.CA
Mark Schmidt[1] SCHMIDTM@CS.UBC.CA
The University of British Columbia, Vancouver, Canada. [1]*Canada CIFAR AI Chair (Amii)*

Abstract

The value of second-order methods lies in the use of curvature information. Yet, this information is costly to extract and once obtained, standard methods to achieve global convergence often discard valuable negative curvature information. This limits the effectiveness of second-order methods in modern machine learning. In this paper, we show that second-order and second-order-like methods are promising optimizers for neural networks provided that we add one ingredient: *negative step sizes*. We show that under very general conditions, methods that produce ascent directions are globally convergent when combined with a Wolfe line search that allows both positive and negative step sizes. We experimentally demonstrate that using negative step sizes is often more effective than common Hessian modification methods.

1. Introduction

Training neural networks involves minimizing a non-convex objective. Often, second-order methods are considered unsuitable for this task because they are attracted to saddle points and local maxima. By and large, gradient based methods such as gradient descent (GD) with momentum, Adam [15] and RMSprop [34] are seen as the only viable optimizers in machine learning. This is a source of frustration. Newton's method and second-order-like methods, such as limited-memory quasi-Newton (QN) methods, excel in traditional machine learning but can fail to even converge in the deep learning setting. First-order methods converge but slowly, as good convergence progress can only be expected if curvature information of the loss landscape is used. We want optimizers that use negative curvature information but these methods may yield search directions that point the wrong way.

Many variants of second-order methods ensure global convergence. Popular approaches include Hessian modifications, trust-region methods and cubic regularization (see Section 2 for a short summary). These methods use the same trick: align the search direction closer to the gradient direction to ensure descent. Yet, this approach could slow down optimization progress. For problems that are ill-conditioned and non-convex, GD ensures progress but at an extremely slow rate [20]. When the second-order direction points uphill, simply taking the opposite direction may be a better choice than moving the step closer to the gradient step. The negative direction points downhill *and* maintains all second-order information (see Figure 1.1). To the best of our knowledge, using negative step sizes is relatively unexplored in optimization. We found one mention of negative step sizes in an analysis of the Fletcher-Reeves [11] non-linear conjugate gradient method [9] for differentiable and Lipschitz-smooth objective functions.

© B. Shea & M. Schmidt[1].

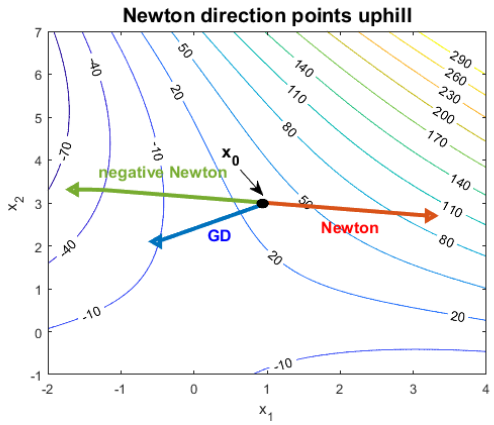

Figure 1: In this two-dimensional non-convex minimization problem, the negative of the Newton's direction (green) is a better search direction than the gradient descent direction (blue), Newton's direction (red) or any non-negative combination of both.

1.1. Contribution

This paper examines the largely unexplored role of negative step sizes in optimization and shows that taking a backward step is a computationally inexpensive way to incorporate negative curvature information. Our experiments suggest that a quasi-Newton (QN) method, symmetric rank one (SR1) [8], combined with negative step sizes, may be an overlooked method useful for training neural networks. This is achieved without using common approaches to "fix" second-order methods[1] such as Hessian modifications. We also show that extending the Wolfe line search to both positive and negative step sizes ensures that second-order methods satisfy the Zoutendijk condition. Thus, second-order methods combined with good step size choices are globally convergent even in non-convex settings.

2. Common approaches for globally convergent second-order methods

Two major drawbacks of using second-order methods to train neural networks are (a) the computation and storage cost of the (approximate) Hessian and, (b) non-convergence. To some extent, the first issue is addressed by QN methods and their limited memory versions [25]. In this section, we summarize some common approaches to the second issue of non-convergence.

