Published in final edited form as:

Methods Mol Biol. 2021 ; 2190: 307–316. doi:10.1007/978-1-0716-0826-5_15.

A Hybrid Levenberg–Marquardt Algorithm on a Recursive Neural Network for Scoring Protein Models

Eshel Faraggi^{1,2}, Robert L Jernigan³, Andrzej Kloczkowski^{4,5}

¹Research and Information Systems, LLC, Indianapolis, IN, USA.

²Department of Physics, Indiana University Purdue University Indianapolis, Indianapolis, IN, USA.

³Roy J. Carver Department of Biochemistry, Biophysics and Molecular Biology, Iowa State University, Ames, IA, USA.

⁴Battelle Center for Mathematical Medicine, The Research Institute at Nationwide Children's Hospital, Columbus, OH, USA.

⁵Department of Pediatrics, The Ohio State University, Columbus, OH, USA.

Abstract

We have studied the ability of three types of neural networks to predict the closeness of a given protein model to the native structure associated with its sequence. We show that a partial combination of the Levenberg–Marquardt algorithm and the back-propagation algorithm produced the best results, giving the lowest error and largest Pearson correlation coefficient. We also find, as previous studies, that adding associative memory to a neural network improves its performance. Additionally, we find that the hybrid method we propose was the most robust in the sense that other configurations of it experienced less decline in comparison to the other methods. We find that the hybrid networks also undergo more fluctuations on the path to convergence. We propose that these fluctuations allow for better sampling. Overall we find it may be beneficial to treat different parts of a neural network with varied computational approaches during optimization.

Keywords

Neural network; Levenberg–Marquardt algorithm; Associative memory; Protein structure; Model quality assessment

1 Introduction

The promise, deployment, and fear of machine learning applications has grown significantly in recent years. Improvements in hardware, software, and our evermore digital activities combine to create the environment where machine learning applications are finding their way into everyday items. Machine learning also holds a promise to help tackle question about complex systems such as living beings. On the fear side, applications using machine learning can be abused by oppressive regimes to control and restrict their population.

Additionally, there is some fear of machine learning algorithms run amok, gaining self-evolving properties and getting loose. Here, we will focus on using machine learning to illuminate on parts of the protein structure problem.

The complexity of characterizing protein models, coupled with larger and larger datasets, represent a challenge to machine learners [1–5]. On the one hand attempting to better characterize proteins leads to more input features describing the protein. This leads to a high dimension input feature space, and with it an increase in the abundance of local minima for the calculated error functions, and a reduction in the ability to find low error solutions using the standard back-propagation (BP) algorithm. On the other hand, the abundance of data means that a faster converging artificial learner will be able to cover a wider swath of the space of examples and hence obtain better learning. This tends to limit the usefulness of more elaborate algorithms that find multi-dimensional directions of change, rather than making changes along single dimensions as in the BP algorithm.

Here we present preliminary results for a hybrid of the Levenberg–Marquardt (LM) and the BP algorithm. In this approach a subset of the weights undergoes a Levenberg–Marquardt step at each epoch. This approach is useful in cases where a crucial part of the network is of a limited extent, while other parts may be very large. For scoring the closeness of a protein model to its native structure as would be displayed for example in an X-ray diffraction, a single real-valued number is enough. For example, one can use the RMSD (root mean square distance) between atoms/residues, or the Template Modeling score (TM-score) [6–8]. On the other hand, as far as we know, many numbers are needed to try and characterize a protein's state. We split the network weights into two parts: a large part associated with protein characterization and a smaller part associated with coalescing this information into an estimate of the TM-score. We apply a BP epoch to all weights and then apply a second LM epoch on the small set of weights connecting to the single node output layer. One should note that for a lack of computer power we are unable to apply the LM algorithm to the entire neural network.

The problem of estimating how close a protein decoy model is to its native structure, if the native structure is unknown, is of significant importance in biology. An ability to do this would aid in estimating the effects of genetic variation and designing new therapies. As we have presented in previous work [3], we use accessible surface area [9], residue depth [10, 11], and residue configuration [3], to create partial sums, delineated by residue types, that characterize the protein. Using a training set of Protein Databank (PDB) [12, 13] structures and computational models we train our networks to use this information

2 Material and Methods

Except where otherwise noted, we use the same methodology as in the Sederf scoring function [3]. Protein residue sequences are scored based on their similarity to the native protein structure of the sequence as in the PDB. We used a two-layer feed-forward neural network with momentum, recently described in detail [14]. A diagram of the neural network architecture is given in Fig. 1.

