ATTENTION IS OPTIMAL BUILDING BLOCK

Dan Navon

November 27, 2021

1 Basic Idea

1.1 Introducing

As shown in [1] if we throw all the activations from the transformers we can remain with polynomial of degree 3^L where L is the depth that is good approximation for the transformer in the sense that the induced networks achieves similar performance up to small degradation, hence we can think on this linear transformer for our analysis and make experiments at the end for standard transformers.

And again by [1] the expressive power of this transformer as measured by the separation rank, is as long as $L > \log_3 d_x$

$$sep_{(A,B)}\left(y\right) = \Theta\left(3^{L\cdot d_x}\right) \tag{1}$$

where d_x is the layer width, and L is the depth, and the 3 is since each layer y_i in the linear transformer is polynomial of degree 3, and now let ask ourselves what if we change the degree of each specific layer would we be able to achieve better expressive power with the same budget.

1.2 Formalizing

Let y_i^d denote layer of linear transformer but with degree d i.e

$$y_i^d\left(X\right) := X\prod_{k=1}^{d-1}\left(W_kX^t\right) \tag{2}$$

then if Y_L^k is linear transformer with p layers each from degree d then the obtained expressive power is

$$sep_{(A,B)}\left(Y_L^d\right) = \Theta\left(d^{L\cdot d_x}\right) \tag{3}$$

while the number of parameters we used is

$$N_{L,d} = L\cdot d\cdot N_{d_x} \tag{4}$$

as long as $L > \log_d d_x$ where N_{d_x} is the number of parameters of some specific weights matrix W_k, hence for fixed budget of parameters B we are looking after

$$\arg\max_{d,L} d^{L\cdot d_x}$$
$$L\cdot d = \frac{B}{N_{d_x}}$$

or alternatively we are looking to maximize

$$f\left(d\right) := d^{\frac{B\cdot d_x}{dN_{d_x}}} = \left(d^{\frac{1}{d}}\right)^{\frac{B\cdot d_x}{N_{d_x}}}$$

And observe that

$$d^* := \arg\max_d \left(d^{\frac{1}{d}}\right)^{\frac{B\cdot d_x}{N_{d_x}}} \underbrace{=}_{\frac{B\cdot d_x}{N_{d_x}}>0} \arg\max_d d^{\frac{1}{d}} = \arg\max_d \frac{\ln d}{d}$$

finally d_c is critical point of $h(d) := \frac{\ln d}{d}$ iff

$$0 = h'(d) = \frac{1}{d^2} - \frac{\ln d}{d^2} \iff \ln d = 1 \iff d = e \tag{5}$$

hence $d_c = e$ is the only critical point of h and since,

$$d > e \implies h'(d) = \frac{1-\ln d}{d^2} < \frac{1-\ln e}{d^2} = 0$$
$$d < e \implies h'(d) = \frac{1-\ln d}{d^2} > \frac{1-\ln e}{d^2} = 0$$

But we know that

$$d > e \implies h(d) = h(e) + \underbrace{\int_e^d h'(s)\, ds}_{<0} < h(e)$$
$$d < e \implies h(d) = h(e) - \underbrace{\int_d^e h'(s)\, ds}_{>0} < h(e)$$

Hence the only maximum is obtained for $d = e \approx 2.71$ and the function is monotone decreasing when we go farther on both sides, hence if we want $d^*_{\mathbb{N}} \in \mathbb{N}$ then it must hold that $d^*_{\mathbb{N}} \in \{2, 3\}$ finally

$$h(3) > h(2) \iff \frac{\ln 3}{3} > \frac{\ln 2}{2} \iff 2\ln 3 > 3\ln 2 \iff \ln 3^2 > \ln 2^3 \iff 3^2 > 2^3 \iff 9 > 8$$

Conclusion 1.1 *Concluding $d^*_{\mathbb{N}} = 3$ while $d^* = e$ and the optimal building block for the transformer mechanism should be polynomial of degree 3 which is exactly the attention mechanism, as long as we assume that it should be polynomial with integer degrees.*
If in someway we know to create building blocks with fractional degrees then further improvement can be achieved by taking each building block to be with degree e.

2 Missing Pieces

Task 1 Extend the proof from [1] into rank k polynomials within each building block.
Task 2 Conduct experiments
Task 3 How to build fractional transformers with $d = e$?
Task 4 It also explain the mixer MLP results (Need Detailed Analysis).

3 Experiments

Since Task-1 is time consuming and low risk, let start with conducting the experiments first.