Hessian modifications There are many variants of second-order method that modify the (approximate) Hessian for positive definiteness and thus ensure a descent direction (see Sections 3.4 of [25]). A spectral decomposition will find the negative eigenvalues. One could then replace the negative eigenvalues by small positive values, flip the sign of the negative eigenvalues [27] or shift the Hessian by subtracting a diagonal matrix that contains the most negative eigenvalue of the Hessian

1. In this paper, we refer to second-order-like methods that approximate the Hessian, such as QN methods, as "second-order methods". This is technically incorrect because these methods do not compute second derivatives. This misuse of terminology is for readability.

(also known as damping). These modifications, however, require an expensive eigenvalue decomposition and may no longer reflect accurate curvature information. Removing negative eigenvalues could also be detrimental for high-dimensional non-convex problems because following paths of negative curvature is a way to escape saddle points [6, 22].

Trust-region approaches Trust region (TR) methods are an option for obtaining global convergence and, unlike line searches, they work with methods that produce ascent directions [7]. Variants of TR methods include the Leveberg-Marquadt [16, 19] and dogleg [28] methods. Newton's and QN methods often employ a TR approach that minimizes the model function over a two-dimensional subspace [5, 29, 30, 32]. Saddle-free Newton's method [10, 26], Newton's method with *generalized* TR, was proposed as a version of Newton's method that can escape saddle points. However, because the solution to the TR subproblem is equivalent to adding a positive definite diagonal matrix to the (approximate) Hessian [23], this approach also loses curvature information. TR conjugate gradient methods [33] use directions of negative curvature, but this also combines the original search direction with the negative gradient direction.

Cubic regularization Cubic-regularization (CR) uses an additional third-order term so that the second-order method is not attracted to local maxima and saddle points. Originally introduced for Newton's method [24], CR was recently extended to QN methods [2, 13]. However, CR generally requires special solvers for its third-order subproblem. CR is equivalent to using a Hessian that is modified to be positive definite [18].

Positive definite-enforcing updates Some QN methods, such as BFGS [3, 11, 12, 31], are designed to guarantee positive definiteness of the Hessian approximation at every update step as long as the step size used satisfies the Wolfe conditions [35]. It has been suggested, however, that QN variants that do not enforce positive definiteness, such as SR1, converge faster by adhering closer to the true Hessian [8, 14]. Other updates that enforce positive (semi-) definiteness include natural gradient methods [1] that use the covariance matrix of the gradients instead of the Hessian, Gauss-Newton methods and Kronecker-factored approximate curvature (KFAC) [21]. These methods, however, may also sacrifice accurate curvature information by enforcing positive (semi-) definiteness.

3. Global convergence of ascent-descent methods

Even though negative step sizes may result in a larger decrease on some iterations, we may be concerned that the algorithm might not converge. In this section, we give a general result showing that methods that may produce ascent directions and that allow negative step sizes converge under very general conditions. Our goal is to minimize a function $f : \mathbb{R}^n \to \mathbb{R}$ without constraints. The function f is assumed to be twice differentiable, Lipschitz-smooth and bounded below. We use a deterministic iterative algorithm that, on the kth iteration, calculates a search direction $\boldsymbol{p}_k$ and step size α_k, and updates its iterate $\boldsymbol{x}$ by

$$\boldsymbol{x}_{k+1} = \boldsymbol{x}_k + \alpha_k \boldsymbol{p}_k. \tag{1}$$

Given our iterate $\boldsymbol{x}_k$ and a search direction $\boldsymbol{p}_k$, we can rewrite f as a function of the step size $\phi(\alpha) = f(\boldsymbol{x}_k + \alpha \boldsymbol{p}_k)$. The directional derivative is defined as its derivative with respect to α, i.e. $\phi'(\alpha) = \nabla f(\boldsymbol{x}_k + \alpha \boldsymbol{p}_k)^\top \boldsymbol{p}_k$. We refer to $\boldsymbol{p}_k$ as an *ascent* or a *descent* direction when $\phi'(0)$ is