HL1 and HL2 refer to the first and second hidden layers, respectively, and W1, W2, and W3 refer to the weights connecting the different layers. We used an all-connected network in which weights connect all the nodes of one layer to all the nodes of the next layer. We use a bias node for the input and hidden layers and a bipolar hyperbolic tangent activation function. For the activation parameter we use the symbol a . Momentum refers to the contribution of the gradient calculated at the previous time step to the correction of the weights. We use the symbol p for the momentum parameter. To make this network a recurrent neural network the values of the nodes of both hidden layers are used as inputs for the following epoch. In this respect the network is said to have associative memory [15–18].

The LM algorithm locates a local minimum of the sum of squares of error. It alternates between the steepest descent and the Gauss–Newton method. When the error is large, the algorithm behaves more like a steepest descent method. When the error is lower, it becomes more like the Gauss–Newton method. Roughly it does this by calculating the local Jacobian and solving an error associated matrix equation. The implementation of the LM algorithm here follows that of Hagan and Menhaj [19], and Wilamowski et al. [20]. The reader is encouraged to consult the literature [19–24] for a review of the LM method.

The score we apply in this case is the closeness of a given protein structure to the native structure as given by the TM-score. Other scores can be used, for example, the probability a given mutation would cause harm. The TM-score is calculated from the distances between the residues of two aligned protein sequences:

$$\text{TM-score} = \max \left[\frac{1}{L_t} \sum_{i=1}^{L_a} \frac{1}{1 + \left(\frac{d_i}{d_0} \right)^2} \right], \quad (1)$$

where the maximum is taken over all possible alignments. L_t is the length of the target protein, L_a is the length of the protein that is aligned to it, d_i is the distance between the residues at alignment location i , and d_0 is a scaling distance, optimized to $1.24\sqrt[3]{L_t - 15} - 1.8$. With such calibration, the TM-score does not depend on the protein length, and varies between 0 and 1, with 1 indicating a perfect match. A TM-score below 0.2 indicates structurally unrelated proteins, while a TM-score greater than 0.5 indicates the two proteins belong to the same fold. We estimate the efficiency of the new neural network on the problem of estimating the “correctness” of protein models.

The input features we use are described in Table 1. A fuller description of the input features is provided with Seder1 [3]. Shortly, **res2res** refers to the partial energy sums between pairs of residues. In this case we binned distances according to the following bins (in Å): [0–3], [3,6], [6,9], and [9, ∞]. **typ233w** contains information about the distance of individual atoms to the solvent and is a critical piece of information. Atoms associated with different residue types are distinguished from each other. Partial sums with respect to the inverse distance and its cube are calculated. A total of 233 heavy atom-residue types are used and the 234th type is used for unknown assignments. A stand-alone version of the DEPTH program [10, 11] is used to calculate the distance to the solvent. Distances are defined by the

shortest paths to a water molecule in a solvated protein. **cmap** refers to a residue-residue contact map with a cutoff distance of 5 and 7 Å. In this case we used the standard 20 residue types, and a 21st type reserved for unknown cases. A final z-score normalization was applied to all features. In total we have 8436 input features. The input and two hidden layers each have a bias neuron. A single output is produced, i.e., the output layer consists of a single neuron trained to predict the TM-score to native.

In Fig. 2 we give the heuristic code for the algorithms studied. The full algorithm represents the hybrid approach we propose here where we apply alternating BP and LM epochs. We shall refer to it as HLM. If steps 2.d–f. are removed, we obtain a back-propagating algorithm with associative memory. We refer to it as MEM. Finally, if in addition to steps 2.d–f. we remove also step 2.b., we obtain a back-propagating neural network. We refer to it as GNN. In what follows we compare the ability of these three approaches to predict the TM-score.

To train these three approaches we used a random set of 30,000 protein models. These models are for sequences for which native coordinates exist in the PDB [12, 13]. They are randomly selected from a list of protein models; this list is composed of native models from the PDB, server models from the Critical Assessment of protein Structure Prediction (CASP) [25], and decoy models from Decoys ‘R’Us [26]. We randomly select 5% of the training set as an over-fit-protection set, and do not use them for training. The remaining protein models are used to optimize the weights of the neural network so as to reduce the error between predicted and observed TM-score to native of a given model. To test these models we used another random set of 30,000 protein models that are distinct from the 30,000 models used for training and over-fit-protection.