3.1 Experiment Design

For each $d \in \{2, 3, 4\}$ we will train DNN $\mathcal{T}_d$ that composes from the layers $\left(L_1^d, ..., L_p^d\right)$ i.e

$$\mathcal{T}_d(x) := L_p^d \circ L_{p-1}^d \circ ... \circ L_1^d(x) \tag{6}$$

where $\forall i,\quad L_i^d := \sigma \circ Q_i^d(x)$ and $Q_i^d(x)$ is polynomial of degree d in x, and σ is some activation maybe combined with normalization.
Summarizing we will train several FC-DNN's but in each of them the linear layers would be replaced by polynomials of degree d, and our goal would be to find which d achieves the best results, for fixed parameters budget B, where our theory claims that $d = 3$ is the optimal one.

Comment 3.1 *Our polynomial for degree* d *is of the form* $P_d(x) := \prod_{k=1}^{d}(W_k x)$ *In particular it doesn't contains all the coefficients and it is sparse polynomial.*

3.2 Experiments Development

3.2.1 Discussion

So now we can run networks with layers L_*^d for $d \in \{2, 3, 4\}$ until convergence, and it all works, in addition the final error rate is even low, the only remaining question is how to compare which is better but for this indeed we need, to run deep networks until convergence and see which of them is winning, also we may need hard task with lot of data, s.t not all the networks would be able to achieve maximal accuracy, and we would be able to see which of them is the best, i.e only the one with maximal expressive power would achieve good accuracy and all the rest will left somewhat behind.

3.2.2 Training Deep Networks

In order to train deep $\mathcal{T}_d$ networks we would need to be able to normalize it all maybe by taking the network to be with tiny width, and maybe also combining Batch-Normalization layers, and multi-head architectures.

3.2.3 Finding Tougher Tasks

If we are looking for tougher tasks, then of course there are many nontrivial tasks, and we are looking for one that would be easy to implement and even medium scale networks which is what I'm planning to execute would be able to achieve some results on them in the level that we would be able to differentiate between the different kind of networks $\mathcal{T}_d$ by their performance i.e final accuracy and speed of convergence.

3.2.4 Easy To Implement

In anyway since we don't really care about the specific task, and exact number of heads in each layer it better for it all to be easy to implement and simple, in order to save time and investment, and make our results more reliable.

3.3 Stage 1.1 Task Replacement

So let start to think on which tasks we are gonna to work on, then maybe a first task to start with is to look after easy to implement but hard to learn tasks, which maybe we can find by starting from some existing repository for this task, and make minimal change by replacing the architectures, and make adaptations for the tensors.

3.4 Stage 1.2 Task Creation

3.4.1 Idea

Since we only train some neural network that should be able to mix information, it may be good enough to create any artificial task, that requires to mix information in order to be learned, so let start with thinking which tasks are we wanna to create.

3.4.2 Candidates

So maybe we can define each function of the form $f(x_1, ..., x_n)$ that have high separation rank, and try to learn it, and if this function also have high ϵ-separation rank, then each network with low separation rank, wouldn't be able to ϵ-approximate this function, hence would fail in the final task, so to say it clearly we are looking after any function that any good approximation of it would require high separation rank, and now we may start by developing some theoretic notion for this high separation rank and which families of functions obey this notion and then all together we would be able to find, the right functions, and run experiments of which networks succeed to approximate those functions better in some sense that we may think later what this exact sense should be.

4 Separation Rank Concept Development

4.1 Reminder

So let start with some reminder of what this concept of separation rank means,

Definition 4.1 *The separation-rank of $f : \mathcal{X}^{2M} \to \mathcal{R}$ w.r.t some partition (A, B) is defined as follows:*

$$sep_{(A,B)}(y) := \min\left\{R \in \mathcal{N} : \exists g_1, ..., g_R, g_1^{'}, ..., g_R^{'} : \mathcal{X}^M \to \mathcal{R} \text{ s.t. } y(A,B) = \sum_{r=1}^{R} g_r(A)\, g_r^{'}(B)\right\} \tag{7}$$

We will further make use of the norm-separation-rank, which we would define as follows, if we denote by Y the space

$$Y := \left\{y : \ y : \mathcal{X}^{2M} \to \mathcal{R}\right\} \quad \text{and} \quad d : Y \times Y \to \mathbb{R}_{>0} \quad \text{Is some distance metric} \tag{8}$$

then the d-separation-rank would be defined as follows

$$\epsilon - sep_{(A,B)}^{d}(y) := \min_{\tilde{y} \in \mathbb{B}_{\epsilon}^{Y}(y)} sep_{(A,B)}(\tilde{y}) \tag{9}$$

And it can be realize as what is the function with minimal separation rank, i.e the simplest function, s.t that can approximate y up to error ϵ according to the metric d, i.e it is our way to measure if some network with low separation rank would be able to approximate y with small error.