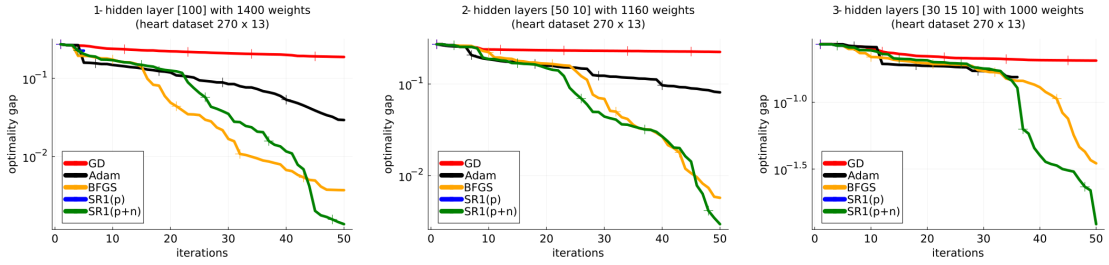

Figure 2: Using negative step sizes in training neural networks. The plots show training error by iteration for neural networks with 1-, 2- and 3- hidden layers on the `heart` dataset. We compare full QN methods BFGS (yellow) and SR1 (blue and green) with GD (red) and Adam (black). SR1 is non-convergent with positive-only step sizes (blue) but outperforms when step sizes are allowed to be negative (green). The outperformance appears more pronounced in deeper networks.

positive or negative respectively. We say that a method is *globally convergent* if it finds a stationary point using update rule 1 starting from any initial estimate $\boldsymbol{x}_0$. Further details of notation, definitions and assumptions are given in Appendix A.

It is known that methods that produce descent-only directions with step sizes satisfying the Wolfe conditions [35] are globally convergent for a loss function f that satisfies the assumptions above [25]. The proposition below extends this result to methods that produce both ascent and descent directions, and that satisfy a variant of the Wolfe conditions that considers both positive and negative step sizes. Global convergence holds with the looser requirement that $\boldsymbol{p}_k$ is not orthogonal to the gradient direction.

Proposition 1 *Consider any algorithm with updates of the form equation 1 where $\boldsymbol{p}_k$ is a direction that is not orthogonal to the gradient direction, and non-zero step sizes $|\alpha_k| \geq \epsilon > 0$ satisfy the Wolfe conditions (6 and 7). Suppose that f is bounded below in $\mathcal{R}^n$ and that f is continuously differentiable in an open set $\mathcal{N}$ containing the level set $\mathcal{L} \triangleq \{\boldsymbol{x} : f(\boldsymbol{x}) \leq f(\boldsymbol{x}_0)\}$, where $\boldsymbol{x}_0$ is the initial iterate. Assume also that the gradient ∇f is Lipschitz continuous on $\mathcal{N}$. Then, the algorithm satisfies Zoutendijk's condition (4) is therefore globally convergent (2).*

The proof of Proposition 1 is given in Appendix B. A modified Wolfe line search that also considers negative step sizes could be easily implemented. We give one possible implementation, which we call `Wolfe±`, in Appendix C.

4. Experiments

4.1. QN methods and neural network training

In this experiment, we looked at full QN methods and their performance on neural network training. All methods use a line search that gives step sizes satisfying the Wolfe conditions (`p`). SR1 is further combined with `Wolfe±` (`p+n`) that uses negative step sizes when $\boldsymbol{p}_k$ is not a descent direction. QN methods are initialized with the identity matrix. Methods are run for a maximum of 50 iterations. Figure 2 shows training error by iteration.

