Learning Nonlinear Feature Interactions in the Data Starved Regime

Adrian Barbu

Florida State University

March 28, 2021

Joint work with Yangzi Guo

Introduction

- In many problems, the predictors (features) interact in complex ways in relation to the response.
- Moreover, only a small number of features are usually relevant for the response
- Examples: Object or action recognition from images, SNPs relation with certain diseases, etc.
- In such cases, we need to go beyond linear models

Introduction

- Artificial neural networks (ANNs) can model complex nonlinear interactions.
- However, they need lots of training data.
- What happens in the data starved regime?

Introduction

- Artificial neural networks (ANNs) can model complex nonlinear interactions.
- However, they need lots of training data.
- What happens in the data starved regime?
- The loss function has many local minima
- The number of local minima grows exponentially with the number of irrelevant variables.
- Finding a deep local minimum results in good generalization even in this case

Introduction

Contributions:

- An empirical study of the learnability of neural networks (NNs) on non-linear, XOR-based data with irrelevant variables.
- Extensive experiments on the number of local minima, and their relation with the number of irrelevant variables.
- A framework for node and feature selection to improve the capability of the ANNs to find a deep local minimum.
- Experiments confirm that our method helps improve generalization on the XOR-like data and several real datasets.

Related Work

- Draxler et al. (2018) and Garipov et al. (2018) showed that the local minima of some convolutional neural networks are equivalent (have the same loss value) and a equi-energy path can be found between the local minima.
- Soudry and Carmon (2016) proved that all differentiable local minima are global minima for the one hidden layer ANNs with piecewise linear activation and square loss.
- The Lottery Tickets Hypothesis (Frankle and Carbin, 2019): a random initialized dense neural network contains a sub-network that if trained in isolation, initialized with the original parameters, will obtain the same test accuracy as the original network.

The Noisy XOR Data

Problem Statement

- We look at an extreme case, the noisy exclusive-OR (XOR) classification problem.
- The k-dimensional XOR is a binary classification problem:

$$y(\mathbf{x}) = \begin{cases} +1 & \text{if } \prod_{i=1}^{k} x_i < 0 \\ -1 & \text{else} \end{cases} \tag{1}$$

  where $\mathbf{x} \in \mathbb{R}^p$ is sampled uniformly from $[-1, +1]^p$.
- We call this data the k-D XOR in p dimensions, where $k \leq p$.
- In this work, we will use $k \in \{3, 4, 5\}$.

- The XOR data can only be modeled by using higher order feature interactions.

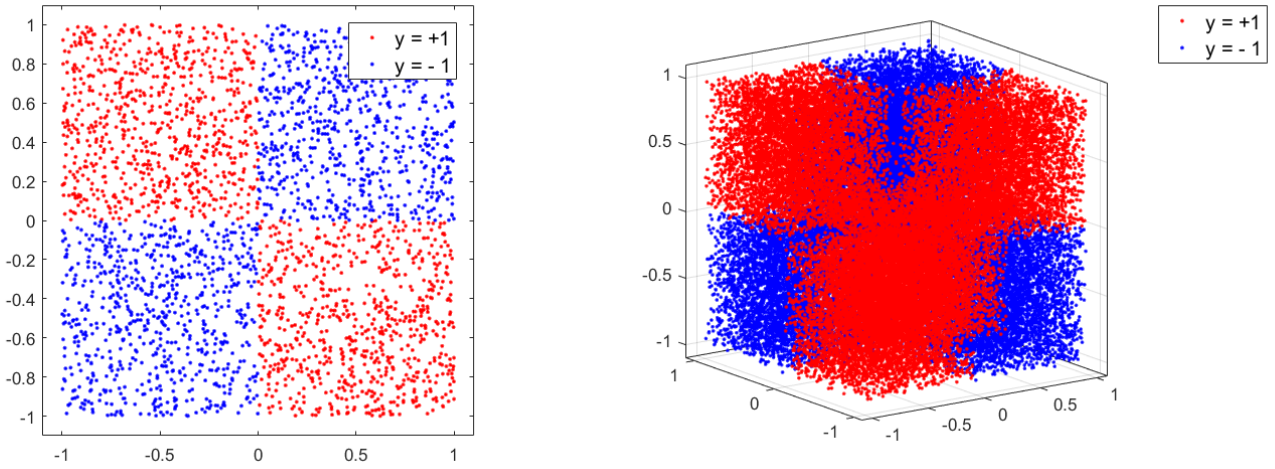
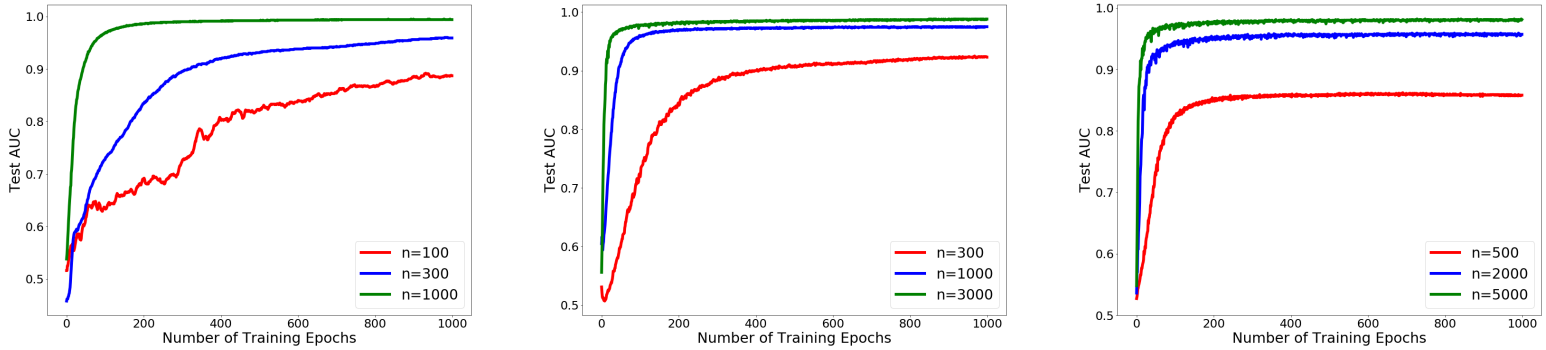


Figure: 2D and 3D XOR data. Left: 2D XOR with $p = 2$. Right: 3D XOR with $p = 3$.