4.2 Short Discussion

As next step let start with thinking, where do we wanna to go, then we want to characterize family of functions with high $\epsilon - sep_{(A,B)}^{d}(y)$ and find exactly what is their separation rank, and if each NN with the relevant separation rank can indeed mimic this function, if indeed we succeed with finding this kind of family of functions, then we can run the experiments on it and it all would be simple.

4.3 separation-rank lower bound

So if we are looking after easy to compute estimations then, the first tool we can use is

Lower Bound 4.2 *Let $M_{(A,B)}^{y}$ be some matricization of y for any specific choice of template vectors, as defined in previous works by (Levine et al) then we know for sure that*

$$rank\ M_{(A,B)}^{y} \le sep_{(A,B)}(y) \tag{10}$$

And now we may want to generalize this lower bound further into the ϵ-separation-rank, and ϵ-d-separation-ranks definitions.
so let $\mathcal{F}_{\epsilon}$ be the set of functions we are conducting the minimum on in those definitions then from the first definition we know that

$$\forall \tilde{y} \in \mathcal{F}_{\epsilon} \quad rank\ M_{(A,B)}^{\tilde{y}} \le sep_{(A,B)}(\tilde{y}) \quad \Longrightarrow \quad \min_{\tilde{y} \in \mathbf{F}_{\epsilon}} rank\ M_{(A,B)}^{\tilde{y}} \le \min_{\tilde{y} \in \mathbf{F}_{\epsilon}} sep_{(A,B)}(\tilde{y}) \tag{11}$$

Conclusion 4.3 *We have the following lower bound on the ϵ-d-separation-ranks definitions.*

$$\min_{\tilde{y} \in \mathcal{F}_{\epsilon}} rank\ M_{(A,B)}^{\tilde{y}} \le \min_{\tilde{y} \in \mathcal{F}_{\epsilon}} sep_{(A,B)}(\tilde{y}) \tag{12}$$

Where $\mathcal{F}_{\epsilon} \subseteq Y$ is the subset of functions induced by (ϵ, Y, d).

Now if for example $d = \|.\|_{\infty}$ then $\epsilon - rank\ M_{(A,B)}^{\tilde{y}} = \min_{\tilde{y} \in \mathcal{F}_{\epsilon}} rank\ M_{(A,B)}^{\tilde{y}}$ and we get some well known notion.

4.4 choosing the right metric

Then what is our metric, then clearly its the accuracy we get on our task, and if our test set is of the form,
$\mathcal{D}_t := \{(\overline{x_i}, y_i)\}_{i \in K}$ and the error for each point is determined by $e(y_i, \hat{y}_i)$ where e_i is some specific function then the most reasonable measure for the overall error is of course

$$E(y, \mathcal{D}_t) = \frac{1}{|\mathcal{D}_t|} \sum_{i=1}^{|\mathcal{D}_t|} e(y_i, \hat{y}_i) \tag{13}$$

So if this is the empirical error then the continuous equivalent generalization should be,

$$E(y, y_{gold}) = \frac{1}{m(\mathbf{X})} \int_{\mathbf{X}} e(y(\overline{x}), y_{gold}(\overline{x}))\, d\overline{x} \tag{14}$$

where y is our trained network, while y_{gold} is the network we are trying to mimic, which is the ground truth prediction function.

Hence the global error should be taken as the $\|.\|_1$ of the error at each specific point separatly, and the only remaining open question in this context is what $e\left(y\left(x\right), y_{gold}\left(x\right)\right)$ should be.

4.5 choosing the pointwise error

4.5.1 general discussion

So before we starting with the mathematical development let start with thinking what do we want from our metric first, then this error is just the loss in the training of any regular network, and as we know the loss is problem dependent, since it should be chosen by what exactly do we want from our problem, and it just say how much do we care about each kind of error.

But in any case the loss must be monotonic increasing with the distance between the predicted and the gold labels, i.e if we assume that $y \in \mathbb{R}$ then it must be that $l\left(y, y_{gold}\right)$ is monotonic increasing in $|y - y_{gold}|$ so we can assume for start that $l\left(y, y_{gold}\right) = g\left(y - y_{gold}\right)$ for some monotonic-increasing g.