Dataset	size	GD	Adam	l-BFGS	l-SR1	l-SR1+damp	l-SR1+`Wolfe±`
a1a	(1605×119)	0.217	0.203	0.315	0.500	0.214	**0.199**
a9a	(32561×123)	0.220	0.212	0.332	0.330	0.216	**0.211**
colon-cancer	(62×2000)	0.006	0.046	0.500	0.183	*DNF*	**0.003**
gisette	(6000×5000)	0.139	0.171	0.395	0.395	*DNF*	**0.050**
heart	(270×13)	0.455	0.382	0.388	**0.237**	0.239	**0.237**
ijcnn1	(35000×22)	0.132	**0.120**	0.131	0.165	0.130	**0.120**
ionosphere	(351×34)	0.201	**0.118**	0.201	0.500	0.199	0.170
leukemia	(38×7129)	0.028	**0.001**	0.160	0.160	*DNF*	0.064
madelon	(2000×500)	0.500	0.500	0.490	0.396	**0.382**	0.396
mushrooms	(8124×112)	0.040	0.018	0.166	0.500	0.019	**0.009**
splice	(1000×60)	0.270	0.260	0.485	0.500	0.262	**0.195**
svmguide3	(1243×22)	0.295	0.276	0.374	0.374	0.267	**0.260**
w1a	(2477×300)	0.088	**0.059**	0.200	0.200	0.071	0.063
w8a	(49749×300)	0.096	**0.066**	0.193	0.193	0.079	0.071

Table 1: Comparing the use of negative step sizes with damping, a common Hessian modification, for limited-memory QN methods. The experiment was run across several datasets fitted with a neural network. The lowest training error achieved in each dataset is highlighted in bold. Limited-memory SR1 with positive only step sizes (`l-SR1`) often does not converge. Damping (`l-SR1+damp`) helps with convergence but is an expensive operation and in many cases did not finish training in the allocated time (shown as *DNF*). Using negative step sizes (`l-SR1+Wolfe±`) showed good performance even against a state-of-the-art optimizer such as Adam.

Because the objective is non-convex, the true Hessian at iterate $\boldsymbol{x}_k$ may not be positive definite. Thus, SR1 produces Hessian approximations that are not positive definite and a $\boldsymbol{p}_k$ that does not point downhill. With a regular line search, SR1 diverges within the first few steps (blue). With negative step sizes, SR1 becomes the an effective optimizer. This effect appears to be more pronounced as the number of network layers increase. BFGS combined with a Wolfe line search guarantees positive definiteness in its Hessian update and thus never diverges even with positive only step sizes. Yet, a positive definite Hessian may not accurately capture the optimization landscape. The lack of negative curvature information in the BFGS Hessian approximation may be the reason its training error is higher than that of SR1 after the maximum number of iterations is reached. For further details, please refer to Appendix D.

4.2. Limited-memory QN methods and neural network training

Table 1 shows the result of using two limited-memory QN methods, l-BFGS [17] and l-SR1 [4], to train a neural network across different datasets. Gradient descent (GD) and Adam are plotted for comparison. The purpose of this experiment is to compare two globalization strategies: damping and negative step sizes. Damping requires an eigenvalue decomposition and is therefore an expensive operation. Thus, in many cases where the dataset has a large number of features and the

Hessian is large (e.g. `colon-cancer`), l-SR1 combined with damping did not finish in the allocated amount of time (shown as *DNF*). l-BFGS combined with a Wolfe line search does not require damping because its Hessian approximations are guaranteed to be positive definite. Yet, a lack of negative curvature information may hurt its effectiveness in training neural networks, and it tends to perform worse than l-SR1 with negative step sizes.

Acknowledgements

Betty Shea is funded by an NSERC Canada Graduate Scholarship. The work was partially supported by the Canada CIFAR AI Chair Program and NSERC Discovery Grant RGPIN-2022-036669.

References

[1] S. I. Amari. Natural gradient works efficiently in learning. *Neural Computation*, 10:251–276, 1998.

[2] H. Y.. Benson and D. F. Shanno. Cubic regularization in symmetric rank-1 quasi-Newton methods. *Math. Prog. Comp.*, 10:457–486, 2018.

[3] C. G. Broyden. Quasi-Newton methods and their application to function minimisation. *Mathematics of Computation*, 21(99):368–381, 1967.

[4] R.H. Bryd, J. Nocedal, and R.B. Schnabel. Representations of quasi-Newton matrices and their use in limited memory methods. *Mathematical Programming*, 63:129–156, 1994.

[5] R.H. Byrd, R.B. Schnabel, and G.A. Schultz. Approximate solution of the trust regions problem by minimization over two-dimensional subspaces. *Mathematical Programming*, 40:247–263, 1988.