3 Results

We wish to compare the effectiveness of the optimization of weights using the three methods (GNN, MEM, HLM). We do this by comparing which method obtained the best optimized weights and how robust was this optimization. For identical inputs, a neural network has various parameters that control its ability to learn. The networks presented here share the following parameters that determine their architecture and dynamics: the size of the hidden layers, the learning rate, the momentum, and the activation parameter. We studied the optimization of the network for variations of these parameters. The networks configurations with the lowest error and the variations of these parameters are given in Table 2. For the rest of the discussion we shall refer to these configurations as “best networks.” We also sampled variations in the learning rate and the momentum. For the learning rate we found all best networks had a value of $\mu = 0.00178$. For the momentum we found that GNN optimized best as in previous studies [3, 27–29] with $p = 0.4$. However, we found $p = 0$ was the optimal value for MEM and HLM.

We start with an example of the convergence. For the three cases of the best networks, we give the training convergence curves in Fig. 3. In Fig. 3a we present the error for the over-fit-protection set as function of epoch. The upper curve in green is for the MEM network, the intermediate curve in blue is for the GNN network, and the bottom curve in red is for the

HLM network. In Fig. 3b we give the error for the over-fit-protection set versus the error for the training set. The color assignments in this case are the same as in part a.

The lowest absolute-value error for the best networks is given in Table 3. Additionally, we calculated the average absolute-value error for the top five architectures to study the robustness of the networks. These results are given in the table as *Top5*. The Pearson correlation coefficient (PCC) was also used to estimate the accuracy of the networks and is also given in Table 3.

4 Discussion

Optimization of the three types of networks we studied yielded identical values for the MEM and HLM versions with an 11-3 configuration for the hidden layers. The optimized GNN version showed an increase in the second hidden layer, possibly due to its lack of memory. No variations in the activation parameter or the learning rate was found between the three optimized versions. However, while the top five networks for the HLM approach all had $\mu = 0.00178$, the MEM and GNN networks each had one case in the top five where $\mu = 0.000178$. For the momentum we found that GNN optimized best as in previous studies [3, 27–29] with $p = 0.4$, while $p = 0$ was the optimal value for both MEM and HLM.

The convergence plot for the three cases shows that the HLM converges the fastest, followed by the MEM network and then the GNN network. Inspection of the curves in Fig. 3 also shows that the HLM suffered a significant reduction in accuracy around epoch 2000. This may be due to numerical problems when the parameter controlling the behavior of the LM algorithm becomes too small or too large. Since we are only interested in the best weights, this fluctuation does not affect the final results. Future studies may choose to investigate the effect of allowing more sampling time to the HLM algorithm. In general we see that the HLM convergence curve contains significantly more fluctuations than both MEM and GNN. The MEM network seems more fluctuating than the GNN network. While not shown, these trends are similar for the other networks attempted with differing number of hidden nodes and activation parameters. We propose that these fluctuations allow the HLM network to sample better the weight space.

In terms of the final optimized states, as shown in Table 3, the HLM network had the lowest absolute-value error and largest PCC, both in terms of top one and for top five. This was followed by the MEM and then the GNN network. We found that adding associative memory reduces the error by approximately 2% relative to the accuracy of GNN. Adding the hybrid LM steps reduces the error of the HLM algorithm by an additional 2% relative to the MEM algorithm. In total there is approximately a 4% reduction in error from GNN to HLM.

5 Conclusions

We have studied the ability of three types of neural networks to predict the closeness of a given protein model to the native structure associated with its sequence. Our comparison between the three methods showed that HLM is the best optimized, producing the lowest error and largest PCC, and that it was the most robust in the sense that other configurations of it experienced less decline than the other methods. We find that the HLM and then the

MEM types of neural networks both undergo more fluctuations on the path to convergence. We propose that these fluctuations allow for better sampling. Overall we find it may be beneficial to treat different parts of a neural network with varied computational approaches during optimization.

Acknowledgements

This work was funded in part by a National Science Foundation grant DBI 1661391, and National Institutes of Health grants R01 GM127701 and R01 GM127701-01S1. Additional support provided by the Lilly Endowment, Inc., through its support for the Indiana University Pervasive Technology Institute, and through the National Science Foundation under Grant No. CNS-0521433.

