Lottery Tickets And Where To Find Them

Ryan Jiang Jack Pan

Abstract

The Lottery Ticket Hypothesis (LTH) postulates that large neural networks contain small sparse subnetworks that can be trained in isolation to perform. The strong LTH says that any neural network can be approximated by a subgraph within a sufficiently large random network. Recent work by Malach et al.[10] and Pensia et al.[14] prove the strong LTH for wider random graphs, using two-layer random graphs substitution. We analyse lottery tickets for linear regression in deep networks, give bounds on random pruning when looking for these tickets, and apply our findings to improve the bound on how large a random graph must be to contain a winning ticket.

1 Introduction

It's well-known that neural networks have a tendency to be over-parameterized. A recent result by Frankle and Carbin[4] showed strong evidence that dense networks contain subnetworks — the namesake lottery tickets — that are able to closely approximate the results of the full network. Though there has been considerable work studying this result empirically[15][1][6], the theoretical aspects of the problem are still largely unexplored.

In this report, we attempt to formalize the concept of neural networks and lottery tickets in the language of graph theory. Then, we apply our findings to theoretical analyses of tickets in neural networks for simple linear regression and multivariate linear regression. Then through framing the problem in terms of multivariate linear regression, we suggest an improvement to proving the lottery ticket hypothesis.

1.1 Contributions

Through analyzed properties of tickets in a neural network modelling simple linear regression, we give an expected number of minimal tickets existing in the network. Then we give an algorithm for randomized pruning of the edges, along with an upper bound on the number of edges we can prune k so that the probability we have destroyed all our tickets remains $\leq \frac{1}{c}$. This is done using a balls-and-bins argument. We then apply this analysis to a neural network modelling multivariate linear regression, and once again find a bound on the edges we can prune before having destroyed all our tickets.

Previous works have proved that winning tickets exist within larger, very wide random graphs[10], and that thin random graphs contain subgraphs larger than the winning ticket that approximate the ticket[14]. However, these works used a concatenation of shallow flat random graphs to approximate layers, which doesn't take advantage of the combinatorial nature of deep networks. By using arbitrarily deep random graph substitutions analyzed through MLR tickets, under a simplifying set of assumptions, we are able to prove approximate winning tickets with linear size exist within a random graph quadratic in the width and linear in the depth of the winning ticket.

2 Related Works

It is natural that achieving smaller, more compact networks that perform as well as a denser network is an attractive goal. As a result, pruning neural networks is an area of active research.

LeCun et al[9], Hassibi and Stork et al.[7], Tanaka et al.[17] and many others have improved our understanding of network pruning. As mentioned before, a recent result by Frankle and Carbin gives strong empirical evidence that: *A randomly-initialized, dense neural network contains a subnetwork that is initialized such that—when trained in isolation—it can match the test accuracy of the original network after training for at most the same number of iterations.* Random pruning was among the various strategies used by the authors to isolate for this subnetwork, and the empirical results from this purely random strategy were better than what one would initially expect[4].

Malach et al.[10] then analysed the theoretical aspects of the problem. They prove that a twice-deep and polynomial-as-wide network with randomly-initialized edge-weights likely contains a subnetwork that can approximate any arbitrary target function. This proves the Lottery Ticket Hypothesis for larger graphs under a simple set of assumptions.

Pensia et al.[14] further improves upon the result by Malach et al. using SubsetSum to show that a subnetwork approximating a target network with depth d and width w can be found in a randomly-initialized unpruned network of depth $2d$ and width wider by a factor of $O(\log dl)$. However, the winning ticket they prove exists does not have width linear with the target ticket. Our contribution is to analyze ticket sub-networks for simple linear regression and multivariate linear regression and improve the bound proposed by Malach et al. under a simplifying set of assumptions.

Orseau et al.[12] very recently showed a similar result to Pensia et al. with a weaker bound introducing product weight distributions for 2 layer graphs. We independently devised a general distribution for arbitrary depth path product weights PT(U). To the best of our knowledge, these are all the theoretical works aiming to prove the strong lottery ticket hypothesis.

3 Pruning Bounds

For the purposes of our analysis, we now give a formalized definition for the problem and the various structures involved.

First, we give a definition for a neural network in terms of graph theory. We can interpret a neural network as a directed acyclic weighted graph with S input nodes s_i and an output node t. For a network to be trainable, we require that there exist at least one directed path from s_i to t. A w-wide and d-deep network has the property that its nodes can be partitioned into d sets of size w such that a node in set L_i only has outgoing edges to all nodes in L_{i+1}. The process of training is equivalent to the process of assigning weights to the edges. We do not focus on training.

Frankle and Carbin[4] define a winning lottery ticket as a subnetwork that, when trained in isolation, can match the test accuracy of the original network. Our definition will be slightly different for the purposes of analysis. We define a ticket T as a subgraph where there exists an assignment of weights such that a corresponding neural network could perform. Note that this definition requires T to be trainable. Naturally, a winning ticket T^* is a ticket T with this assignment of edge-weights. A minimal winning ticket is thus a smallest possible winning ticket in a neural network — any larger tickets contain junk edges that can be pruned. By our definition, we note that a full unpruned neural network is trivially considered a winning ticket.