[6] A. Choromanska, M. Henaff, M. Mathieu, G. B. Arous, and Y. LeCun. The loss surfaces of multilayer networks. *AISTATS*, 38:192–204, 2015.

[7] A. Conn, N. Gould, and P. Toint. *Trust-region methods*. MPS/SIAM Series on Optimization. SIAM, 2000. ISBN 0-89871-460-5.

[8] A.R. Conn, N.I.M. Gould, and P.L. Toint. Convergence of quasi-Newton matrices generated by the symmetric rank one update. *Mathematical Programming*, 50:177–195, 1991.

[9] Y. Dai. Further insight into the convergence of the fletcher-reeves method. *Sci. China Ser. A-Math*, 42:905–916, 1999.

[10] Y. Dauphin, R. Pascanu, C. Gulcehre, K. Cho, S. Ganguli, and Y. Bengio. Identifying and attacking the saddle point problem in high-dimensional non-convex optimization. *arXiv:1406.2572*, 2014.

[11] R. Fletcher and C. M. Reeves. Function minimization by conjugate gradients. *The computer journal*, 7(2):149–154, 1964.

[12] D. Goldfarb. A family of variable-metric methods derived by variational means. *Mathematics of Computation*, 24(109):23–26, 1970.

[13] D. Kamzolov, K. Ziu, A. Agafonov, and Takáč. Cubic regularization is the key! the first accelerated quasi-Newton method with a global convergence rate of $o(k^{-1})$ for convex functions. *arXiv:2302.04987*, 2023.

[14] H. F. Khalfan, R. H. Byrd, and R. B. Schnabel. A theoretical and experimental study of the symmetric rank-one update. *SIAM Journal on Optimization*, 3(1):1–24, 1993.

[15] D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. *arXiv:1412.6980*, 2014.

[16] K. Levenberg. A method for the solution of certain non-linear problems in least squares. *Quarterly of Applied Mathematics*, 2:164–168, 1944.

[17] D. C. Liu and J. Nocedal. On the limited memory BFGS method for large scale optimization. *Mathematical Programming*, 45:503–528, 1989.

[18] Y. Malitsky and K. Mishchenko. Adaptive gradient descent without descent. In Hal Daumé III and Aarti Singh, editors, *Proceedings of the 37th International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research*, pages 6702–6712. PMLR, 13–18 Jul 2020.

[19] D.W. Marquardt. An algorithm for least squares estimation of non-linear parameters. *SIAM Journal*, 11:431–441, 1963.

[20] J. Martens. Deep learning via hessian-free optimization. In *ICML*, volume 27, 2010.

[21] J. Martens and R. Grosse. Optimizing neural networks with kronecker-factored approximate curvature. In *ICML*, 2015.

[22] J. J. Moré and D. C. Sorensen. On the use of directions of negative curvature in a modified newton method. *Mathematical Programming*, 16:1–20, 1979.

[23] Jorge J. Moré and D. C. Sorensen. Computing a trust region step. *SIAM Journal on Scientific and Statistical Computing*, 4(3):553–572, 1983.

[24] Y. Nesterov and B. Polyak. Cubic regularization of Newton method and its global performance. *Math. Program.*, 108:177–205, 2006.

[25] J. Nocedal and S. J. Wright. *Numerical Optimization, 2nd Ed.* Springer, 2006.

[26] R. Pascanu, Ganguli S. Dauphin, Y. N., and Y. Bengio. On the saddle point problem for non-convex optimization. *arXiv:1405.4604*, 2014.

[27] S. Paternain, A. Mokhtari, and A. Ribeiro. A Newton-based method for nonconvex optimization with fast evasion of saddle points. *SIAM Journal on Optimization*, 29(1):343–368, 2019.

[28] M.J.D. Powell. A new algorithm for unconstrained optimization. *Nonlinear Programming*, pages 31–66, 1970.

[29] V. Ramanmurthy and N. Duffy. L-sr1: A novel second order optimization method for deep learning, 2016.