Neural Networks on the XOR Data

- Neural network: fully connected, one hidden layer with h nodes, ReLU activation, logistic loss
- Can handle the non-noisy XOR data ($p = k \in \{3, 4, 5\}$) well with sufficient samples and hidden nodes.



$p = k = 3, h = 20$ $p = k = 4, h = 100$ $p = k = 5, h = 500$

Figure: Test AUC for non-noisy XOR data.

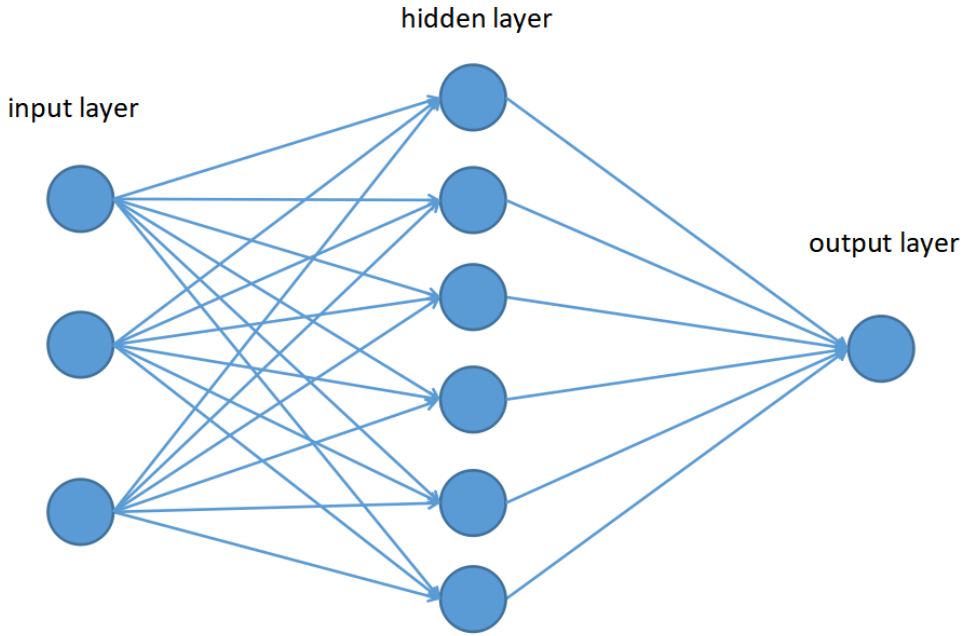


Figure: One hidden layer neural network.

The input layer is not a real layer, it contains the input variables that feed into the NN.

Mathematical Formulation

- We will work on XOR data with many irrelevant features.
- The hidden node weights are vectors $\mathbf{w}_j = (w_{j1}, ..., w_{jp}, w_{jp+1})^T \in \mathbb{R}^{p+1}$, $j = 1, ..., h$.
- The output neuron has weight vector $\boldsymbol{\beta} = (\beta_1, ..., \beta_h)^T \in \mathbb{R}^h$, and bias $\beta_0 \in \mathbb{R}$.
- Using ReLU activation $\sigma(z) = \max(0, z)$, the neural network is:

$$f(\mathbf{x}) = \sum_{j=1}^{h} \beta_j \sigma(\mathbf{w}_j^T \mathbf{x}) + \beta_0 = \begin{cases} > 0 & \text{predict } +1 \\ < 0 & \text{predict } -1 \end{cases} \quad (2)$$

- Unconstrained optimization with the logistic loss:

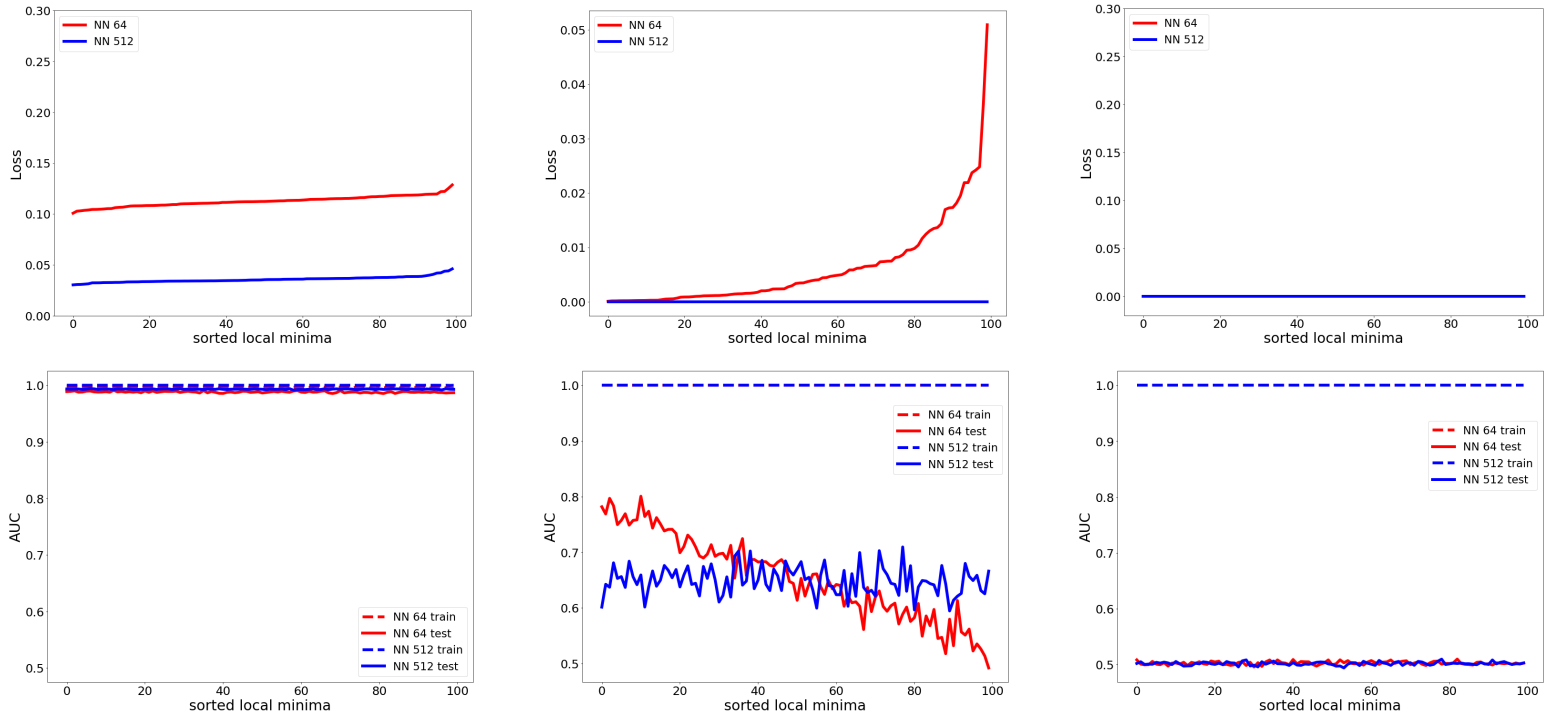
$$\min_{\mathbf{w},\boldsymbol{\beta},\beta_0} L(\mathbf{w},\boldsymbol{\beta},\beta_0)$$

$$L(\mathbf{w},\boldsymbol{\beta},\beta_0) = \sum_{i=1}^{n} \log\left(1+\exp\left\{-y_i \sum_{j=1}^{h} \beta_j \sigma(\mathbf{w}_j^T \mathbf{x}_i) + \beta_0\right\}\right), \tag{3}$$

  where $(\mathbf{x}_i, y_i), i = 1, ..., n$ are the training examples.

- We use stochastic gradient descent (SGD) based optimizers via backpropagation (Werbos, 1974) to minimize the loss in an iterative way.

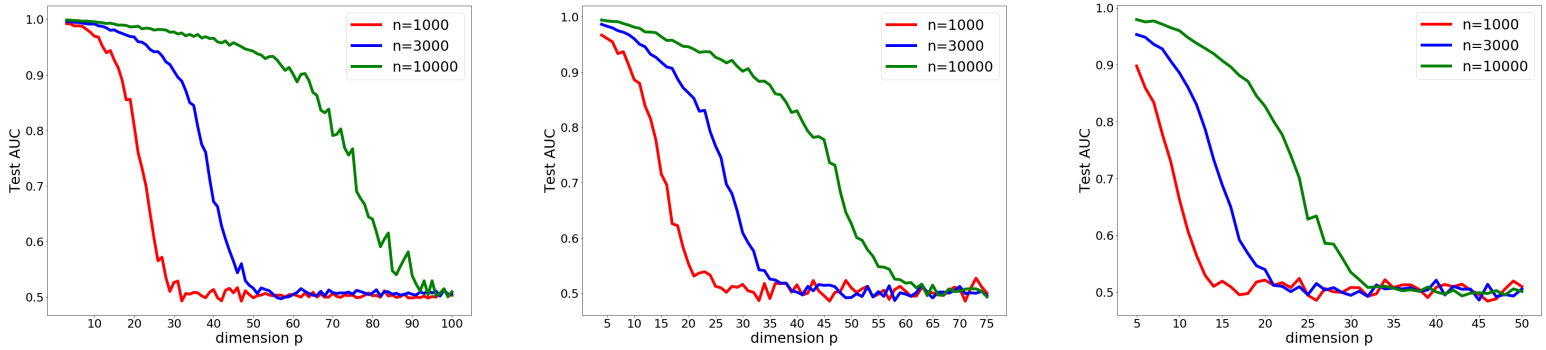
- We ran NNs with 100 random initializations, and report the sorted local minima and corresponding test AUC.
- Deeper minima have better generalization in the presence of irrelevant variables, .



