

Nonparametric Neural Networks

George Philipp, Jamie G. Carbonell¹

¹Carnegie Mellon University

ICLR, 2017

Presenter: Bargav Jayaraman

- 1 Introduction
- 2 Related Works
- 3 Nonparametric Neural Networks
- 4 Training Nonparametric Neural Networks
 - Controlling Network Size with Zero-Units
 - Self-Similar Nonlinearities
 - Capped Batch Normalization
 - Adaptive Radial-Angular Gradient Descent (AdaRad)
- 5 Experiments
 - Performance
 - Analysis of Nonparametric Training Process
 - Scalability
- 6 Conclusion

Introduction

- Problem of model selection deals with finding the best model for a given task.
- Goal of model selection: find the hyperparameter $\theta \in \Theta$ that minimizes a criterion $c(\theta)$.
- Problem: Parameter space Θ is large, thus finding optimal θ is hard.

- ① Black-box models - select a θ , test $c(\theta)$, select another θ until convergence or time over. E.g. grid search, random search, etc.
Problem: expensive, cannot alter θ during runtime.
- ② Pruning based models - Begin eliminating unnecessary weight connections from a trained model via regularization.
Problem: require a pre-trained model to begin with.

Nonparametric Neural Networks

Optimization problem of nonparametric neural network is represented as:

$$\min_{\mathbf{d}=(d)_I, d_I \in \mathbb{Z}_+, 1 \leq I \leq L-1} \min_{\mathbf{w}=(w)_I, w_I \in \mathbb{R}^{d_{I-1} * d_I}, 1 \leq I \leq L} \frac{1}{|D|} \sum_{(x,y) \in D} e(f(\mathbf{w}, x), y) + \Omega(\mathbf{w})$$

d_0 and d_L are fixed because input data and error function e are fixed.
Parameters form the pair (\mathbf{d}, \mathbf{w}) .

Fan-in and fan-out regularizers are defined as:

$$\Omega_{in}(\mathbf{w}, \lambda, p) = \lambda \sum_{l=1}^L \sum_{j=1}^{d_l} \|[W_l(1,j), W_l(2,j), \dots, W_l(d_{l-1},j)]\|_p$$

$$\Omega_{out}(\mathbf{w}, \lambda, p) = \lambda \sum_{l=1}^L \sum_{i=1}^{d_{l-1}} \|[W_l(i,1), W_l(i,2), \dots, W_l(i,d_l)]\|_p$$

Outline

- 1 Introduction
- 2 Related Works
- 3 Nonparametric Neural Networks
- 4 **Training Nonparametric Neural Networks**
 - **Controlling Network Size with Zero-Units**
 - Self-Similar Nonlinearities
 - Capped Batch Normalization
 - Adaptive Radial-Angular Gradient Descent (AdaRad)
- 5 Experiments
 - Performance
 - Analysis of Nonparametric Training Process
 - Scalability
- 6 Conclusion

Controlling Network Size with Zero-Units

- Zero units are units with either fan-in or fan-out or both as zero vectors.
 - generated by fan-in or fan-out regularization.
- f-equivalence defines the notion of similarity between two network architectures $(\mathbf{d}_1, \mathbf{W}_1)$ and $(\mathbf{d}_2, \mathbf{W}_2)$: $f(\mathbf{W}_1, x) = f(\mathbf{W}_2, x)$, where not necessarily $\mathbf{d}_1 = \mathbf{d}_2$.
- Adding or removing zero-units preserves f-equivalence. (provided we use non-linearity function σ such that $\sigma(0) = 0$ - e.g. ReLu)

Outline

- 1 Introduction
- 2 Related Works
- 3 Nonparametric Neural Networks
- 4 Training Nonparametric Neural Networks**
 - Controlling Network Size with Zero-Units
 - **Self-Similar Nonlinearities**
 - Capped Batch Normalization
 - Adaptive Radial-Angular Gradient Descent (AdaRad)
- 5 Experiments
 - Performance
 - Analysis of Nonparametric Training Process
 - Scalability
- 6 Conclusion

Self-Similar Nonlinearities

- Self-similar nonlinearities are invariant to scaling, i.e., they satisfy $\sigma(cs) = c\sigma(s)$, $\forall c \in \mathbb{R}_{\geq 0}, s \in \mathbb{R}$ E.g. ReLu.
- Self-similar nonlinearities are required because of the usage of fan-in and fan-out regularization that shrink (or rescale) the weights.