For the setting of our problem, we also make the following simplifying assumptions that are used in our analysis.

1. Having "junk" edges in a ticket negligibly impacts the effectiveness of the ticket

2. The probability that a given ticket is winning is p. For simplification purposes, we do not consider the process for training weights, only that a successful assignment of weights may or may not exist

3. We consider fully connected neural networks with ReLU activation without biases

With this machinery in place, we begin with some analysis on random pruning.

3.1 A Neural Network for Simple Linear Regression

Suppose we have a neural network, represented as a graph G, that models simple linear regression. G has depth d and width w. Then, a minimal ticket assuming linear activation or positive inputs is simply an s-t path. It is easily extended to ReLU activation in Section 4. The proof for this is given in the appendix.

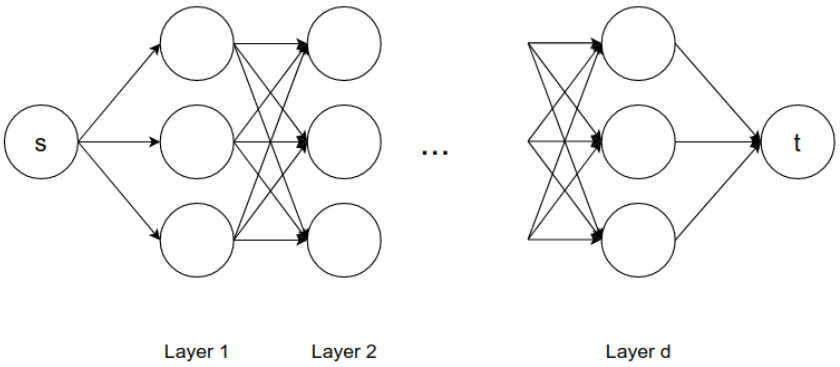

Figure 1: A fully-connected feed-forward neural network modelling simple linear regression as a directed graph

Then we have an expression for the expected number of minimal winning tickets in our network. By our definition, we know that the total number of s-t paths is w^d. Since p is the probability that any given ticket is winning and we know the number of tickets, the expected number of minimal winning tickets is $p * W^d$.

Now let's consider the number of edges we can prune by random sampling before we expect our graph to have no tickets. Since by our simplifying assumption, extraneous edges in a ticket do not impact the performance of the neural network, this gives an expected number of edges we can prune from our network. This is a fairly messy process to analyze but with some slight adjustment to the random pruning algorithm, the problem becomes easier to tackle. The pruning algorithm is very simple and is as follows, given a pruning bound b:

1. While $k < b$:
   (a) Pick a layer uniformly randomly (excluding the last layer)
   (b) Pick an outgoing edge from a node in that layer and delete it
   (c) Increase k by 1
   (d) Delete all nodes (and incoming edges to that node) that do not have a path to the output

The pruning bound b tells us how many edges we can prune. This next section of analysis will tell us how to get a good value for b.

Since there are w^2 outgoing edges from a layer, we disconnect a layer if we pick the same layer from which to remove an edge w^2 times. When we disconnect a layer, clearly we have no s-t paths which also means we have no tickets. We choose from $d-1$ layers (note that we do not prune from the last layer). Then we can treat this as a balls-and-bins problem where each of the $d-1$ layers is a bin and we want to know how many balls we can throw before a bin contains w^2 balls. Specifically, we want the largest k such that we can throw k balls without there being w^2 balls in a single bin. We calculate the following:

$$\begin{aligned}
Pr(\text{bin } i \text{ has } w^2 \text{ balls}) &= \binom{k}{w^2}(\frac{1}{d-1})^{w^2}(\frac{d-2}{d-1})^{k-w^2} \\
Pr(\text{some bin } i \text{ has } w^2 \text{ balls}) &= d\binom{k}{w^2}(\frac{1}{d-1})^{w^2}(\frac{d-2}{d-1})^{k-w^2} \\
&\leq \binom{k}{w^2}(\frac{1}{d-1})^{w^2-1}(\frac{d-2}{d-1})^{(k-w^2)}
\end{aligned}$$

Suppose we throw $k > w^2$, so the term $k - w^2$ is positive and we continue:

$$\begin{aligned}
&\leq \binom{k}{w^2}(\frac{1}{d-1})^{w^2-1} \\
&\leq (\frac{ke}{w^2})^{w^2}(\frac{1}{d-1})^{w^2-1} \\
&\leq e^{w^2+w^2\ln k - w^2\ln(w^2) - (w^2-1)\ln(d-1)}
\end{aligned}$$