$n = 3000, p = 4$ $n = 3000, p = 27$ $n = 3000, p = 100$

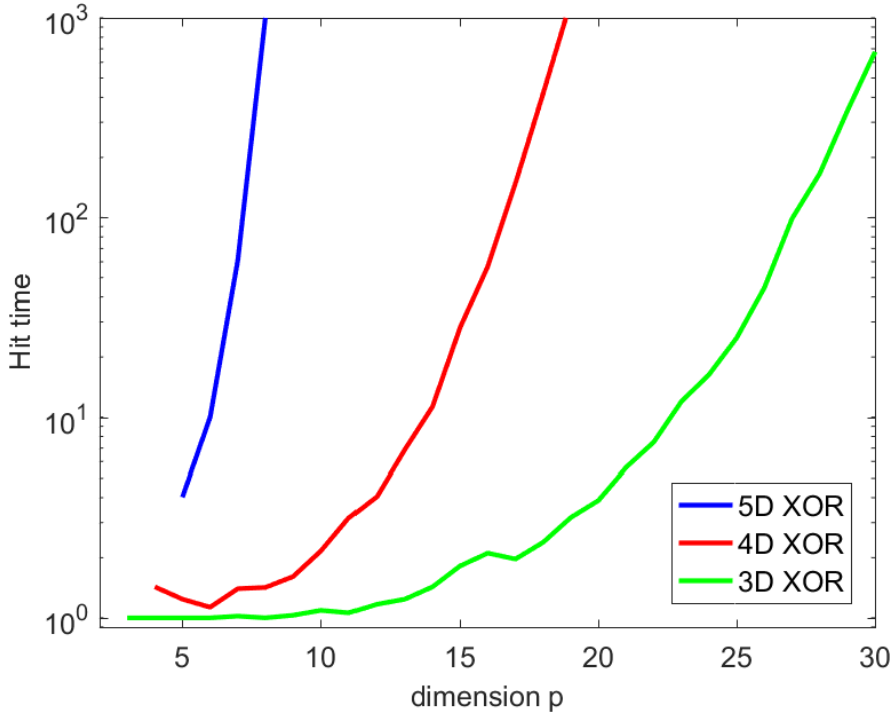
Figure: Values of sorted local minima (top) and AUC (bottom) for 4D XOR

- NN with $h = 512$ hidden nodes
- Keep smallest loss solution out of 10 random initializations.
- Test AUC vs p, averaged over 10 independent runs.
- Deepest local minimum out of 10 initializations only works up to some p



$k = 3$ $k = 4$ $k = 5$

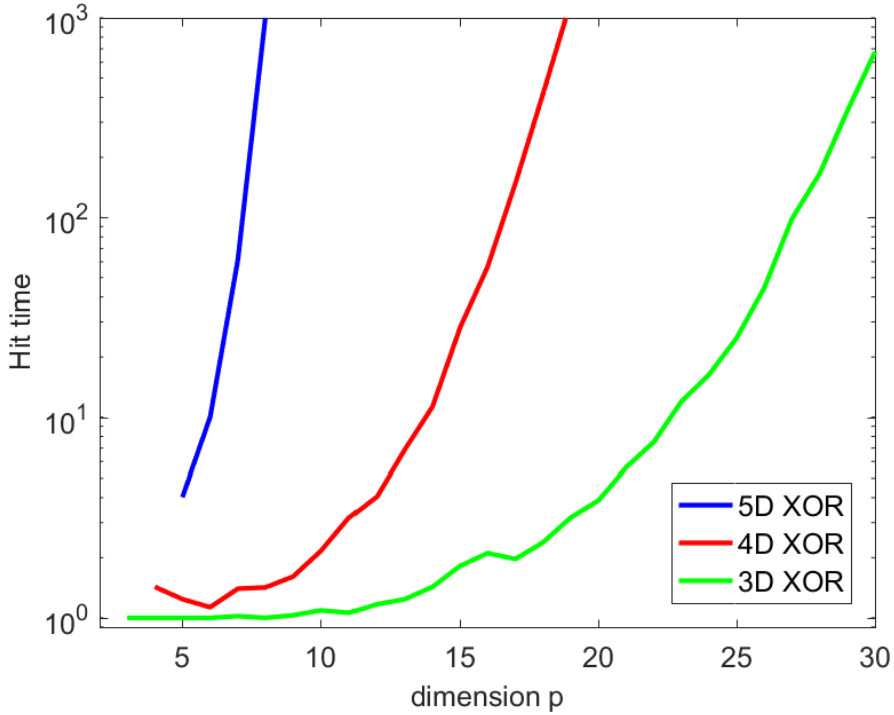
Figure: Test AUC of best solution out of 10 random initializations vs. data dimension p for a NN with $h = 512$ hidden nodes.



- NN with $h = 20$ hidden nodes, $n = 3000$.
- Hit time: number of random training initializations until one local minimum has train AUC ≥ 0.95.
- Hit time vs p, averaged over 10 independent runs.

Feature Interactions with Neural Networks

NN Generalization vs Number of Irrelevant Variables



- NN with $h = 20$ hidden nodes, $n = 3000$.
- Hit time: number of random training initializations until one local minimum has train AUC ≥ 0.95.
- Hit time vs p, averaged over 10 independent runs.
- Number of tries (local optima) grows super-exponentially with p.

Conclusions from the above study:

- If the training data is difficult (such as the XOR data), not all local minima are equivalent.
- For a fixed training size n, the number of shallow local minima quickly blows up as the number of irrelevant variables increases and finding the deep local minima becomes extremely hard.
- If the number of irrelevant variables is not too large, an NN with a sufficiently many hidden nodes will find a deep optimum more often, but does not generalize.

These observations form the basis for the proposed node and feature selection methodology presented next.

