

Representational Capacity of Deep Neural Networks: A Computing Study

Bernhard Bermeitinger, Tomas Hrycej, Siegfried Handschuh
KDIR2019, 2019-09-18, Vienna

Outline

1. Motivation
2. Attainable Representational Capacity
3. Test Problems with a Known Minimum
4. Optimization Methods
5. Experimental Setup
 - a. Network Generation
 - b. Data Generation
6. Computing Results
7. Conclusion

Motivation

- Strong revival of interest in layered neural networks
- Typical applications:
 - Computer Vision (VGG, ResNet, etc.)
 - Corpus-Based Semantics
- Common Statement:

“Deeper Networks exhibit larger Representational Capacity than Shallow Networks”

- This paper:
 - Use sufficiently determined problems
 - Use optimization methods that can be expected to converge to local minima
 - Compare shallow and deep networks with (roughly) the same number of parameters

Attainable Representational Capacity

- Usefulness of a network type consists of
 - Algorithm Efficiency
 - Representational Capacity
- For Mean-Square Minimization:
 - Numerical efficiency of optimization method is key parameter
- For Convex Minimization Problems:
 - Usual measure of (potential) efficiency is the *Condition Number* of Hessian Matrix
 - Defined as ratio of the largest and smallest Eigenvalue of Hessian
- But can't applied here:
 - Error function is not convex (at points far away from the minimum)
 - Eigenvalues are very close to zero: very small effect on search directions

Optimization Methods

- Stochastic Gradient Descent, Adadelata, RMSprop
 - Used for their widespread use
 - Implementation from Keras¹ within TensorFlow²
- Conjugate Gradient
 - Implementation from SciPy³
 - Well-defined convergence properties
 - Results little as possible subject to random disturbances
 - GPU not available, runtime not recorded

1: <https://keras.io/optimizers/>

2: <https://www.tensorflow.org/beta/>

3: <https://docs.scipy.org/doc/scipy/reference/generated/scipy.optimize.minimize.html>

Experimental Setup

- Assess relationship between attainable representational capacities of *shallow* and *deep* networks
- Comparison done by identical mapping problems and observing output of networks
- All problems deliberately defined to have minimum at zero
- Three problem sizes defined: A, B, C with each three different network layouts
- Dimensions for input/hidden/output chosen:
 - Hidden layer as *feature extractor* smaller than input/output
 - Full regression under-determined
 - Narrow hidden layer slightly over-determined
 - Overfitting excluded

Computing Results: Network Generation

- Activation function: symmetric sigmoid functions rescaled to have unity derivative at $x = 0$:

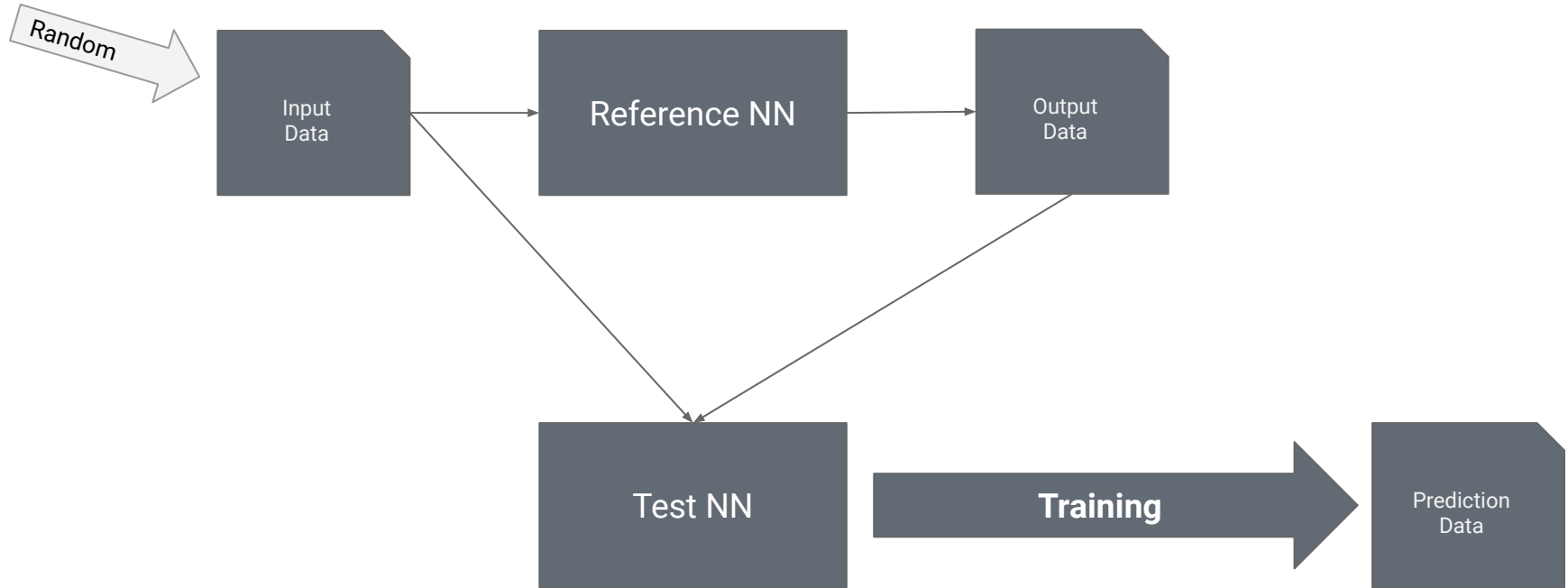
$$s(x) = \frac{1}{1 + e^{-x}}$$
$$f(x) = 2 \cdot s(2x) - 1 = \frac{2}{1 + e^{-2x}} - 1$$

