Cheap DNN Pruning with Performance Guarantees. / Conference Submissions

Anonymous authors
Paper under double-blind review

Abstract

Recent DNN pruning algorithms have succeeded in reducing the number of parameters in fully connected layers often with little or no drop in classification accuracy. However most of the existing pruning schemes either have to be applied during training or require a costly retraining procedure after pruning to regain classification accuracy. In this paper we propose a cheap pruning algorithm based on difference of convex (DC) optimisation. We also provide theoretical analysis for the growth in the Generalization Error (GE) of the new pruned network. Our method can be used with any convex regulariser and allows for a controlled degradation in classification accuracy while being orders of magnitude faster than competing approaches. Experiments on common feedforward neural networks show that for sparsity levels above 90% our method achieves 10% higher classification accuracy compared to Hard Thresholding.

1 Introduction

Recently, deep neural networks have achieved state-of-the-art results in a number of machine learning tasks [LeCun et al. (2015)]. Training such networks is computationally intensive and often requires dedicated and expensive hardware. Furthermore the resulting networks often require a considerable amount of memory to be stored. Using a Pascal Titan X GPU the popular AlexNet and VGG-16 models require 13 hours and 7 days respectively to train, while requiring 200MB and 600MB respectively to store. The large memory requirements limit the use of DNNs in embedded systems and portable devices such as smartphones, which are now ubiquitous.

A number of approaches have been proposed to reduce the DNN size during training time, often with little or no degradation to classification performance. Approaches include introducing bayesian sparsity inducing priors [Louizos et al. (2017); Blundell et al. (2015); Molchanov et al. (2017)] and binarization [Hou et al. (2016); Courbariaux et al. (2016)]. Other methods include the hashing trick has been used in [Chen et al. (2015)], tensorisation [Novikov et al. (2015)] and efficient matrix factorisations [Yang et al. (2015)].

However trained DNN models are used by researchers and developers that do not have dedicated hardware to train them, often as general feature extractors for transfer learning. In such settings it is important to introduce a cheap compression method, i.e., one that can be implemented as a postprocessing step with little or no retraining. Some first work in this direction has been [Kim et al. (2015); Han et al. (2015a); Han et al. (2015b)] although these still require a lengthy retraining procedure. Closer to our approach recently in [Aghasi et al. (2016)] the authors propose a convexified layerwise pruning algorithm termed Net-Trim. Building upon Net-Trim the authors in [Dong et al. (2017)] propose LOBS an algorithm for layerwise pruning by loss function approximation.

Pruning a neural network layer introduces a perturbation to the latent signal representations generated by that layer. As the pertubated signal passes through layers of non-linear projections the perturbation could become arbitrary large. In [Aghasi et al. (2016); Dong et al. (2017)] the authors conduct a theoretical analysis using the Lipschitz properties of DNNs showing the stability of the latent representations, over the training set, after pruning. The methods employed have connections to recent work [Sokolic et al. (2017); Bartlett et al. (2017); Neyshabur et al. (2017)] that have used the
Lipschitz properties to analyze the Generalization Error (GE) of DNNs, a more useful performance measure.

1.1 Contributions

In this work we introduce an cheap pruning algorithm for dense layers of DNNs. We also conduct a theoretical analysis of how pruning affects the Generalization Error of the trained classifier.

- We show that the sparsity inducing objective proposed in Aghasi et al. (2016) can be cast as a difference of convex problem, that has an efficient solution. For a fully connected layer with input dimension $d_1$ and output dimension $d_2$ while Net-Trim and LOBS scale like $O(d_1^2 N)$ and $O(d_2^2 N)$ respectively our algorithm scales like $O(d_1 d_2 N)$ with $d_2 \ll d_1$. Empirically our algorithm is orders of magnitude faster than competing approaches. We then extend our formulation to allow retraining a layer with any convex regulariser.

- We build upon the work of Sokolic et al. (2017) to bound the GE of a DNN after pruning. Our theoretical analysis provides a principled way of pruning while managing the GE.