We would like this probability to be small, which means we would like the term $w^2 + w^2\ln k - w^2\ln(w^2) - (w^2-1)\ln(d-1)$ to be negative. So we form:

$$\begin{aligned}
0 &> w^2 + w^2\ln k - w^2\ln(w^2) - (w^2-1)\ln(d-1) \\
&> w^2 + w^2\ln k - w^2\ln(w^2) - w^2\ln(d-1) + \ln(d-1) \\
-w^2 + w^2\ln(w^2) + w^2\ln(d-1) - \ln(d-1) &> w^2\ln k \\
w^2\ln k &< w^2\ln(d-1) - w^2 + w^2\ln(w^2) - \ln(d-1) \\
\ln k &< \ln(d-1) - 1 + \ln(w^2) - \frac{\ln(d-1)}{w^2} \\
e^{\ln k} &< e^{\ln d - 1 + 2\ln w - \frac{\ln(d-1)}{w^2}} \\
k &< e^{\ln(d-1)}e^{-1}e^{2\ln w}(e^{-\frac{\ln(d-1)}{w^2}}) \\
k &< \frac{dw^2}{e}(e^{-\frac{\ln(d-1)}{w^2}}) \\
&< \frac{(d-1)w^2}{e^{1+\frac{\ln(d-1)}{w^2}}}
\end{aligned}$$

So we have an expression for an upper bound on k to have a low probability that we've disconnected a layer. Analysing this expression, we consider the denominator:

$$\begin{aligned}
e^{1+\frac{\ln(d-1)}{w^2}} &= e \cdot (e^{\ln(d-1)})^{\frac{1}{w^2}} \\
&= e \cdot (d-1)^{\frac{1}{w^2}} \\
&< e(d-1) \\
&< (d-1)w^2 \quad \text{since } w^2 > e \text{ for } w \geq 2
\end{aligned}$$

And so we see that this upper bound for k scales up as d and w increase, as expected.

Note we can repeat the above argument to find an appropriate k such that the probability we destroy all our tickets is $\leq \frac{1}{c}$ for any positive c. To achieve this, instead of requiring the term

$w^2 + w^2 \ln k - w^2 \ln(w^2) - (w^2 - 1)\ln(d-1)$ to be negative, we would require:

$$-\ln c > w^2 + w^2 \ln k - w^2 \ln(w^2) - (w^2 - 1)\ln(d-1)$$

The same analysis from earlier follows. We show a shortened version here:

$$\begin{aligned}
-\ln c &> w^2 + w^2 ln k - w^2 ln(w^2) - (w^2 - 1) ln(d-1) \\
w^2 ln k &< w^2 ln(d-1) - w^2 + w^2 ln(w^2) - ln(d-1) - \ln c \\
\ln k &< \ln(d-1) - 1 + ln(w^2) - \frac{\ln(d-1)}{w^2} - \frac{\ln c}{w^2} \\
k &< e^{ln(d-1)} e^{-1} e^{2lnw} (e^{-\frac{ln(d-1)}{w^2}})(e^{-\frac{\ln c}{w^2}}) \\
k &< \frac{(d-1)w^2}{e} (e^{-\frac{ln(d-1)}{w^2}}) \\
&< \frac{(d-1)w^2}{e^{1+\frac{ln(d-1)}{w^2}+\frac{\ln c}{w^2}}}
\end{aligned}$$

Thus the above expression gives an upper bound on the number of edges we can prune, k, if we would like a probability $\leq \frac{1}{c}$ that we have disconnected the network.

3.2 A Neural Network for Multivariate Linear Regression

We can extend this analysis to multivariate linear regression. The difference is that a ticket has S source nodes with one output node. In a neural network that models multivariate linear regression, a subgraph that contains S not-necessarily-edge-disjoint paths, one from each s to t is considered a ticket. A minimal ticket would be a set of these S paths. The proof for this is also given in the appendix.

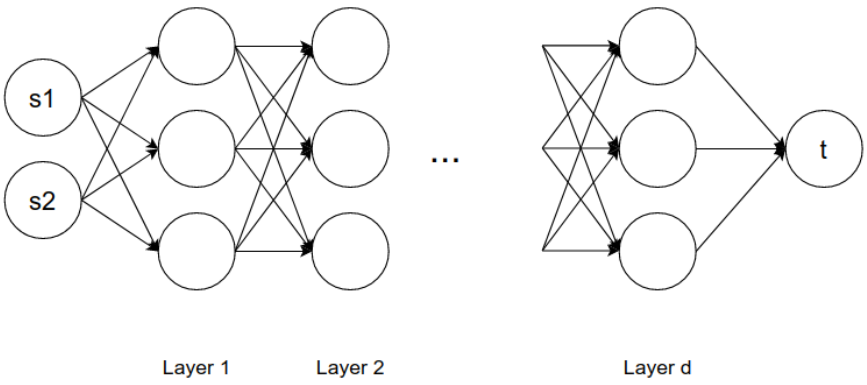

Figure 2: A fully-connected feed-forward neural network modelling multivariate linear regression as a directed graph

Once again, we find the expected number of winning tickets. To do this, we must first find the number of tickets present in the graph. We know from the simple linear regression case that for each of the S source nodes, there are w^d possible paths to the sink t. Since these paths do not require edge-disjointness, a minimal ticket is formed by a set that contains an s_i-t path for each s_i. This gives a total of $(w^d)^S$ possible minimal tickets. As before, p is the probability a ticket is winning so $p(w^d)^S$ gives the expected number of minimal winning tickets in such a network.