- Fifteen different random parametrizations with corresponding training sets generated
 - All with a known mean-square minimum of zero

Experimental Setup: Data Generation

Problem	Input	Output	Data size	# hidden layers	# nodes	# parameters	# constraints
A_1	100	50	80	1	20	3,070	4,000
A_2	100	50	80	3	16	3,010	4,000
A_3	100	50	80	5	14	3,004	4,000
B_1	300	150	240	1	60	27,210	36,000
B_2	300	150	240	3	49	27,149	36,000
B_3	300	150	240	5	43	27,111	36,000
C_1	1000	500	800	1	200	300,700	400,000
C_2	1000	500	800	3	164	300,784	400,000
C_3	1000	500	800	5	144	300,164	400,000

Experimental Setup: Data Generation



Computing Results: For Class B

Source Network	Target Network	Optimizer	# iterations	Mean-Square-Error ($\times 10^{-3}$)
B₁	B₁	Adadelta	2,000	6.748
		RMSprop	2,000	0.098
		SGD	2,000	90.402
		CG	1,490	0.012
B₅	B₅	Adadelta	2,000	4.915
		RMSprop	2,000	0.277
		SGD	2,000	33.233
		CG	821	0.120
B₅	B₁	Adadelta	2,000	1.594
		RMSprop	2,000	0.070
		SGD	2,000	32.404
		CG	333	0.030
B₁	B₅	Adadelta	2,000	44.980
		RMSprop	2,000	9.629
		SGD	2,000	136.118
		CG	2,125	8.451

Computing Results: For Class B

Source Network	Target Network	Optimizer	# iterations	Mean-Square-Error ($\times 10^{-3}$)
B_1	B_1	Adadelata	2,000	6.748
		RMSprop	2,000	0.098
		SGD	2,000	90.402
		CG	1,490	0.012
B_5	B_5	Adadelata	2,000	4.915
		RMSprop	2,000	0.277
		SGD	2,000	33.233
		CG	821	0.120
B_5	B_1	Adadelata	2,000	1.594
		RMSprop	2,000	0.070
		SGD	2,000	32.404
		CG	333	0.030
B_1	B_5	Adadelata	2,000	44.980
		RMSprop	2,000	9.629
		SGD	2,000	136.118
		CG	2,125	8.451

Computing Results: For Class B

Source Network	Target Network	Optimizer	# iterations	Mean-Square-Error ($\times 10^{-3}$)
B_1	B_1	Adadelta	2,000	6.748
		RMSprop	2,000	0.098
		SGD	2,000	90.402
		CG	1,490	0.012
B_5	B_5	Adadelta	2,000	4.915
		RMSprop	2,000	0.277
		SGD	2,000	33.233
		CG	821	0.120
B_5	B_1	Adadelta	2,000	1.594
		RMSprop	2,000	0.070
		SGD	2,000	32.404
		CG	333	0.030
B_1	B_5	Adadelta	2,000	44.980
		RMSprop	2,000	9.629
		SGD	2,000	136.118
		CG	2,125	8.451

Computing Results: For Class B

Source Network	Target Network	Optimizer	# iterations	Mean-Square-Error ($\times 10^{-3}$)
B_1	B_1	Adadelta	2,000	6.748
		RMSprop	2,000	0.098
		SGD	2,000	90.402
		CG	1,490	0.012
B_5	B_5	Adadelta	2,000	4.915
		RMSprop	2,000	0.277
		SGD	2,000	33.233
		CG	821	0.120
B_5	B_1	Adadelta	2,000	1.594
		RMSprop	2,000	0.070
		SGD	2,000	32.404
		CG	333	0.030
B_1	B_5	Adadelta	2,000	44.980
		RMSprop	2,000	9.629
		SGD	2,000	136.118
		CG	2,125	8.451

Conclusion

- Superiority of **shallow** networks in attaining low mean-square minima
- Shallow networks have been superior:
 - For all test problems
 - For all optimizers
- Gap relatively small for SGD and Adadelta as well as for large networks
- Conjugate Gradient (theoretically) guarantees finding a minimum
 - Although the search is quadratic, so the number of iterations explodes
- No explanation why deeper networks “are more favorable”
 - It may argued that the parameter search is more of a random search