Experiments on common feedforward architectures show that our method is orders of magnitude faster than competing pruning methods, while allowing for a controlled degradation in GE.

2 Our Formulation

2.1 DC Decomposition

We consider a classification problem, where we observe a vector $x \in \mathcal{X} \subseteq \mathbb{R}^N$ that has a corresponding class label $y \in \mathcal{Y}$. The set $\mathcal{X}$ is called the input space, $\mathcal{Y} = \{1, 2, \ldots, N_Y\}$ is called the label space and $N_Y$ denotes the number of classes. The samples space is denoted by $\mathcal{S} = \mathcal{X} \times \mathcal{Y}$ and an element of $\mathcal{S}$ is denoted $s = (x, y)$. We assume that samples from $\mathcal{S}$ are drawn according to a probability distribution $P$ defined on $\mathcal{S}$. A training set of $m$ samples drawn from $P$ is denoted by $\mathcal{S}_m = \{s_i\}_{i=1}^m = \{(x_i, y_i)\}_{i=1}^m$.

We start from the Net-Trim formulation and show that it can be cast as a difference of convex problem. For each training signal $x \in \mathbb{R}^N$ we assume also that we have access to the inputs $a \in \mathbb{R}^{d_1}$ and the outputs $b \in \mathbb{R}^{d_2}$ of the fully connected layer, with a rectifier non-linearity $\rho(x) = \max(0, x)$. The optimisation problem that we want to solve is then:

$$\min_U \frac{1}{m} \sum_{s_j \in \mathcal{S}_m} \|\rho(U^T a_j) - b_j\|_2^2 + \lambda \Omega(U)$$  \hspace{1cm} (1)$$

where $\lambda$ is the sparsity parameter. The term $\|\rho(U^T a_j) - b_j\|_2^2$ ensures that the nonlinear projection remains the same for training signals. The term $\lambda \Omega(U)$ is the convex regulariser which imposes the desired structure on the weight matrix $U$. 

2
The objective in Equation 1 is non-convex. We show that the optimisation of this objective can be cast as a difference of convex (DC) problem. We assume just one training sample for simplicity. 

\[
||\rho(U^T a) - b||^2_2 + \lambda \Omega(U)
\]

\[
= \sum_i [\rho(u_i^T a) - b_i]^2 + \lambda \Omega(U)
\]

\[
= \sum_i [\rho^2(u_i^T a) - 2\rho(u_i^T a)b_i + b_i^2] + \lambda \Omega(U)
\]

\[
= \sum_i [\rho^2(u_i^T a) + b_i^2] + \lambda \Omega(U) + \sum_i [-2b_i\rho(u_i^T a)]
\]

\[
= \sum_i [\rho^2(u_i^T a) + b_i^2] + \lambda \Omega(U) + \sum_{b_i < 0} [-2b_i\rho(u_i^T a)] + \sum_{b_i \geq 0} [-2b_i\rho(u_i^T a)]
\]

Notice that after the split the first term \((b_i < 0)\) is convex while the second \((b_i \geq 0)\) is concave. We note that \(b_i \geq 0\) by definition of the ReLu and set:

\[
g(U; x) = \sum_i [\rho^2(u_i^T a) + b_i^2]
\]

\[
h(U; x) = \sum_{b_i \geq 0} [2b_i\rho(u_i^T a)]
\]

Then by summing over all the samples we get:

\[
f(U) = \sum_j g(U; x_j) + \lambda \Omega(U) - \sum_j h(U; x_j)
\]

\[
= g(U) + \lambda \Omega(U) - h(U)
\]

which is difference of convex. The rectifier nonlinearity is non-smooth we can alleviate that by assuming a smooth approximation. A common choice for this task is \(\rho(x) = \frac{1}{\beta} \log(1 + \exp(\beta x))\) with \(\beta\) a positive constant.