This analysis requires a small trick. Note that so long as we do not prune any outgoing edges from the very first or last layer, any path from the second layer to the sink is a ticket as each of the s_i will be able to follow along this path. So we guarantee our pruning algorithm does not prune this layer.

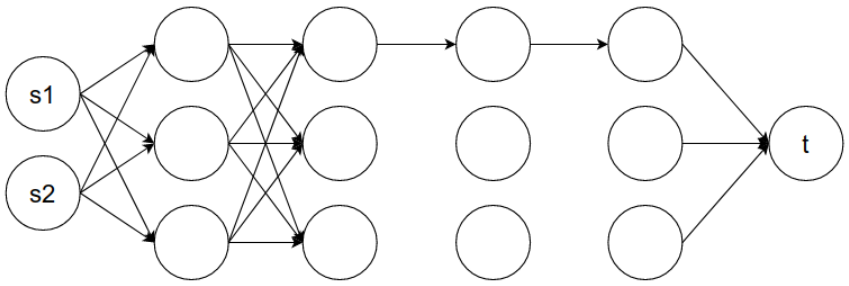

Figure 3: Notice that we have a path an s_1-t path and an s_2-t path so long as we do not disconnect the layers

We can reuse the majority of the analysis from the simple linear regression case. In fact, the only difference is that now we have a depth of $d-2$ instead of $d-1$. The remainder of the analysis is identical. So for a probability $\leq \frac{1}{c}$ that we disconnect the network, we would prune k edges such that:

$$k < \frac{(d-2)w^2}{e^{1+\frac{ln(d-2)}{w^2}+\frac{\ln c}{w^2}}}$$

4 Proving Large Networks Contain Winning Tickets

The idea behind the lottery ticket hypothesis is that given a ticket, whether the ticket can be trained to perform depends on its random initial edge weights. This suggests that large neural networks perform because they contain a combinatorial number of tickets[4].

Recent work by Malach et al.[10] showed that a random network of width $O(w^4d^2)$ and depth $2d$ can be pruned to approximate any network of depth d and width w without updating any of the weights. This June, Pensia et al.[14] showed using subset-sum that the large random network only has to be $O(log(wd))$ wider. However the ticket they find may not have width linear with the target ticket. This performant subgraph with random weights that Malach and Pensia et al. show to exist is a winning ticket which tells us that large networks contain winning tickets with high probability. Under a similar set of assumptions we show that winning tickets, with width linear to the target ticket, exist in deep random graphs with width $O(w^2)$. Our analysis follows Malach et al.'s analysis but the same ideas can be applied to improve the bound for the existence of non-minimal-tickets.

The input to every node in a layer of the winning ticket is the output of a multivariate linear regression function on the previous layer. This is shown in the diagram below.

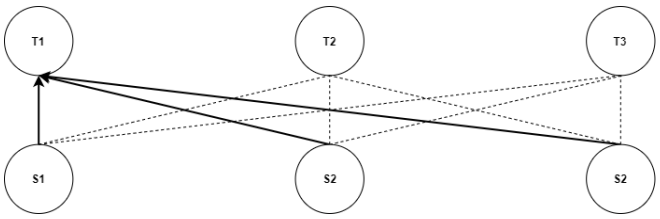

Figure 4: The edges between two layers in a fully connected network can be viewed as $|T|$ MLR graphs.

Previous works have shown the existence of a winning ticket subgraph in a twice-deep graph by replacing each layer in the ticket with 2 wider layers that have random edge weights. This method shows that within a wide random graph of depth 2 there exists a subgraph that approximates any single layer. Malach et al.'s approach works by introducing a wide middle layer between two winning ticket layers such that each path through this intermediate layer is an attempt to approximate some weight of an edge in one of the output layers[10]. Naturally, given a target weight β, they need to sample $\frac{1}{\epsilon}$ paths before they expect to find a path that has weight within ϵ of β. This makes the intermediate layer introduced very wide. So the constructed random graph by Malach et al. is linear in the depth of the winning ticket but very wide. In order to approximate a winning ticket of width 100, their bound requires a random graph of size at least 10^8. We show that under some simplifying assumptions that winning tickets exist in thin random graphs without having to increase pruned winning ticket width as in SubsetSum[14].

Note that the weights on the edges that connect one layer to the next can be framed as vector-output multivariate linear regression (MLR). So every path through the random graph with randomly-weighted edges is an attempt to approximate β_{ij}, the contribution of input s_i to some output t_j in the vector MLR.

Let p be the probability that a path in our random graph approximates β_{ij} for the MLR. Let a bad event be when a path does not approximate β_{ij}. We conjecture that the bad events — though they have dw^{d-1} dependencies — behave as if they were independent.