[30] G.A. Schultz, R.B. Schnabel, and R.H. Byrd. A family of trust-region-based algorithms for unconstrained minimization with strong global convergence properties. *SIAM Journal on Numerical Analysis*, 22:47–67, 1985.

[31] D. F. Shanno. Conditioning of quasi-newton methods for function minimization. *Mathematics of Computation*, 24(111):647–656, 1970.

[32] D. C. Sorensen. Newton's method with a model trust region modification. *SIAM Journal on Numerical Analysis*, 19(2):409–426, 1982.

[33] T. Steihaug. The conjugate gradient method and trust regions in large scale optimization. *SIAM Journal on Numerical Analysis*, 20(3), 1983.

[34] T. Tieleman and G. Hinton. Lecture 6.5 - rmsprop: Divident the gradient by a running average of its recent magnitude, 2012.

[35] P. Wolfe. Convergence conditions for ascent methods. *SIAM review*, 11(2):226–235, 1969.

Appendix A. Notation and further background details

We say an algorithm is *globally convergent* if it produces a sequence of gradients that converge to zero, or

$$\liminf_{k\to\infty} \|\nabla f(\boldsymbol{x}_k)\| = 0. \tag{2}$$

The angle θ_k between search direction $\boldsymbol{p}_k$ and the steepest descent direction $-\nabla f(\boldsymbol{x}_k)$ is given by

$$\cos\theta_k = \frac{-\nabla f(\boldsymbol{x}_k)^\top \boldsymbol{p}_k}{\|\nabla f(\boldsymbol{x}_k)\|\|\boldsymbol{p}_k\|}. \tag{3}$$

Equation (3) is sometimes referred to as the cosine similarity between the $\boldsymbol{p}_k$ and the steepest descent direction. Search direction $\boldsymbol{p}_k$ is not orthogonal to the gradient if $|\cos\theta_k| \geq \delta > 0$.

Zoutendijk's condition is satisfied if

$$\sum_{k\geq 0} \cos^2\theta_k \|\nabla f(\boldsymbol{x}_k)\|^2 < \infty, \tag{4}$$

which implies that

$$\cos^2\theta_k \|\nabla f(\boldsymbol{x}_k)\|^2 \to 0.$$

If f has a *Lipschitz continuous gradient* on open set $\mathcal{N}$, then there exists a constant $L > 0$ such that

$$\|\nabla f(\boldsymbol{x}) - \nabla f(\tilde{\boldsymbol{x}})\| \leq L\|\boldsymbol{x} - \tilde{\boldsymbol{x}}\|, \text{ for all } \boldsymbol{x}, \tilde{\boldsymbol{x}} \in \mathcal{N} \tag{5}$$

A line search is an auxiliary method that finds a step size α_k along $\boldsymbol{p}_k$ that has desirable properties. For example, for constants $c_1 \in (0, 1)$ and $c_2 \in (c_1, 1)$ and descent direction $\boldsymbol{p}_k$, a step size $\alpha_k > 0$ that satisfies the Wolfe condition has two desirable properties. Firstly, α_k guarantees making progress in minimizing f, or

$$f(\boldsymbol{x}_k + \alpha_k \boldsymbol{p}_k) \leq f(\boldsymbol{x}_k) + c_1\alpha_k \nabla f(\boldsymbol{x}_k)^\top \boldsymbol{p}_k. \tag{6}$$

Equation 6 is also known as the Armijo condition. Secondly, α_k does not lie too far from an optimal step size choice.

$$0 \geq \nabla f(\boldsymbol{x}_k + \alpha_k \boldsymbol{p}_k)^\top \boldsymbol{p}_k \geq c_2 \nabla f(\boldsymbol{x}_k)^\top \boldsymbol{p}_k. \tag{7}$$

Equation equation 7 is also known as the curvature condition. Taken together, equations equation 6 and equation 7 are often referred to as the Wolfe conditions. Note that this standard curvature condition assumes that $\boldsymbol{p}_k$ is a descent direction and thus $\nabla f(\boldsymbol{x}_k)^\top \boldsymbol{p}_k < 0$. In our case where $\boldsymbol{p}_k$ is an ascent direction, the curvature condition becomes

$$0 \leq \nabla f(\boldsymbol{x}_k + \alpha_k \boldsymbol{p}_k)^\top \boldsymbol{p}_k \leq c_2 \nabla f(\boldsymbol{x}_k)^\top \boldsymbol{p}_k. \tag{8}$$

Appendix B. Proof of Proposition 1

Proof The proof of Theorem 1 follows closely that of Theorem 3.2 in [25], which considers only the case of descent directions paired with positive step sizes. Here we extend the results to ascent directions $\boldsymbol{p}_k$ paired with negative step sizes α_k.