Proposition

If all nonlinearities in a nonparametric network model except possible σ_L are self-similar, then the objective function using a fan-in or fan-out regularizer with different regularization parameters $\lambda_1, \dots, \lambda_L$ for each layer is equivalent to the same objective function using the single regularization parameter $\lambda = (\prod_{l=1}^L \lambda_l)^{\frac{1}{L}}$ for each layer, up to rescaling of weights.

Outline

- 1 Introduction
- 2 Related Works
- 3 Nonparametric Neural Networks
- 4 Training Nonparametric Neural Networks**
 - Controlling Network Size with Zero-Units
 - Self-Similar Nonlinearities
 - Capped Batch Normalization**
 - Adaptive Radial-Angular Gradient Descent (AdaRad)
- 5 Experiments
 - Performance
 - Analysis of Nonparametric Training Process
 - Scalability
- 6 Conclusion

Capped Batch Normalization

- Batch normalization cannot be applied directly to nonparametric neural network as it negates the effect of regularization - since fan-in or fan-out regularizer will try to shrink the weights arbitrarily while compensating the batch normalization layer.
- Capped Batch Normalization (CapNorm) is introduced for compatibility with regularization.
- CapNorm replaces each pre-activation z with $\frac{z-\mu}{\max(\sigma,1)}$, where μ is mean and σ is standard deviation of a unit's pre-activations across the current mini-batch.

Outline

- 1 Introduction
- 2 Related Works
- 3 Nonparametric Neural Networks
- 4 Training Nonparametric Neural Networks**
 - Controlling Network Size with Zero-Units
 - Self-Similar Nonlinearities
 - Capped Batch Normalization
 - Adaptive Radial-Angular Gradient Descent (AdaRad)
- 5 Experiments
 - Performance
 - Analysis of Nonparametric Training Process
 - Scalability
- 6 Conclusion