References

1. Cheng J, Tegge AN, Baldi P (2008) Machine learning methods for protein structure prediction. *IEEE Rev Biomed Eng* 1:41–49 [PubMed: 22274898]
2. Jain P, Garibaldi JM, Hirst JD (2009) Supervised machine learning algorithms for protein structure classification. *Comput Biol Chem* 33 (3):216–223 [PubMed: 19473879]
3. Faraggi E, Kloczkowski A (2014) A global machine learning based scoring function for protein structure prediction. *Proteins Struct Funct Bioinf* 82(5):752–759
4. Lima AN, Philot EA, Trossini GHG, Scott LPB, Maltarollo VG, Honorio KM (2016) Use of machine learning approaches for novel drug discovery. *Expert Opin Drug Discov* 11 (3):225–239 [PubMed: 26814169]
5. Baldi P (2018). Deep learning in biomedical data science. *Annu Rev Biomed Data Sci* 1:181–205
6. Zhang Y, Skolnick J (2004) Scoring function for automated assessment of protein structure template quality. *Proteins Struct Funct Bioinf* 57(4):702–710
7. Zhang Y, Skolnick J (2005) Tm-align: a protein structure alignment algorithm based on the tm-score. *Nucleic Acids Res* 33 (7):2302–2309 [PubMed: 15849316]
8. Xu J, Zhang Y (2010) How significant is a protein structure similarity with tm-score=0.5? *Bioinformatics* 26(7):889–895 [PubMed: 20164152]
9. Faraggi E, Zhou Y, Kloczkowski A (2014) Accurate single-sequence prediction of solvent accessible surface area using local and global features. *Proteins Struct Funct Bioinf* 82 (11):3170–3176
10. Chakravarty S, Varadarajan R (1999) Residue depth: a novel parameter for the analysis of protein structure and stability. *Structure* 7 (7):723–732 [PubMed: 10425675]
11. Tan KP, Varadarajan R, Madhusudhan MS (2011) Depth: a web server to compute depth and predict small-molecule binding cavities in proteins. *Nucleic Acids Res* 39(suppl_2): W242–W248 [PubMed: 21576233]
12. Berman HM, Westbrook J, Feng Z, Gilliland G, Bhat TN, Weissig H, Shindyalov IN, Bourne PE (2000) The protein data bank. *Nucleic Acids Res* 28(1):235–242 [PubMed: 10592235]
13. Berman H, Henrick K, Nakamura H (2003) Announcing the worldwide protein data bank. *Nat Struct Mol Biol* 10(12):980
14. Faraggi E, Kloczkowski A (2015) Genn: a general neural network for learning tabulated data with examples from protein structure prediction In: *Artificial neural networks*. Springer, Berlin, pp 165–178
15. Elman JL (1990) Finding structure in time. *Cogn Sci* 14(2):179–211
16. Baldi P, Pollastri G, Andersen CAF, Brunak S (2000) Matching protein beta-sheet partners by feedforward and recurrent neural networks. In: *Proceedings of the 2000 conference on intelligent systems for molecular biology (ISMB00)*, La Jolla AAAI Press, Palo Alto, pp 25–36
17. Pollastri G, Baldi P, Fariselli P, Casadio R (2001) Improved prediction of the number of residue contacts in proteins by recurrent neural networks. *Bioinformatics* 17(suppl_1): S234–S242 [PubMed: 11473014]
18. Guler NF, Ubeyli ED, Guler I (2005) Recurrent neural networks employing Lyapunov exponents for EEG signals classification. *Expert Syst Appl* 29(3):506–514