### 2.2 Optimisation

It is well known that DC programs have efficient optimisation algorithms. We propose to use a variant of the DCA algorithm [Tao & An (1997)]. DCA is an iterative algorithm that consists in solving, at each iteration, the convex optimisation problem obtained by linearizing \(h(\cdot)\) (the non-convex part of \(f = g - h\)) around the current solution. Although DCA is only guaranteed to reach local minima the authors of [Tao & An (1997)] state that DCA often converges to the global minimum, and has been used successfully to optimise a fully connected DNN layer [Fawzi et al., 2015].

At iteration \(k\) of DCA, the linearised optimisation problem is given by:

\[
\arg\min_U \{ g(U) + \lambda \Omega(U) - Tr(U^T \nabla h(U^k)) \}
\]

where \(U^k\) is the solution estimate at iteration \(k\). The detailed procedure is then given in algorithms 1 and 2. We assume that the regulariser is convex but possibly non-smooth in which case the optimisation can be performed using proximal methods.

In more details for algorithm 2 at each iteration a minibatch \(A\) and \(B\) is drawn. The gradient for the smooth part is calculated and the algorithm takes a step in that direction with step size \(\rho_t\). Then the proximal operator for the non-smooth regulariser \(\lambda \Omega(\cdot)\) is applied to the result. We find that for the outer iterations \(K\) the values 15 to 50 are usually sufficient, while for the inner iterations \(T = 150\) is usually sufficient. We name our algorithm FeTa, Fast and Efficient Trimming Algorithm.
3 Generalization Error

3.1 Generalization Error of Pruned Layer

Having optimized our pruned layer for the training set we want to see if it is stable for the test set. We denote $f^1(\cdot, W^1)$ the original representation and $f^2(\cdot, W^2)$ the pruned representation. We assume that after training $\forall s_i \in S_m \|f^1(a_i, W^1) - f^2(a_i, W^2)\|_2^2 \leq C_1$. Second, we assume that $\forall s \in S \exists s_i \in S_m \Rightarrow \|a - a_i\|_2^2 \leq c$. Third the linear operators in $W^1, W^2$ are frames with upper frame bounds $B_1, B_2$ respectively.

**Theorem 3.1.** For any testing point $s \in S$ the distance between the original representation $f^1(a, W^1)$ and the pruned representation $f^2(a, W^2)$ is bounded by $\|f^1(a, W^1) - f^2(a, W^2)\|_2^2 \leq C_2$ where $C_2 = C_1 + (B_1 + B_2)c$.

The detailed proof can be found in Appendix A.

3.2 Generalization Error of Classifier

In this section we use tools from the robustness framework [Xu & Mannor, 2012] to bound the generalization error introduced by our approximation. We consider DNN classifiers defined as

$$g(x) = \max_{i \in [N_y]} (f(x))_i,$$

where $(f(x))_i$ is the $i$-th element of $N_y$ dimensional output of a DNN $f : \mathbb{R}^N \rightarrow \mathbb{R}^{N_y}$. We assume that $f(x)$ is composed of $L$ layers:

$$f(x) = f_L(f_{L-1}(\ldots f_1(x, W_1), \ldots W_{L-1}), W_L),$$

where $f_l(\cdot, W_l)$ represents the $l$-th layer with parameters $W_l$, $l = 1, \ldots, L$. The output of the $l$-th layer is denoted $z^l$, i.e. $z^l = f_l(z^{l-1}, W_l)$. The input layer corresponds to $z^0 = x$ and the output of the last layer is denoted by $z = f(x)$. We then need the following two definitions of the classification margin and the score that we take from [Sokolic et al., 2017]. These will be useful later for measuring the generalization error.
Definition 3.1. (Classification Margin). The classification margin of a training sample \( s_i = (x_i, y_i) \) measured by a metric \( d \) is defined as

\[
\gamma^d(s_i) = \sup \{ a : d(x_i, x) \leq a \Rightarrow g(x) = y, \forall x \}
\]  

(11)

The classification margin of a training sample \( s_i \) is the radius of the largest metric ball (induced by \( d \)) in \( \mathcal{X} \) centered at \( x_i \) that is contained in the decision region associated with class label \( y_i \). We then restate a useful result from Sokolic et al. (2017).