Theorem:
If our conjecture is true, then for some $\epsilon, \delta \in (0, 1)$ and for an arbitrary winning ticket F of depth F_{depth}, width F_{width}, and input size F_{input} where every edge weight $w \in F$ satisfies $||W||_2 \leq 1$, $|w| \leq \frac{1}{\sqrt{F_{width}}}$, then any random graph G with $G_{width} \geq \text{poly}(F_{depth}, F_{width}, F_{input}, d, \frac{1}{\epsilon}, \log\frac{1}{\delta})$ and depth $d * F_{depth}$ contains a subgraph that approximates F, where d is the depth of the random graph layer substitution. We initialize the weights of G from $PT(U, d)$. Then w.p at least $1 - \delta$ there exists a winning ticket $\tilde{G}$ within G such that $\sup_{x \in X} |\tilde{G} - F(x)| \leq \epsilon$.

The proof, given in the appendix, follows Malach et al.[10] with random graph substitution of depth d instead of 2, with edge weights sampled from $PT(U, d)$.

The motivation for using deeper random graph substitutions is to use a substitution that contains combinatorially many paths, and hence combinatorially many tickets for the MLR. Compared to using a 2-layer random graph which has paths linear with the width, our graph contains $O(w^d)$ paths and so $O(w^d)$ tickets. So we want a clever initialization of the weights such that the product of the weights along any path would be close to uniformly random. Then the product the weights along a path can have high density over the interval of possible target weights. If each path had a product of weights that was almost independent of the other paths we would have a combinatorial number of random samples to try and find close value for each β_{ij} in the MLR. Note that it is impossible for each path to have a product of weights that is independent from other paths because there are

many more paths than edges which means many more paths than random bits, hence the weight along a path cannot truly be random.

We also note that when pruning random graphs, the bulk of the random graph is dedicated to containing a subgraph that approximates MLR. Random graphs that use non-linear activation functions like ReLU actually require two disjoint paths from each input node to the MLR output node. Because of this the random graph could be smaller if we used linear activation functions when fishing for coefficients. However proving a random graph with linear activation contains a winning ticket would not explain why current over-parameterized networks with ReLU activations contain winning tickets. This does imply that perhaps linear activations have use in larger networks.

$PT(U, d)$ is the distribution such that the product of d random variables is almost uniform from interval [-x, x]. If the weights are drawn from this distribution, then the product of weights along a path of depth d are samples drawn from an almost uniform distribution. We are not aware of any previous work on finding a distribution for which its d-product is uniform. The probability density function for a d-product uniform distribution is piecewise and somewhat complicated, though if we were able to sample from a distribution with probability density function equal to its inverse, the resulting product samples would be uniform. However such an inverse is not tractable analytically. Instead we opt for an original approach using data transformation taking a variation on the box-cox transform for negative values. As power functions have the property $f(a * b) = f(a) * f(b)$, we can take a modified power-transform on the weights to reduce the kurtosis of the product distribution. By sampling from the PowerTransform($U[-1, 1]$, d), the product of the weights has an almost normal distribution with support $[-1, 1]$. By using a clipped form of $PT(U, d)$, clipped to the target weight interval $[-x, x]$, the support of our paths weights spans the possible winning ticket edge weights with high likelihood, at arbitrary depth. $PowerTransform(U) = sign(U)|U|^{\alpha}$, where α is the implicit solution to minimizing the kurtosis of $sign(U)|U|^{\alpha}$ with the constraint that the distribution is not bimodal. When the range of weights in the target graph is limited to a small range $[-x, x]$, there exists an α such that the range $[-x, x]$ is an almost uniform high density interval of $PT(U, \alpha)$.

By increasing the depth of our random substitution networks, we can adjust our random graph to had width closer to the width of our winning ticket, which has not been shown by previous works.

5 Conclusion

We formalized the concepts of fully-connected feed-forward neural networks and lottery tickets as graph problems. Then based on this formalization, we give a naive randomized pruning algorithm for finding tickets. For networks trying to model SLR or MLR, we give bounds for the number of edges we can prune while preserving tickets with high probability.

We showed the expected number of minimal winning tickets for SLR and MLR problems in deep networks, then proved that given independence, deep random graph substitutions can be used to improve the bounds on the necessary size of a minimal winning ticket containing random graph.

6 Further Work

In this section, we describe extensions of our results, follow-up questions, as well as the dead ends we encountered through the course of putting together this report.

6.1 Extensions

In Section 3, we gave a randomized pruning algorithm along with some bounds for the number of iterations we can expect to safely prune by. This was done by relating treating the pruning process as a balls-and-bins scenario and finding a maximal number of balls to throw before the maximum load of one bucket (ie. the number of edges we've pruned from a layer) becomes too high. A natural

extension of this is the *power of two choices*. This gives an exponential improvement on maximum load when number of balls thrown equals number of bins. Though it may not be practical for pruning — it would be costly to remember the number of edges in each layer for very large networks for when we compare the "number of balls" in two "bins" — it is interesting to think of how our pruning bounds would change in such a setting.

A recent work by Hayou et al.[8] gives theoretical results on effectively initializing weights in a network such that pruning yields a subnetwork that can be trained to be competitive to the full network — in other words, a ticket. Using their methods for identifying trainable weights, we can extend our setting from subgraphs that contain weights that perform to a more realistic setting where tickets can be trained to perform.