19. Hagan MT, Menhaj MB (1994) Training feedforward networks with the Marquardt algorithm. *IEEE Trans Neural Netw* 5(6):989–993 [PubMed: 18267874]
20. Wilamowski BM, Iplikci S, Kaynak O, Efe MO (2001). An algorithm for fast convergence in training neural networks. In: *IJCNN'01. International joint conference on neural networks, proceedings (Cat. No. 01CH37222)*, vol 3. IEEE, Piscataway, pp 1778–1782
21. Moré JJ (1978) The Levenberg-Marquardt algorithm: implementation and theory In: *Numerical analysis*. Springer, Berlin, pp 105–116
22. Battiti R (1992) First-and second-order methods for learning: between steepest descent and newton's method. *Neural Comput* 4 (2):141–166
23. Lourakis MLA, Argyros AA (2005) Is Levenberg-Marquardt the most efficient optimization algorithm for implementing bundle adjustment? In: *Tenth IEEE international conference on computer vision (ICCV'05) volume 1*, vol 2. IEEE, Piscataway, pp 1526–1531
24. Suratgar AA, Tavakoli MB, Hoseinabadi A (2007) Modified Levenberg-Marquardt method for neural networks training. *Int J Comput Inf Eng* 1(6):1745–1747
25. Moulton J, Fidelis K, Kryshchuk A, Schwede T, Tramontano A (2016) Critical assessment of methods of protein structure prediction: progress and new directions in round xi. *Proteins Struct Funct Bioinf* 84:4–14
26. Samudrala R, Levitt M (2000) Decoys : a database of incorrect conformations to improve protein structure prediction. *Protein Sci* 9(7):1399–1401 [PubMed: 10933507]
27. Faraggi E, Xue B, Zhou Y (2009) Improving the prediction accuracy of residue solvent accessibility and real-value backbone torsion angles of proteins by guided-learning through a two-layer neural network. *Proteins Struct Funct Bioinf* 74(4):847–856
28. Faraggi E, Yang Y, Zhang S, Zhou Y (2009) Predicting continuous local structure and the effect of its substitution for secondary structure in fragment-free protein structure prediction. *Structure* 17(11):1515–1527 [PubMed: 19913486]
29. Faraggi E, Zhang T, Yang Y, Kurgan L, Zhou Y (2012) Spine x: improving protein secondary structure prediction by multistep learning coupled with prediction of solvent accessible surface area and backbone torsion angles. *J Comput Chem* 33(3):259–267 [PubMed: 22045506]

Fig. 1.
General architecture of the neural network

Heuristic Code

1. Randomize initial weights
2. While error for over-fit-protection set is decreasing
 - a. Calculate network node values
 - b. Record hidden layers node values as input (associative memory)
 - c. Apply one epoch of steepest descent back-propagation to all weights
 - d. Recalculate network node values
 - e. Record hidden layers node values as input (associative memory)
 - f. Apply one epoch of the LM algorithm to the output layer weights (W3)
 - g. Calculate new error over test set
3. Save weights if error for over-fit-protection set is reduced

Fig. 2.

Heuristic code for the algorithms studied. The full code represents the hybrid approach we propose here (HLM). Removing steps 2.d–f. we obtain a back-propagating algorithm with associative memory (MEM). Removing steps 2.d–f. and step 2.b. we obtain a standard back-propagating neural network (GNN)

Fig. 3.

Convergence plots for the error of the neural network. **(a)** Over-fit-protection set error versus epoch number. The outermost curve in green is for the MEM network, the intermediate curve in blue is for the GNN network, and the bottom curve in red is for the HLM network. **(b)** Over-fit-protection set error versus training error. The color assignments in this case are the same as in part **(a)**

Table 1

Neural network inputs

Input feature type	Number of features
typ233w ^a	936
res2res ^b	6615
cmap ^c	882
Length and mass ^d	2
Seder1 ^e	1

^aResidue delineated atom depth as described in Seder1 [3]

^bPartial sums of residue-residue distances as described in Seder1

^cResidue contact maps for cutoff distance of 5 and 7 Å

^dNumber of residues and atoms in protein chain

^eSeder1 prediction

Table 2

Neural network inputs

Item	Parameter		
	HL1 ^a	HL2 ^b	a ^c
Variation	3–21	3–21	0.01–0.17
<i>Optimal</i> ^d			
GNN	11	15	0.03
MEM	11	3	0.03
HLM	11	3	0.03

^aNumber of nodes in first hidden layer

^bNumber of nodes in second hidden layer

^cActivation parameter for a hyperbolic tangent activation function

^dValues for the best networks

Table 3

Accuracy for best and top5 networks

Method	Error ^a		PCC ^b	
	Best ^c	Top5 ^d	Best	Top5
GNN	0.100	0.103	0.854	0.845
MEM	0.098	0.101	0.849	0.845
HLM	0.096	0.097	0.854	0.848

^a Absolute-value error^b Pearson correlation coefficient^c Best network^d Average over five top performing architectures