Node Selection with Annealing

- We introduce a node selection method for training NNs, which can avoid many local optima.
- We start with a large model with many hidden nodes and gradually remove neurons to obtain a compact network.

$$f(\mathbf{x}) = \sum_{j=1}^{h} \beta_j \sigma(\mathbf{w}_j^T \mathbf{x}) + \beta_0 = \sum_{j=1}^{h} \beta_j a_j + \beta_0 = \boldsymbol{\beta}^T \cdot \mathbf{a} + \beta_0, \quad (4)$$

- We gradually drop the hidden nodes based on the magnitude of the associated weights $|\beta_j|$ of the output neuron during the training
- We only keep a few relevant hidden nodes at the end.
- The number of kept nodes at iteration e is:

$$M_e = \begin{cases} p & 1 \leq e \leq N^{pretrain} \\ k + (p-k)\max\left(0, \frac{(N-N^{pretrain})-2e}{2e\mu+(N-N^{pretrain})}\right) & N^{pretrain} < e \leq N \end{cases} \quad (5)$$

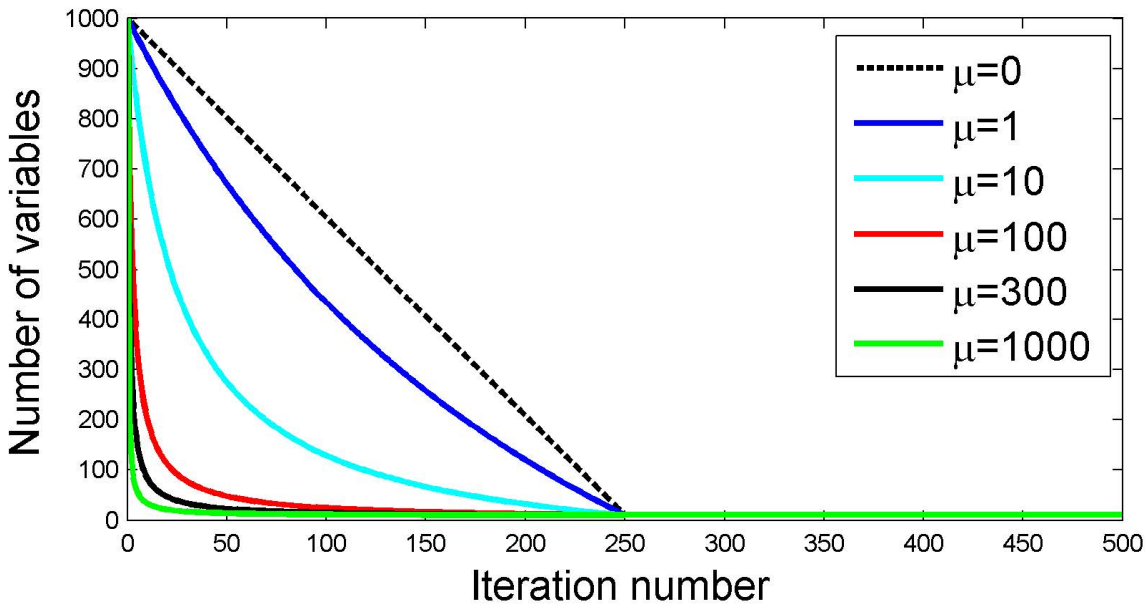


Figure: The number of kept features M_e vs iteration e for different schedules with $p = 1000$, $k = 10$, $N = 500$. Figure source from Barbu et al. (2017).

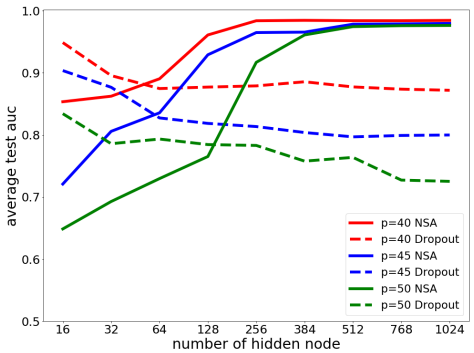
Algorithm Node Selection with Annealing (NSA)

Input: Training set $T = \{(\mathbf{x}_i, y_i) \in \mathbb{R}^p \times \mathbb{R}\}_{i=1}^n$, desired number h of hidden neurons, starting number H of hidden neurons, annealing schedule $M_e, e = 1, .., N^{iter}$.

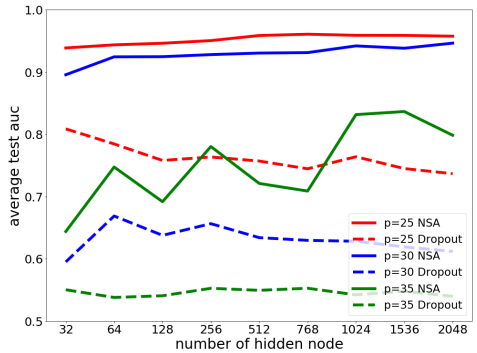
Output: Trained NN with h hidden neurons.

1: Initialize a NN with H hidden neurons with random initialization
2: **for** $e = 1$ to N^{iter} **do**
3: Update $\mathbf{w}$, $\boldsymbol{\beta}$ and β_0 via backpropagation with a gradient descent based optimizer
4: Remove hidden nodes to keep the M_e nodes with largest $|\beta_j|$
5: **end for**

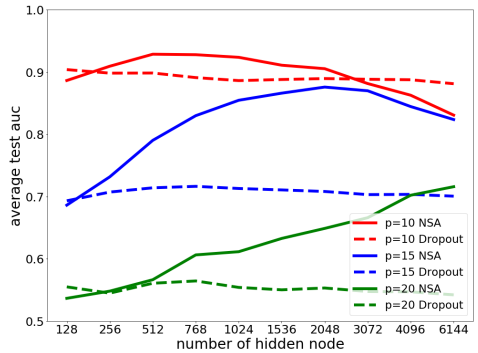
- We compare NSA with Dropout (Hinton et al., 2012) for different number of hidden nodes h.
- For NN+NSA, we finally keep $h = 8$ hidden nodes for $k = 3$, $h = 16$ for $k = 4$ and $h = 64$ for $k = 5$ at the end of training.
- We show the average test AUC for NN+NSA and NN+Dropout vs initial number of hidden nodes H.