Corollary 3.1.1. Assume that \( \mathcal{X} \) is a (subset of) \( C_M \) regular \( k \)-dimensional manifold, where \( \mathcal{N}(\mathcal{X}; d; \rho) \leq (\frac{C_M}{\rho})^k \). Assume also that DNN classifier \( g(x) \) achieves a lower bound to the classification margin \( \gamma^d(s_i) > \gamma \), \( \forall s_i \in S_m \) and take \( l(g(x_i, y_i)) \) to be the 0–1 loss. Then for any \( \delta > 0 \), with probability at least \( 1 - \delta \),

\[
GE(g) \leq A \cdot (\gamma) - \frac{\delta}{2} + B
\]

(12)

where \( A = \sqrt{\frac{\log(2)N_s}{2^{k+1} \cdot (C_M)^k} \rho^2} \) and \( B = \sqrt{\frac{2 \log 1/\delta}{m}} \) are constants related to the data manifold.

We are now ready to state our main result.

Theorem 3.2. Assume that \( \mathcal{X} \) is a (subset of) \( C_M \) regular \( k \)-dimensional manifold, where \( \mathcal{N}(\mathcal{X}; d; \rho) \leq (\frac{C_M}{\rho})^k \). Assume also that DNN classifier \( g_1(x) \) achieves a lower bound to the classification margin \( \gamma^d(s_i) > \gamma \), \( \forall s_i \in S_m \) and take \( l(g(x_i, y_i)) \) to be the 0–1 loss. Assume also a new classifier \( g_2(x) \) which we have on layer \( i_* \) using Algorithm 1. Then for any \( \delta > 0 \), with probability at least \( 1 - \delta \),

\[
GE(g_2) \leq A \cdot \left( \gamma - \sqrt{C_2 \cdot \prod_{i > i_*} ||W_i||_F} \right) - \frac{\delta}{2} + B
\]

(13)

where \( A = \sqrt{\frac{\log(2)N_s}{2^{k+1} \cdot (C_M)^k} \rho^2} \) and \( B = \sqrt{\frac{2 \log 1/\delta}{m}} \) are constants related to the data manifold.

The detailed proof can be found in Appendix B. We must note here that \( k \) is the intrinsic dimensionality of the data manifold which is much smaller than the ambient dimension. Our result is quite pessimistic, and assumes that the \( GE \) grows exponentially in respect to the remaining layer depth of the pertubated layer. This is in line with previous work [Raghu et al. (2016); Han et al. (2015b)] that demonstrates that layers closer to the input are much less robust compared to layers close to the output. Our algorithm is applied to the fully connected layers of a DNN which are much closer to the output compared to convolutional layers.

4 Experiments

4.1 Time Complexity

We make experiments to compare the execution time of FeTa with that of LOBS and NetTrim-ADMM. We set \( \Omega(U) = ||U||_1 \) and aim for 95% sparsity. We see that given that \( d_1 \) are the input dimensions, \( d_2 \) are the output dimensions and \( N \) is the number of training samples FeTa scales like \( \mathcal{O}(d_1d_2N) \). Conversely LOBS scales like \( \mathcal{O}(d_1^3N) \) while NetTrim-ADMM scales like \( \mathcal{O}(d_2^3N) \) due to the required Cholesky factorisation. This gives a computational advantage to our algorithm in settings where the input dimension is large. We validate this by constructing a synthetic experiment with \( d_2 = 10, d_1 = \{2000 : 100 : 3000\} \) and \( N = 1000 \). The samples \( a \in \mathbb{R}^{d_1} \) and \( b \in \mathbb{R}^{d_2} \) are generated with i.i.d Gaussian entries. We plot in Figure 1 the results, which are in line with the theoretical predictions.