6.2 Dead Ends

Initially, we wanted to apply graph sparsification methods and analyses on graph pruning, however it is difficult to give concise conditions for the performance of a graph based on its weights. Spectral methods do not work on directed graphs and proving performance of random sampled graphs requires unrealistic assumptions about the target ticket and data.

An area we thought was initially promising was the relation between a neural network and a flow network. In the simple linear regression case where tickets took the form of an s-t path, there is a clear connection between a ticket and a max-flow problem where all edges have unit capacities. This connection is even more prominent in the multivariate linear regression case, where we are looking for a set of paths, one from each s_i to a sink t. We first considered a connection with multi-commodity flows, as each of the S sources naturally extend themselves as commodities in that problem. Then we considered a simple reduction: By adding a dummy source connecting to each of the s_i, where each of the connecting edges have unit capacity and the remaining edges of the network have infinite capacity, the presence of a ticket is exactly the presence of an S-unit flow from our dummy source to the sink. Unfortunately, we could not find anything concrete as a result of this connection.

7 Appendix

Here we include proofs for claims referenced throughout the project.

7.1 Ticket in SLR is an s-t path

For this case, we can assume positive inputs or linear activation function in the network. Then a neural network trying to model a simple linear regression is simply trying to model a single value — the slope of the linear model. An s-t path is a ticket since a winning assignment of edge-weights would be some set of values such that their product is this slope.

7.2 Ticket in MLR is a set of s_i-t paths, one from each source

This follows from the above result for SLR. In this case, we are trying to model several values β_i per source. The winning assignment for a ticket of this form would be a set of edge-weights such that the product along each path is a value β_i.

7.3 Random graphs contain Simple Linear Regression (SLR)

We show that any simple linear regression model, $\beta x = y$, represented as an edge from x to y with weight β, is approximated up to ϵ by a subgraph in a random graph of depth d, width w, source x, sink y.

$$(x)\beta \Rightarrow (y)$$

Sampling our weights from the clipped $PT(U, d)$ distribution, the product of the weights along a path are from the almost uniform distribution from $[-x, x]$ where $\beta \in [-x, x]$. Assuming that the product of path weights is uniform, the probability that the weight-product is within ϵ of the target β is at least $p \geq \frac{\epsilon}{2x}$. This is because if the path weight is some $x' \in [-x, x]$, then we need the weight-product to be within $[x' - \epsilon, x' + \epsilon]$. The worst-case scenario for this is when x' is one of the edge values, either x or $-x$. In this case, we only have a range of ϵ for the weight-product to be within ϵ of x'. Since we assume the weight-product is uniformly sampled (by our $PT(U, d)$ distribution), the probability we sample value in a range of ϵ from a range of $2x$ is exactly $\frac{\epsilon}{2x}$. Unlike Malach et al. we do not require all but one edge in a path to be within ϵ of 1 for our path's contribution to be β.

To model simple linear regression with a subgraph with ReLU activation, a ticket must contain two disjoint paths where the first edge in both paths have opposite signs. This way if x is positive, the positive path contributes βx to y, the negative path is blocked by ReLU activation. If x is negative the negative path contributes.

Assuming the performance of a path is independent of other paths, the probability of having no path with weight within ϵ of β is $(1 - p)^{w^{d-1}}$, as the number of paths from s to t is w^{d-1} as nodes in the last layer only have one edge to t.

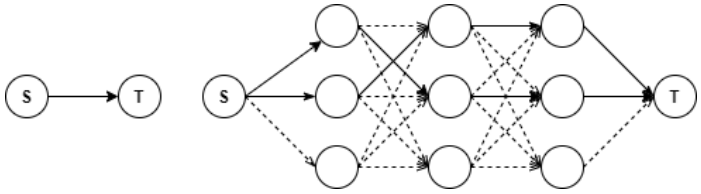

Figure 5: Every path weight product from S to T in the random graph is a sample that attempts to approximate the SLR edge.

The probability that there does not exist a path that starts with a positive edge is less than $\frac{\delta}{2}$, where δ is the probability that we fail to find an ϵ-approximation for the weight-product. This is equal to the probability of existence of the negative path. By union bound the probability of not containing either path is at most δ, so the probability we have a winning ticket is at least $1 - \delta$.

$$
\begin{aligned}
Pr(\text{no path approximates } \beta) &\approx (1-p)^{w^{d-1}} \\
&\leq exp(-w^{d-1}p) \quad \text{from } 1 - x \leq e^{-x}
\end{aligned}
$$

We want this probability to be $\leq \frac{\delta}{2}$ so we set:

$$
\begin{aligned}
exp(-w^{d-1}p) &\leq \frac{\delta}{2} \\
e^{-w^{d-1}p} &\leq \frac{\delta}{2} \\
-w^{d-1}p &\leq \log\frac{\delta}{2} \\
w^{d-1} &\geq \frac{\log(\frac{2}{\delta})}{p} \\
w &\geq (\frac{\log(\frac{2}{\delta})}{p})^{\frac{1}{d}}
\end{aligned}
$$

Previous works use SLR and MLR as building blocks to approximate layers. As this approach is linearly reductive, as our building blocks are $\hat{}\,\frac{1}{d}$ slimmer than if we had used a two-layer random

approximation graph, we expect our final graph to be notably smaller.