$k = 3, p = \{40, 45, 50\}$



$k = 4, p = \{25, 30, 35\}$



$k = 5, p = \{10, 15, 20\}$

Figure: Average test AUC vs number of hidden nodes H for NNs with NSA or Dropout.

Node Importance and Normalization

- The importance information of a hidden node lies in two parts:

$$f(\mathbf{x}) = \sum_{j=1}^{h} \left(\underbrace{\beta_j}_{I} \cdot \underbrace{\sigma(\mathbf{w}_j^T \mathbf{x})}_{II} \right) + \beta_0$$

- The first part I is the weight β we considered as the node importance measure.
- In fact the second part II will also change value during backpropagation.
- The second part also carries importance information about the hidden neurons for dropping consideration.

- The XOR data is uniformly distributed in range $[-1, +1]^p$. The norm of the inner product for j-th hidden node as:

$$\begin{aligned} ||\mathbf{w}_j^T \mathbf{x}|| &= ||\mathbf{w}_j|| \cdot ||\mathbf{x}|| \cdot \underbrace{\cos\theta}_{\leq 1} \\ &\leq ||\mathbf{w}_j|| \cdot \underbrace{||\mathbf{x}||}_{\leq 1} \\ &\leq ||\mathbf{w}_j|| \end{aligned}$$

  where θ is the angle between vectors $\mathbf{w}_j$ and $\mathbf{x}$.

- The range of a hidden node activation is determined by the magnitude of the internal weight vector $\mathbf{w}$.

- We can simultaneously incorporate the importance information and rescale the activation of hidden nodes.
- We only need to transform the NNs score function, due to the usage of **ReLU** as activation

$$
\begin{aligned}
f(\mathbf{x}) &= \sum_{j=1}^{h} \beta_j \sigma(\mathbf{w}_j^T \mathbf{x}) + \beta_0 \\
&= \sum_{j=1}^{h} \beta_j \cdot \|\mathbf{w}_j\|_2 \cdot \frac{1}{\|\mathbf{w}_j\|_2} \cdot \sigma(\mathbf{w}_j^T \mathbf{x}) + \beta_0 \\
&= \sum_{j=1}^{h} \underbrace{(\beta_j \cdot \|\mathbf{w}_j\|_2)}_{incorporation} \cdot \underbrace{\left(\max\left(0, \frac{\mathbf{w}_j^T \mathbf{x}}{\|\mathbf{w}_j\|_2}\right)\right)}_{normalization} + \beta_0 \\
&= \sum_{j=1}^{h} \tilde{\beta}_j \sigma(\tilde{\mathbf{w}}_j^T \mathbf{x}) + \beta_0
\end{aligned}
$$

Algorithm Node Selection with Normalization and Annealing (NSNA)

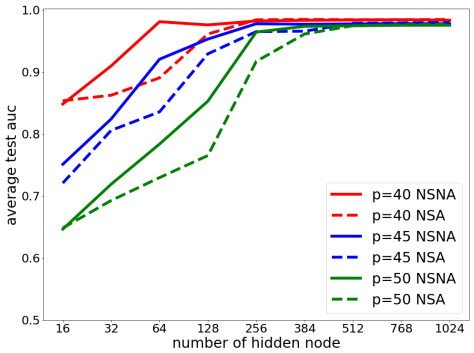
Input: Training set $T = \{(\mathbf{x}_i, y_i) \in \mathbb{R}^p \times \mathbb{R}\}_{i=1}^n$, desired number h of hidden neurons, starting number H of hidden neurons, annealing schedule $M_e, e = 1, .., N^{iter}$.

Output: Trained NN with h hidden neurons.

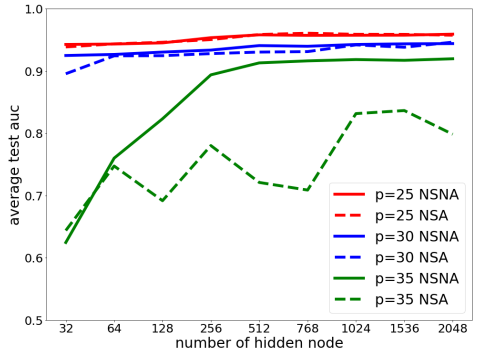
1: Initialize a NN with H hidden neurons with random initialization
2: **for** $e = 1$ to N^{iter} **do**
3: Update $\mathbf{w}$, β and β_0 via backpropagation with a gradient descent based optimizer
4: Normalize hidden nodes and incorporate the normalizers to β_j:

$$\tilde{\beta}_j \leftarrow \|\mathbf{w}_j\|\beta_j, \tilde{\mathbf{w}}_j \leftarrow \frac{\mathbf{w}_j}{\|\mathbf{w}_j\|}, j = 1, ..., h \tag{6}$$

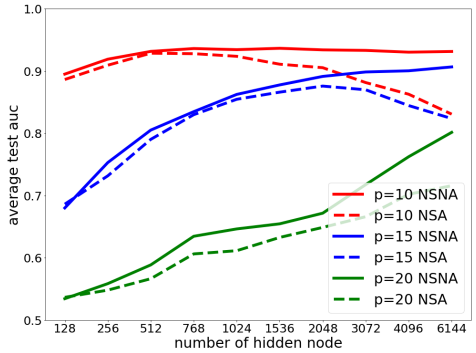
5: Remove hidden nodes to keep the M_e nodes with largest $|\tilde{\beta}_j|$
6: **end for**