Adaptive Radial-Angular Gradient Descent (AdaRad)

```
1  input:  $\alpha_r$ : radial step size;  $\alpha_\phi$ : angular step size;  $\lambda$ : regularization hyperparameter;  $\beta$ : mixing
    rate;  $\epsilon$ : numerical stabilizer;  $\mathbf{d}^0$ : initial dimensions;  $\mathbf{W}^0$ : initial weights;  $\nu$ : unit addition
    rate;  $\nu_{\text{freq}}$ : unit addition frequency;  $T$ : number of iterations
2   $\phi_{\text{max}} = 0$ ;  $c_{\text{max}} = 0$ ;  $\mathbf{d} = \mathbf{d}^0$ ;  $\mathbf{W} = \mathbf{W}^0$ ;
3  for  $l = 1$  to  $L$  do
4      set  $\tilde{\phi}_l$  (angular quadratic running average) and  $c_l$  (angular quadratic running average capacity)
      to zero vectors of size  $d_l^0$ ;
5  end
6  for  $t = 1$  to  $T$  do
7      set  $D^t$  to mini-batch used at iteration  $t$ ;
8       $\mathbf{G} = \frac{1}{|D|} \nabla_{\mathbf{W}} \sum_{(x,y) \in D^t} e(f(\mathbf{W}, x), y)$ ;
9      for  $l = L$  to 1 do
10         for  $j = d_l$  to 1 do
11             decompose  $[G_l(i, j)]_i$  into a component parallel to  $[W_l(i, j)]_i$  (call it  $r$ ) and a
                component orthogonal to  $[W_l(i, j)]_i$  (call it  $\phi$ ) such that  $[G_l(i, j)]_i = r + \phi$ ;
12              $\tilde{\phi}_l(j) = (1 - \beta)\tilde{\phi}_l(j) + \beta\|\phi\|_2^2$ ;  $c_l(j) = (1 - \beta)c_l(j) + \beta$ ;
13              $\phi_{\text{max}} = \max(\phi_{\text{max}}, \tilde{\phi}_l(j))$ ;  $c_{\text{max}} = \max(c_{\text{max}}, c_l(j))$ ;
14              $\phi_{\text{adj}} = \frac{\sqrt{\frac{\phi_{\text{max}}}{c_{\text{max}}}}}{\sqrt{\frac{\tilde{\phi}_l(j)}{c_l(j)} + \epsilon}} \phi$ ;
15              $[W_l(i, j)]_i = [W_l(i, j)]_i - \alpha_r r$ ;
16             rotate  $[W_l(i, j)]_i$  by angle  $\alpha_\phi \|\phi_{\text{adj}}\|_2$  in direction  $-\frac{\phi_{\text{adj}}}{\|\phi_{\text{adj}}\|_2}$ ;
17             shrink( $[W_l(i, j)]_i$ ,  $\alpha_r \lambda \frac{|D^t|}{|D|}$ );
18             if  $l < L$  and  $[W_l(i, j)]_i$  is a zero vector then
19                 remove column  $j$  from  $W_l$ ; remove row  $j$  from  $W_{l+1}$ ; remove element  $j$  from  $\tilde{\phi}_l$ 
                and  $c_l$ ; decrement  $d_l$ ;
20             end
21         end
22         if  $t = 0 \bmod \nu_{\text{freq}}$  then
23              $\nu' = \nu$ ; // if  $\nu \notin \mathbb{Z}$ , we can set e.g.  $\nu' = \text{Poisson}(\nu)$ 
24             add  $\nu'$  randomly initialized columns to  $W_l$ ; add  $\nu'$  zero rows to  $W_{l+1}$ ; add  $\nu'$  zero
                elements to  $\tilde{\phi}_l$  and  $c_l$ ;  $d_l = d_l + \nu'$ ;
25         end
26     end
27 end
28 return  $\mathbf{W}$ ;
```

Outline

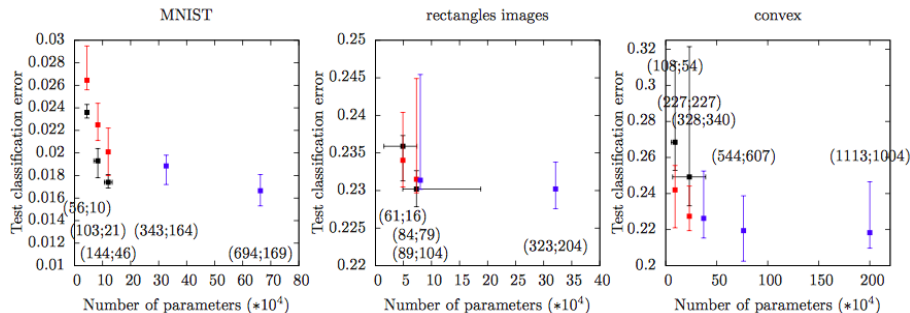
- 1 Introduction
- 2 Related Works
- 3 Nonparametric Neural Networks
- 4 Training Nonparametric Neural Networks
 - Controlling Network Size with Zero-Units
 - Self-Similar Nonlinearities
 - Capped Batch Normalization
 - Adaptive Radial-Angular Gradient Descent (AdaRad)
- 5 Experiments
 - Performance
 - Analysis of Nonparametric Training Process
 - Scalability
- 6 Conclusion

Performance

$\alpha_\phi = 30$, repeatedly divided by 3 when validation error stops improving.

$$\alpha_r = \frac{1}{50\lambda}.$$

λ values are 3×10^{-3} , 10^{-3} and 3×10^{-4} for MNIST, 3×10^{-5} and 10^{-6} for rectangles images and 10^{-8} for convex.