Combining update rule (1) and curvature condition (8) where $\nabla f(\boldsymbol{x}_k)^\top \boldsymbol{p}_k > 0$ and $\alpha_k < 0$ gives

$$\left(\nabla f(\boldsymbol{x}_k) - \nabla f(\boldsymbol{x}_{k+1})\right)^\top \boldsymbol{p}_k \geq (1 - c_2)\nabla f(\boldsymbol{x}_k)^\top \boldsymbol{p}_k. \tag{9}$$

Lipschitz continuous gradient gives

$$\|\nabla f(\boldsymbol{x}_k) - \nabla f(\boldsymbol{x}_{k+1})\| \leq L\|\boldsymbol{x}_k - \boldsymbol{x}_{k+1}\| = L\| - \alpha_k \boldsymbol{p}_k\| = |-\alpha_k| L\|\boldsymbol{p}_k\| = -\alpha_k L\|\boldsymbol{p}_k\|$$

and

$$\left(\nabla f(\boldsymbol{x}_k) - \nabla f(\boldsymbol{x}_{k+1})\right)^\top \boldsymbol{p}_k \leq -\alpha_k L\|\boldsymbol{p}_k\|^2. \tag{10}$$

Combining (9) and (10)

$$(1 - c_2)\nabla f(\boldsymbol{x}_k)^\top \boldsymbol{p}_k \leq -\alpha_k L\|\boldsymbol{p}_k\|^2.$$

Rearranging gives

$$\alpha_k \leq -\frac{1 - c_2}{L} \frac{\nabla f(\boldsymbol{x}_k)^\top \boldsymbol{p}_k}{\|\boldsymbol{p}_k\|^2} \leq 0 \tag{11}$$

where the middle term in inequality (11) is negative because $0 < c_1 < c_2 < 1$ and $\nabla f(\boldsymbol{x}_k)^\top \boldsymbol{p}_k > 0$. Using (11) with the sufficient decrease condition (6) gives

$$\begin{aligned}
f(\boldsymbol{x}_{k+1}) &\leq f(\boldsymbol{x}_k) + c_1 \alpha_k \nabla f(\boldsymbol{x}_k)^\top \boldsymbol{p}_k \\
&\leq f(\boldsymbol{x}_k) - \frac{c_1(1 - c_2)}{L} \frac{\left(\nabla f(\boldsymbol{x}_k)^\top \boldsymbol{p}_k\right)^2}{\|\boldsymbol{p}_k\|^2} \\
&= f(\boldsymbol{x}_k) - \frac{c_1(1 - c_2)}{L} \left(\frac{\left(\nabla f(\boldsymbol{x}_k)^\top \boldsymbol{p}_k\right)}{\|\nabla f(\boldsymbol{x}_k)\|\|\boldsymbol{p}_k\|}\right)^2 \|\nabla f(\boldsymbol{x}_k)\|^2 \\
&= f(\boldsymbol{x}_k) - c \cdot \cos^2 \theta_k \|\nabla f(\boldsymbol{x}_k)\|^2
\end{aligned}$$

where the last step uses (3) and $c = \frac{c_1(1-c_2)}{L}$. Summing across iterations 0 to k,

$$f(\boldsymbol{x}_{k+1}) \leq f(\boldsymbol{x}_0) - c \sum_{j=0}^{k} \cos^2 \theta_j \|\nabla f(\boldsymbol{x}_j)\|^2$$