4.2 Classification Accuracy

4.2.1 Sparse Regularisation

In this section we perform experiments on the proposed compression scheme with feedforward neural networks. We compare the original full-precision network (without compression) with the
following compressed networks: (i) FeTa with $\Omega(U) = \|U\|_1$ (ii) Net-Trim (iii) LOBS (vi) Hard Thresholding. We refer to the respective papers for Net-Trim and LOBS, Hard Thresholding is defined as $F(x) = x \odot I(|x| > t)$ where $I$ is the elementwise indicator function, $\odot$ is the Hadamard product and $t$ is a positive constant.

Experiments were performed on two commonly used datasets:

1. **MNIST**: This contains $28 \times 28$ gray images from ten digit classes. We use 55000 images for training, another 5000 for validation, and the remaining 10000 for testing. We use the LeNet-5 model:

$$
(1 \times 6C5) - MP2 - (6 \times 16C5) - MP2 - 120FC - 84FC - 10SM
$$

where $C5$ is a $5 \times 5$ ReLU convolution layer, $MP2$ is a $2 \times 2$ max-pooling layer, $FC$ is a fully connected layer and $SM$ is a linear softmax layer.

2. **CIFAR-10**: This contains 60000 $32 \times 32$ color images for ten object classes. We use 50000 images for training and the remaining 10000 for testing. The training data is augmented by random cropping to $24 \times 24$ pixels, random flips from left to right, contrast and brightness distortions to 200000 images. We use a smaller variant of the AlexNet model:

$$
(3 \times 64C5) - MP2 - (64 \times 64C5) - MP2 - 384FC - 192FC - 10SM
$$

We prune only the first fully connected layer (the one furthest from the output) for clarity. Figure 2 shows the classification accuracy vs compression ratio for FeTa and Hard Thresholding. We see that Hard Thresholding works adequately up to $85\%$ sparsity. From this level of sparsity and above the performance of Hard Thresholding degrades rapidly and FeTa has $10\%$ higher accuracy on average.

<table>
<thead>
<tr>
<th>Method</th>
<th>Networks</th>
<th>Original Accuracy</th>
<th>CR</th>
<th>Pruned Accuracy</th>
<th>Computation Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>Net-Trim</td>
<td>LeNet-5</td>
<td>99.2%</td>
<td>95%</td>
<td>95.2%</td>
<td>455s</td>
</tr>
<tr>
<td>LOBS</td>
<td>LeNet-5</td>
<td>99.2%</td>
<td>95%</td>
<td>93.5%</td>
<td>75s</td>
</tr>
<tr>
<td>Thresholding</td>
<td>LeNet-5</td>
<td>99.2%</td>
<td>95%</td>
<td>83.1%</td>
<td>-</td>
</tr>
<tr>
<td>FeTa</td>
<td>LeNet-5</td>
<td>99.2%</td>
<td>95%</td>
<td>91%</td>
<td>32s</td>
</tr>
<tr>
<td>Net-Trim</td>
<td>CifarNet</td>
<td>86%</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>LOBS</td>
<td>CifarNet</td>
<td>86%</td>
<td>97.4%</td>
<td>80.5%</td>
<td>2h 47min</td>
</tr>
<tr>
<td>Thresholding</td>
<td>CifarNet</td>
<td>86%</td>
<td>97.4%</td>
<td>65.2%</td>
<td>-</td>
</tr>
<tr>
<td>FeTa</td>
<td>CifarNet</td>
<td>86%</td>
<td>97.4%</td>
<td>74.6%</td>
<td>9min</td>
</tr>
</tbody>
</table>
For the LeNet-5 model when compared to Net-Trim and LOBS we see that FeTa achieves lower classification accuracy but manages to prune the dense layer $2 \times$ to $15 \times$ faster. For the CifarNet model Net-Trim is not feasible on the machine used for the experiments as it requires over 16GB of RAM. Compared to LOBS FeTa again achieves lower accuracy but is $18 \times$ faster. Note that as mentioned in Dong et al. (2017) and Wolfe et al. (2017) retraining can recover classification accuracy that was lost during pruning. Starting from a good pruning which doesn’t allow for much degradation significantly reduces retraining time.