$k = 3, p = \{40, 45, 50\}$

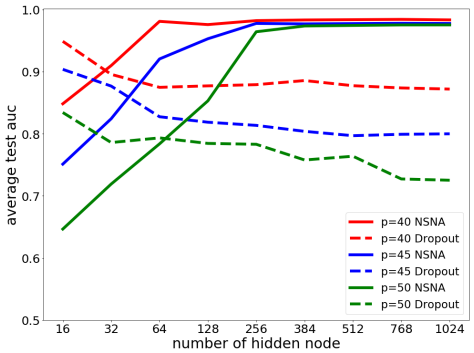


$k = 4, p = \{25, 30, 35\}$

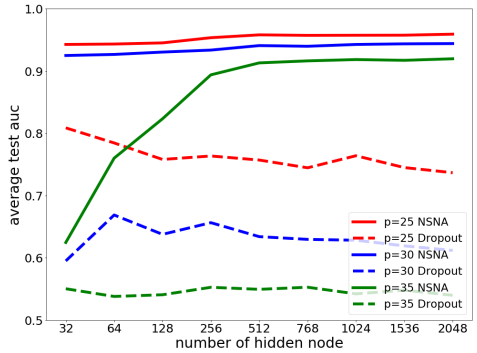


$k = 5, p = \{10, 15, 20\}$

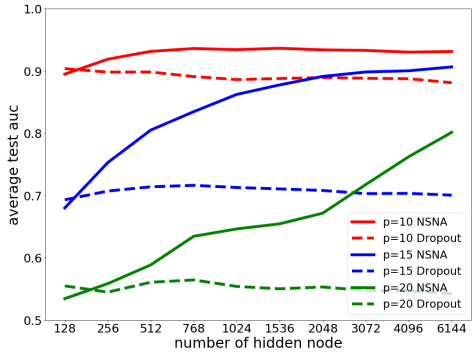
Figure: Average test AUC vs number of hidden nodes for NNs with NSNA or NSA.



$k = 3, p = \{40, 45, 50\}$



$k = 4, p = \{25, 30, 35\}$



$k = 5, p = \{10, 15, 20\}$

Figure: Average test AUC vs number of hidden nodes for NNs with NSNA or Dropout.

Experiments - Parity Data

- The parity data is a classical problem in computational learning theory (Zhang et al., 2017).
- It has the same labels as the XOR data but each variable in $\mathbf{x}$ is uniformly drawn from $\{-1, +1\}$.
- We follow Zhang et al. (2017) to generate the data labels: 10% of data will have the opposite labels

$$y = \begin{cases} x_{i_1} x_{i_2} ... x_{i_k} & \text{with probablity } 0.9 \\ -x_{i_1} x_{i_2} ... x_{i_k} & \text{with probablity } 0.1 \end{cases}$$

- The perfect classifier would have a prediction error of 0.1.
- Parity data is frequently used to test different optimizers and regularization techniques for NNs.

Experiments - Parity Data

- We perform the experiment in $p = 50$ dimensional data with parities $k = 5$.
- The training set, validation set, and testing set contain respectively 15,000, 5,000 and 5,000 data points.
- We train a one hidden layer NN with default SGD or Adam (Kingma and Ba, 2014) optimizer
- Compare with BoostNet (Zhang et al., 2017) with various number of hidden neurons $h \in [1, 100]$.
- We train a one hidden layer NN with Adam+NSNA starting with $H = 256$ hidden nodes, and down to a hidden node number $h \in [1, 16]$ using annealing schedule M_e.
- We report the best result out from 10 independent random initializations.

Experiments - Parity Data

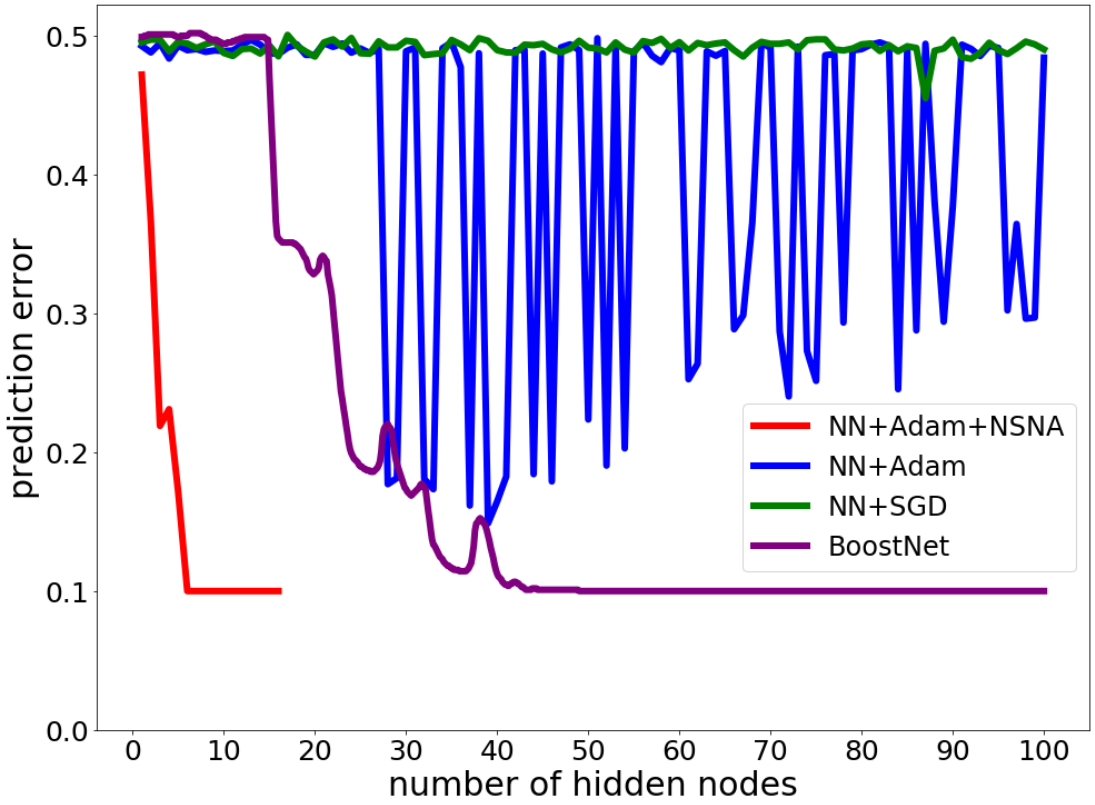


Figure: Test error vs number of hidden nodes.