7.4 Random graphs contain MLR

We show that any scalar function multivariate linear regression model, $\sum \beta_i x_i = y$, represented as a graph with source nodes $S = \{x_1, ..., x_s\}$ and sink node y, with $|S|$ edges from x_i to y with weight β_i, is approximated up to ϵ by a subgraph in a random graph of depth d, width w, sources S, sink y.

The proof of existence for MLR is similar to that for SLR. If each β_i is approximated by a subgraph in G with probability at least $1 - \frac{\delta}{S}$, then with probability $\frac{\beta}{|S|}$, β is not approximated in G, then by union bound with probability δ we are missing the contribution of some input S_i. Hence with probability $1 - \delta$ some subgraph of G approximates the MLR.

If we want the sum of β_i approximations to be within ϵ of the true value, we want each β_i SLR approximation ticket to have error less than $\frac{\epsilon}{|S|}$. Hence, for a random graph to contain each β_i with error less than $\frac{\epsilon}{2|S|}$ with probability $1 - \frac{\delta}{|S|}$ from our SLR bound we can use a random graph with width greater than

$$\frac{w}{|S|} \geq (\frac{log(\frac{2}{\delta'})}{p'})^{\frac{1}{d-1}}, \quad \delta' \leq \frac{\delta}{|S|}, \quad p' \geq \frac{\epsilon'}{2} = \frac{\epsilon}{4|S|x} = \frac{p}{2|S|}$$

$$w \geq |S|(\frac{log(\frac{2|S|}{\delta})2|S|}{p})^{\frac{1}{d-1}}$$

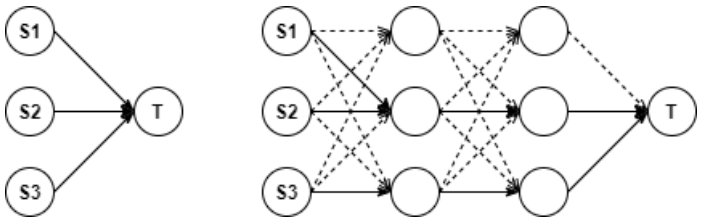

Figure 6: We insert a random graph with size large enough by our 7.3 bounds such that every weight beta in the MLR is well approximated in the random graph

7.5 Random graphs contain layers

We show that any vector function multivariate linear regression model, $\sum \beta_{ij} x_i = y_j$, represented as a graph with source nodes $S = \{x_1, ..., x_s\}$ and sink nodes $T = \{y_1, ..., y_t\}$, with $|S|$ edges for each of the $|T|$ scalar MLRs, is approximated up to ϵ by a subgraph in a random graph of depth d, width w, sources S, sinks T.

Each layer S in a network connected to the next layer T can be framed as $|T|$ scalar MLR graphs with source nodes S and sink nodes T_i. We want each of the $|T|$ MLR models to be approximated

in G with probability $1 - \frac{\delta}{|T|}$ within $\frac{\epsilon}{(|T|)^{1/d}}$. From our bound in MLR:

$$\frac{w}{|T|} \geq |S|(\frac{log(\frac{2|S|}{\delta'})2|S|}{p'})^{\frac{1}{d-1}}$$
$$\delta' \leq \frac{\delta}{|T|}$$
$$\epsilon' \leq \frac{\epsilon}{\sqrt{|T|}}$$
$$p' \geq \frac{\epsilon'}{2} = \frac{\epsilon}{2|S||T|^{\frac{1}{d}}} = \frac{p}{2|S||T|^{\frac{1}{d}}}$$
$$w \geq |T||S|(\frac{log(\frac{2|S||T|}{\delta'})2|S||T|^{\frac{1}{d}}}{p})^{\frac{1}{d-1}} \quad \text{from 7.4}$$

Hence there is a subgraph $\tilde{G}$ that approximates each scalar MLR with probability at least $1 - \frac{\delta}{|T|}$. As there are $|T|$ scalar MLR graphs, by union bound the probability that all scalar MLR graphs are well approximated is at least $1 - \delta$.

Consider the output vector T of sink nodes. The approximate vector MLR subgraph of G is off by:

$$||\tilde{G'}(x) - F(x)||_d^d = \sum(\tilde{G}_j(x) - F(x)_j)^d \leq \sum(\frac{\epsilon}{|T|^{\frac{1}{d}}})^d = \epsilon^d$$

7.6 Random graphs contain full networks

We show that any neural network F with depth F_{depth}, width F_{width}, number of source nodes F_{input}, and a single output, is approximated up to ϵ by a subgraph in a random graph of depth d, width w, sources S, sink t.