4.2.2 Low Rank Regularisation

As a proof of concept for the generality of our approach we apply our method while imposing low-rank regularisation on the learned matrix $U$. For low rank $k$ we compare two methods (i) FeTa with $\Omega(U) = ||U||_1$, and (ii) Hard Thresholding of singular values using the truncated SVD defined as $U = N \Sigma V^*$, $\Sigma = \text{diag}(\{\sigma_i\}_{1 \leq i \leq k})$. We plot the results in Figure 3.

In the above given $U \in \mathbb{R}^{d_1 \times d_2}$ the Compression Ratio (CR) is defined as $\text{CR} = (k * d_1 + k + k * d_2) / (d_1 * d_2)$. The results are in line with the $l_1$ regularisation, with significant degradation in classification accuracy for Hard Thresholding above 85% CR.

4.3 Generalization Error

According to our theoretical analysis the GE drops exponentially as the pruning moves away from the output layer. To corroborate this we train a Lenet-5 to high accuracy, then we pick a single layer and gradually increase it’s sparsity using Hard Thresholding. We find that the layers closer to the input are exponentially less robust to pruning, in line with our theoretical analysis. We plot the results in Figure 4.
Figure 4: Layer Robustness

For some layers there is a sudden increase in accuracy around 90% sparsity which could be due to the small size of the DNN. We point out that in empirical results [Raghu et al. (2016)] Han et al. (2015b) for much larger networks the degradation is entirely smooth.

5 CONCLUSION

In this paper we have presented an efficient pruning algorithm for fully connected layers of DNNs, based on difference of convex optimisation. Our algorithm is orders of magnitude faster than competing approaches while allowing for a controlled degradation in the Generalization Error. We provided a theoretical analysis of the degradation in GE resulting from our pruning algorithm. This analysis validates the previously observed phenomenon that network layers closer to the input are exponentially less robust to pruning compared to layers close to the output. Experiments on common feedfoward architectures validate our results.

6 APPENDIX

A. PROOF OF THEOREM 3.1.

We denote $f^1(\cdot, W^1)$ the original representation and $f^2(\cdot, W^2)$ the pruned representation. We assume that after training $\forall s \in S_m \|f^1(a, W^1) - f^2(a, W^2)\|^2_2 \leq C_1$. Second, we assume that $\forall s \in S \exists s_i \in S_m \Rightarrow \|a - a_i\|^2_2 \leq c$. Third the linear operators in $W^1, W^2$ are frames with upper frame bounds $B_1, B_2$ respectively. The following two lemmas will be useful:

Lemma 6.1. The operator $f^1(\cdot, W^1)$ is Lipschitz continuous with upper Lipschitz constant $B_1$.

Proof. See [Bruna et al. (2013)] for details, the derivation is not entirely trivial due to the non-smoothness of the rectifier non-linearity.

Lemma 6.2. The operator $f^2(\cdot, W^2)$ is Lipschitz continuous with upper Lipschitz constant $B_2$.

Proof. We see that: $\frac{d}{dx} \rho(x) = \frac{d}{dx} \frac{1}{2} \log(1 + \exp(\beta x)) = \frac{\exp(\beta x)}{1 + \exp(\beta x)} \leq 1$. Therefore the smooth approximation to the rectifier non-linearity is Lipschitz smooth with Lipschitz constant $k = 1$. Then $\|f^2(x, W^2) - f^2(y, W^2)\|^2_2 \leq k\|W^2 x - W^2 y\|^2_2 \leq \|W^2(x - y)\|^2_2 \leq B_2\|x - y\|^2_2$.