Experiments - Real Data

- We compare a fully connected NN and the compact NN obtained by FSA+NSNA on real datasets.
- The real datasets were selected from the UCI ML repository (Dua and Graff, 2017).
- We ensure that the dataset is not too large and a one hidden layer fully connected NN can generalize reasonably.

Dataset	Number of classes	Number of features	Number of observations
Car Evaluation	4	21	1728
Image Segmentation	7	19	2310
Optical Recognition of Handwritten Digits	10	64	5620
Multiple Features	10	216	2000
ISOLET	26	617	7797

Table: Datasets used for evaluating the performance of fully connected NN and sparse NN with FSA+NSNA.

Experiments - Real Data

- We split all real datasets into a training and test set with a ratio $4:1$.
- The obtained training dataset will be used in a 10-run averaged 5-fold cross-validation grid search to find the best hyper-parameter settings of our trained NNs.
- We use the best hyper-parameter to retrain the NNs with the entire training dataset 10 different times, and each time we record the best test accuracy.

Experiments - Real Data

- This procedure is used for the fully connected NN, and the NN with FSA+NSNA with different sparsity levels.
- An "equivalent" fully connected NN with roughly the same number of connections as the best sparse neural network we get from FSA+NSNA is also trained.
- The hidden node number, L_2 regularization coefficient and mini-batch size were searched in $\{16, 32, 64, 128, 256, 512\}$, $\{0.0001, 0.001, 0.01, 0.1\}$, and $\{16, 32, 64\}$ respectively.

Experiments - Real Data

	NN(best)	NN(equivalent)	NN+FSA+NSNA
Car Evaluation, $p = 21, n = 1728$, 4 classes.			
Number of weights (nodes)	1600 (64)	150 (6)	120+32 = 152
Test Accuracy	**100.0±0.00**	98.23±0.06	**100.0±0.00**
Image Segmentation, $p = 19, n = 2310$, 7 classes.			
Number of weights (nodes)	6656 (256)	364 (14)	266+98 = 364
Test Accuracy	96.87±0.72	96.27±0.58	**98.40±0.32**
Optical Recognition of Handwritten Digits, $p = 64, n = 5620$, 10 classes.			
Number of weights (nodes)	37888 (512)	1998 (27)	1792+160 = 1952
Test Accuracy	98.80±0.29	98.25±0.19	**99.01±0.20**
Multiple Features, $p = 216, n = 2000$, 10 classes.			
Number of weights (nodes)	14464 (64)	904 (4)	583+320 = 903
Test Accuracy	97.85±0.80	95.45±0.98	**98.15±0.82**
ISOLET, $p = 617, n = 7797$, 26 classes.			
Number of weights (nodes)	41152 (64)	5787 (9)	4683+1118 = 5801
Test Accuracy	96.73±0.50	94.31±0.61	**96.91±0.54**

Table: Performance results of NN(best), NN(equivalent) and NN+FSA+NSNA for each dataset.

Conclusions

- A study of the number of local optima for training NNs in the data starved regime, with irrelevant features
- A node selection method for training a neural network to find deep local optima, starting with a model with many hidden neurons and gradually removing the weaker ones.
- A neuron normalization technique to better measure node importance during the dropping procedure.
- Experiments show that the proposed approach on a two layer neural network obtains very good results on XOR-related and real UCI datasets.

Barbu, A., Y. She, L. Ding, and G. Gramajo
2017. Feature selection with annealing for computer vision and big data learning. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 39(2):272–286.

Draxler, F., K. Veschgini, M. Salmhofer, and F. A. Hamprecht
2018. Essentially no barriers in neural network energy landscape. *arXiv preprint arXiv:1803.00885*.

Dua, D. and C. Graff
2017. UCI machine learning repository.

Frankle, J. and M. Carbin
2019. The lottery ticket hypothesis: Finding sparse, trainable neural networks. *ICLR*.

Garipov, T., P. Izmailov, D. Podoprikhin, D. P. Vetrov, and A. G. Wilson
2018. Loss surfaces, mode connectivity, and fast ensembling of dnns. In *Advances in Neural Information Processing Systems*, Pp. 8803–8812.

Hinton, G. E., N. Srivastava, A. Krizhevsky, I. Sutskever, and R. R. Salakhutdinov
2012. Improving neural networks by preventing co-adaptation of feature detectors. *arXiv preprint arXiv:1207.0580*.

Kingma, D. P. and J. Ba

2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Soudry, D. and Y. Carmon

2016. No bad local minima: Data independent training error guarantees for multilayer neural networks. *arXiv preprint arXiv:1605.08361*.

Werbos, P.

1974. Beyond regression:" new tools for prediction and analysis in the behavioral sciences. *Ph. D. dissertation, Harvard University*.

Zhang, Y., J. Lee, M. Wainwright, and M. Jordan

2017. On the learnability of fully-connected neural networks. In *Artificial Intelligence and Statistics*, Pp. 83–91.