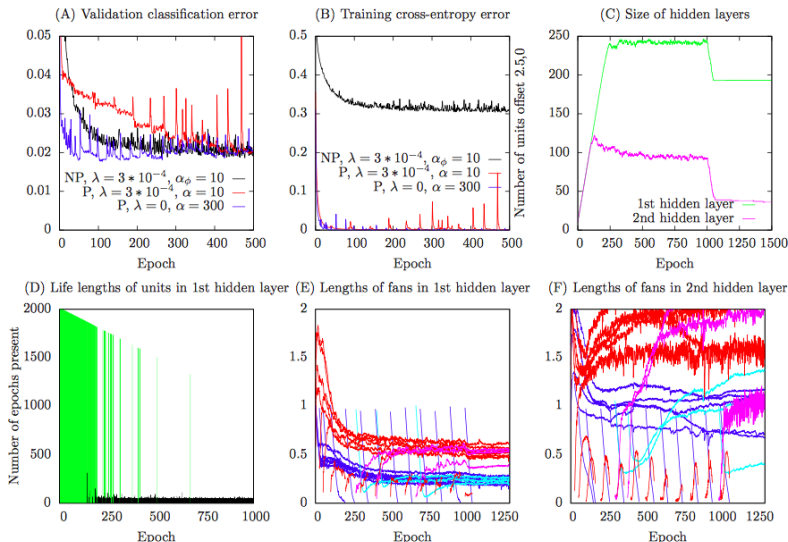


Outline

- 1 Introduction
- 2 Related Works
- 3 Nonparametric Neural Networks
- 4 Training Nonparametric Neural Networks
 - Controlling Network Size with Zero-Units
 - Self-Similar Nonlinearities
 - Capped Batch Normalization
 - Adaptive Radial-Angular Gradient Descent (AdaRad)
- 5 Experiments
 - Performance
 - Analysis of Nonparametric Training Process
 - Scalability
- 6 Conclusion

Analysis of Nonparametric Training Process

$\alpha_\phi = 10$, $\lambda = 3 * 10^{-4}$. Final model has 193 X 36 units on MNIST.



Outline

- 1 Introduction
- 2 Related Works
- 3 Nonparametric Neural Networks
- 4 Training Nonparametric Neural Networks
 - Controlling Network Size with Zero-Units
 - Self-Similar Nonlinearities
 - Capped Batch Normalization
 - Adaptive Radial-Angular Gradient Descent (AdaRad)
- 5 Experiments
 - Performance
 - Analysis of Nonparametric Training Process
 - Scalability
- 6 Conclusion

4 hidden layers instead of 2, $\alpha_r = \frac{1}{5\lambda}$, adding new units every 10th epoch.

Table 2: Test classification error of various models trained on the *poker* dataset.

Algorithm	λ	Starting net size	Final net size	Error
Logistic regression (ours)				49.9%
Naïve bayes (OpenML)				48.3%
Decision tree (OpenML)				26.8%
Nonparametric net	10^{-3}	10-10-10-10	23-24-15-4	0.62%
	10^{-5}	10-10-10-10	94-135-105-35	0.022%
	10^{-6}	10-10-10-10	210-251-224-104	0.001%
	10^{-7}	10-10-10-10	299-258-259-129	0%
		23-24-15-4	unchanged	0.20%
Parametric net		94-135-105-35	unchanged	0.003%
		210-251-224-104	unchanged	0.003%
		299-258-259-129	unchanged	0.002%

Conclusion

- ① Nonparametric neural network is proposed which automatically learns the optimal network structure.
- ② Experimental results supporting that nonparametric neural networks outperform parametric neural networks (in most of the cases) under the same settings of network size.
- ③ Theoretical soundness of the framework is provided.