We drop the $W^1$ from the layer notation for clarity. Using the triangle inequality:
\[ \|f^1(a) - f^2(a)\|^2_2 = \|f^1(a) + f^1(a_i) - f^1(a_i) - f^2(a)\|^2_2 \]
\[ \leq \|f^1(a) - f^1(a_i)\|^2_2 + \|f^1(a_i) - f^2(a)\|^2_2 \]
\[ = \|f^1(a) - f^1(a_i)\|^2_2 + \|f^1(a_i) + f^2(a_i) - f^2(a)\|^2_2 \]
\[ \leq \|f^1(a) - f^1(a_i)\|^2_2 + \|f^1(a_i) - f^2(a_i)\|^2_2 + \|f^2(a_i) - f^2(a)\|^2_2 \]
\[ \leq B_1\|a_i - a\|^2_2 + C + B_2\|a_i - a\|^2_2 \]
\[ = C_1 + (B_1 + B_2)\|a_i - a\|^2_2 \]
\[ \leq C_1 + (B_1 + B_2)e \]

where we used Lemma 6.1 and Lemma 6.2 in line 5.

B. PROOF OF THEOREM 3.2.

We first introduce the following definition which will prove useful:

**Definition 6.1.** (Score). Take the score of a training sample \( s_i = (x_i, y_i) \)

\[ o(s_i) = \min_{j \neq y_i} \sqrt{2} (\delta_{y_i} - \delta_j)^T f(x_i), \]

(17)

where \( \delta_i \in \mathcal{R}^N_w \) is the Kronecker delta vector with \( (\delta_i)_i = 1 \).

Throughout the proof we will use the notation \( o_1(s_i) = o_1(x_i, y_i) \) and \( v_{ij} = \sqrt{2} (\delta_i - \delta_j) \).

Assume that the classification margin \( \gamma^d(s_i) \) of a training sample \( (x_i, y_i) \) is given and take \( j^* = \arg \min_{j \neq y_i} \min v_{y_{i,j},f(x_i)} \). We then take a point \( x^* \) that lies on the decision boundary between \( y_i \) and \( j^* \) such that \( o_2(x^*, y_i) = 0 \). The subscripts 1 and 2 indicate the original and the pruned architecture.

We get

\[ o_1(x_i, y_i) = o_1(x_i, y_i) - o_2(x^*, y_i) = v_{y_{i,j},f(x_i) - f^2(x^*)} \]
\[ \leq \|v_{y_{i,j},f(x_i)}\|_2 \|f^1(x_i) - f^2(x^*)\|_2 = \|f^1_L(x_i) - f^2_L(x^*)\|_2 \]
\[ \leq \prod_{i > i^*} \|W_i\|_F \|f^1_i(x_i) - f^2_i(x^*)\|_2 \]
\[ \leq \prod_{i > i^*} \|W_i\|_F \{\|f^1_i(x_i) - f^2_i(x^*)\|_2 + \|f^1_i(x^*) - f^2_i(x^*)\|_2\} \]
\[ \leq \prod_{i > i^*} \|W_i\|_F \{\|f^1_i(x_i) - f^2_i(x^*)\|_2 + \sqrt{C_2}\} \]
\[ \leq \prod_{i} \|W_i\|_F \|x_i - x^*\|_2 + \sum_{i > i^*} \|W_i\|_F \sqrt{C_2} \]
\[ \leq \prod_{i} \|W_i\|_F \gamma^d(s_i) + \sum_{i > i^*} \|W_i\|_F \sqrt{C_2} \]

where we used Theorem 3.1 in line 5. From the above we can therefore write

\[ \frac{o_1(x_1, y_i) - \sqrt{C_2} \prod_{i > i^*} \|W_i\|_F}{\prod_{i} \|W_i\|_F} \leq \gamma^d(s_i) \]

(19)

we know that a lower bound for \( o_1(x_1, y_i)\) \((\prod_{i} \|W_i\|_F)^{-1}\) exists \( o_1(x_1, y_i)\) \((\prod_{i} \|W_i\|_F)^{-1} = \gamma \).

Therefore we can finally write

\[ \gamma - \frac{\sqrt{C_2} \prod_{i > i^*} \|W_i\|_F}{\prod_{i} \|W_i\|_F} \leq \gamma^d(s_i) \]

(20)

The theorem follows from direct application of Corollary 3.1.1.

REFERENCES