Each layer is approximated by a subgraph within a random graph from 7.5. Define a graph G where we replace each layer of F with a random graph. Since each layer in F is approximated by a random subgraph with high probability, F is approximated by G. We want each of the $|T|$ layers to be approximated in G with probability $1 - \frac{\delta}{|F_{depth}|}$ within $\frac{\epsilon}{2F_{depth}}$. From our bound in MLR:

$$w \geq |T||S|(\frac{log(\frac{2|S|}{\delta'})2|S|}{p'})^{\frac{1}{d-1}}$$
$$\delta' \leq \frac{\delta}{|F_{depth}|}$$
$$\epsilon' \leq \frac{\epsilon}{2F_{depth}}$$
$$p' \geq \frac{\epsilon'}{2} = \frac{\epsilon}{4|S||T|^{\frac{1}{d}}F_{depth}} = \frac{p}{2|S||T|^{\frac{1}{d}}F_{depth}}$$
$$w \geq |T||S|(\frac{log(\frac{2|S||T|F_{depth}}{\delta})4|S||T|^{\frac{1}{d}}F_{depth}}{p})^{\frac{1}{d-1}}$$

As each layer has error Ld norm less than $\frac{\epsilon}{F_{depth}}$ the sum of errors is less than ϵ following the proof by Malach et al. Each layer is well-approximated with probability at least $1 - \frac{\delta}{F_{depth}}$. Then by union bound, every layer is well approximated by some subgraph of G with probability at least $1 - \delta$.

Notice that by increasing the depth of our random substitution networks we can adjust our random graph to have smaller width. In addition this is not the smallest bound we can prove using deep random substitution graphs, by combining this with the SubsetSum approach from Pensia et al.[14] we can prove smaller than log width random graphs can be pruned to find a winning ticket.

References

[1] Zeyuan Allen-Zhu, Yuanzhi Li, and Zhao Song. *A Convergence Theory for Deep Learning via Over-Parameterization*. 2018. eprint: arXiv:1811.03962.

[2] Davis Blalock et al. *What is the State of Neural Network Pruning?* 2020. eprint: arXiv:2003.03033.

[3] L. Deng et al. "Model Compression and Hardware Acceleration for Neural Networks: A Comprehensive Survey". In: *Proceedings of the IEEE* 108.4 (2020), pp. 485–532.

[4] Jonathan Frankle and Michael Carbin. "The Lottery Ticket Hypothesis: Finding Sparse, Trainable Neural Networks". In: (2018). eprint: arXiv:1803.03635.

[5] Jonathan Frankle et al. *Linear Mode Connectivity and the Lottery Ticket Hypothesis*. 2019. eprint: arXiv:1912.05671.

[6] Jonathan Frankle et al. *Stabilizing the Lottery Ticket Hypothesis*. 2019. eprint: arXiv:1903.01611.

[7] Babak Hassibi and David G. Stork. "Second Order Derivatives for Network Pruning: Optimal Brain Surgeon". In: *Proceedings of the 5th International Conference on Neural Information Processing Systems*. NIPS'92. Denver, Colorado: Morgan Kaufmann Publishers Inc., 1992, pp. 164–171. ISBN: 1558602747.

[8] Soufiane Hayou et al. *Pruning untrained neural networks: Principles and Analysis*. 2020. eprint: arXiv:2002.08797.

[9] Yann Lecun, John Denker, and Sara Solla. "Optimal Brain Damage". In: vol. 2. Jan. 1989, pp. 598–605.

[10] Eran Malach et al. *Proving the Lottery Ticket Hypothesis: Pruning is All You Need*. 2020. eprint: arXiv:2002.00585.

[11] Ari S. Morcos et al. *One ticket to win them all: generalizing lottery ticket initializations across datasets and optimizers*. 2019. eprint: arXiv:1906.02773.

[12] Laurent Orseau, Marcus Hutter, and Omar Rivasplata. *Logarithmic Pruning is All You Need*. 2020. eprint: arXiv:2006.12156.

[13] Abhishek Panigrahi, Abhishek Shetty, and Navin Goyal. *Effect of Activation Functions on the Training of Overparametrized Neural Nets*. 2019. eprint: arXiv:1908.05660.

[14] Ankit Pensia et al. *Optimal Lottery Tickets via SubsetSum: Logarithmic Over-Parameterization is Sufficient*. 2020. eprint: arXiv:2006.07990.

[15] Vivek Ramanujan et al. *What's Hidden in a Randomly Weighted Neural Network?* 2019. eprint: arXiv:1911.13299.

[16] Alex Renda, Jonathan Frankle, and Michael Carbin. "Comparing Rewinding and Fine-tuning in Neural Network Pruning". In: (2020). eprint: arXiv:2003.02389.

[17] Hidenori Tanaka et al. *Pruning neural networks without any data by iteratively conserving synaptic flow*. 2020. eprint: arXiv:2006.05467.

[18] Fang Zhou, Sébastien Mahler, and Hannu Toivonen. "Simplification of Networks by Edge Pruning". In: vol. 7250. Jan. 2012, pp. 179–198. DOI: 10.1007/978-3-642-31830-6_13.

[19] Hattie Zhou et al. *Deconstructing Lottery Tickets: Zeros, Signs, and the Supermask*. 2019. eprint: arXiv:1905.01067.