Because f is bounded below, we know that $f(\boldsymbol{x}_0) - f(x_k)$ is less than a positive constant for all k. Taking limits gives

$$\sum_{k=0}^{\infty} \cos^2 \theta_k \|\nabla f(\boldsymbol{x}_k)\|^2 < \infty$$

and thus satisfies the Zoutendijk condition, and implies that

$$\cos_k^\theta \|\nabla f(\boldsymbol{x}_k)\|^2 \to 0$$

Because $|\cos\theta_k| \geq \delta > 0$ for all k, then $\cos^2\theta_k \geq \delta^2 > 0$ for all k. This further implies that

$$\lim_{k\to\infty} \|\nabla f(\boldsymbol{x}_k)\| = 0$$

and thus the algorithm is globally convergent. ■

Appendix C. Generalized Wolfe conditions

Suppose we have an implementation of the standard Wolfe line search ([25] Section 3.5). Then `Wolfe±` can be implemented trivially as in Algorithm C. First we check the sign of the directional derivative. If $\boldsymbol{p}_k$ is a descent direction, then we continue by calling the standard Wolfe line search. If $\boldsymbol{p}_k$ is an ascent direction, then the standard Wolfe line search is called with $-\boldsymbol{p}_k$ as the direction and the step size returned is negated afterwards.

Input: $\alpha_{\max} > 0$, p_k, function `Wolfe(`p_k`,` $\alpha_{\max}$`)`
Output: step size $\alpha_{W\pm}$ satisfying `Wolfe±`
if $\nabla f(x_k)^\top p_k < 0$ **then**
  $\alpha_{W\pm} \leftarrow$ `Wolfe(`p_k`,` $\alpha_{\max}$`)`;
else
  $\alpha_{W\pm} \leftarrow -$ `Wolfe(`$-p_k$`,` $\alpha_{\max}$`)`;
end

Algorithm 1: `Wolfe±` is easy to implement with access to a standard Wolfe line search.

Appendix D. Further details on experiments

Figure 3 plots the step sizes and cosine similarity of the search direction $\boldsymbol{p}_k$ with the steepest descent direction of the different optimizers in the experiments discussed in Section 4.1.

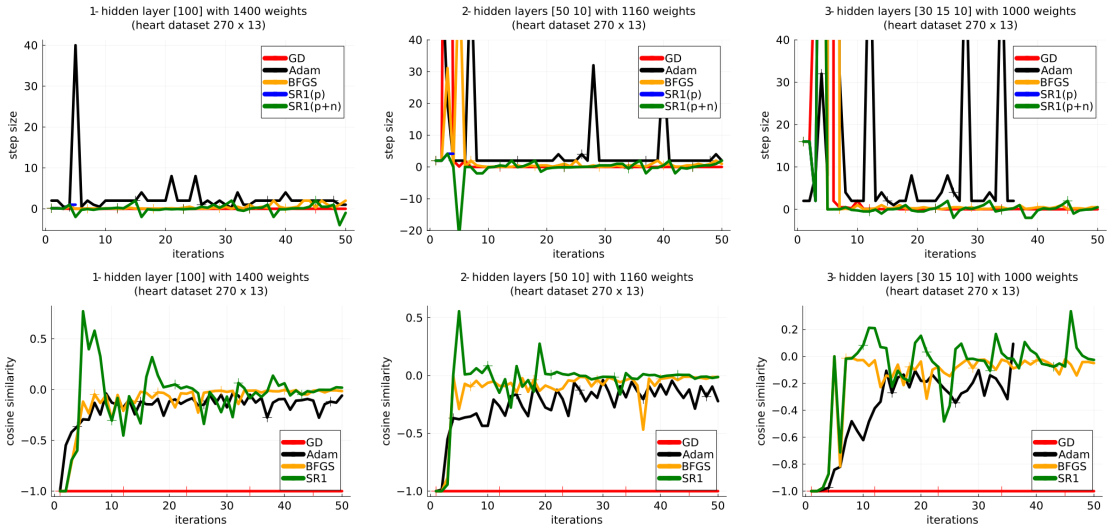

Figure 3: Step sizes (top row) and cosine similarity (bottom row) of different optimizers for neural networks with 1, 2 and 3 hidden layers. SR1 often produces ascent directions and this is dealt with effectively by using negative step sizes.