But here we can get some support form the notion of equivalence of monotonic increasing functions, which can be defined by bounding up to some constants, and since $\|x^2\|_1 = \|x\|_2^2$ and by the same way we have $\|x^p\|_1^{\frac{1}{p}} = \|x\|_p$ so what I'm trying to claim that as long as all the losses are monotone increasing in the way we talked about above then since all the norms on $\mathbb{R}^n$ are equivalent then handling one loss function is equivalent for handling all the rest, up to multiplicative constants factors, so WLOG $e \in \{e_1, e_2\}$ for $e_1\left(y, y'\right) = \left|y - y'\right|$ or $e_2\left(y, y'\right) = \left(y - y'\right)^2$ where we will choose e_1 or e_2 according for which of them is easier to compute, in each place and it would only affect the relevant ϵ i.e the radius of the environment by multiplicative constants, which we don't really care about.

4.5.2 Concluding

All together we get

$$d_1\left(y, y_{gold}\right) := \mathbb{E}\left[|y - y_{gold}|\right] \qquad d_2\left(y, y_{gold}\right) := \mathbb{E}\left[|y - y_{gold}|^2\right] \tag{15}$$

which are the first and second moments of $|y - y_{gold}|$ i.e of the absolute error variable, and that those metrics are representative for the whole space of possible options.

4.5.3 Where are we going

Now let ask ourselves where are we going with this discussion, so just as reminder we are looking to characterize which functions has high separation rank w.r.t the above metric, and want to find families of such functions.

5 Functions Characterization

5.1 Simple Starting

So let start with the most simple cases which are polynomials, and multinationals, since they are already written in right form hence are easy to interpret.

5.1.1 Single Product

Let $\mathcal{F}_1 := \left\{p_1\left(\overline{x}\right) \mid p_1\left(\overline{x}\right) := \prod_{i=1}^{n} x_i^{\alpha_i} \quad \forall i\ \alpha_i \geq 1\right\}$ then clearly $sep_{(A,B)}\left(\mathcal{F}_1\right) = 1$ continuing by this way if we take $\mathcal{F}_d := \left\{p_d\left(\overline{x}\right) \mid p_d\left(\overline{x}\right) := \sum_{l=1}^{d} a_l \prod_{i=1}^{n} x_i^{\alpha_i^l} \quad \forall i\ \alpha_i^l \geq 1\right\}$ and we would write sep for $sep_{(A,B)}$ from now for easy of notation then clearly $\forall p_d \in \mathcal{F}_d\ \ sep\left(p_d\right) \leq d$ but of course it could be smaller since for example $p_2\left(x, y\right) = \left(x + x^2\right) y = xy + x^2 y$ satisfies $sep\left(p_2\right) = 1 < 2$ so it reasonable to assume that in each $\mathcal{F}_d$ there are polynomials from sep-ranks of $\{1, ..., d\}$ and from each of these ranks, hence the main questions we can ask is
Questions

- which polynomials in $\mathcal{F}_d$ are of sep-rank d.

- what is their measure
- which of them are from ϵ-sep-rank d.
- what is their measure

5.1.2 Discussion

So what are the different tools we have to attack this problem then
Tools

- we have the matricization lower bound.
- we can do algebra, with brute-force arguments.

In addition, we can try to claim that almost all of the relevant polynomials are from high ϵ-sep-rank, now it seems like going through matricization is easier way to go but it would come with the price of recalling how the matricization exactly works.

5.1.3 Matricization Recap

So if we take set $T := \{x_1, ..., x_M\}$ of template vectors then we have the n dimensional tensor $A(y)_{j_1,...,j_n} = y(x_{j_1}, .., x_{j_n})$ and if we reshape it into matrix form $\left([[A(y)]]_{A,B}\right)_{row,col} = A_{j_1,...,j_n}$ where $row = 1 + \sum_{t=1}^{\frac{N}{2}} (j_{a_t} - 1) M^{\frac{N}{2}-t}$ and $col = 1 + \sum_{t=1}^{\frac{N}{2}} (j_{b_t} - 1) M^{\frac{N}{2}-t}$ for $A := \left\{a_1, ..., a_{\frac{N}{2}}\right\}$ where $a_1 < ... < a_{\frac{N}{2}}$ and $B := \left\{b_1, ..., b_{\frac{N}{2}}\right\}$ where $b_1 < ... < b_{\frac{N}{2}}$.

5.1.4 Measure Detection

A

6 Proof Generalization

6.1 Strategy

So now our goal is to find the expressive power of the relevant networks, so let $\mathcal{T}_d^p$ be the network with depth p where each layer has degree d, as defined above, then we want to measure the expressive power of the network as measured by the separation rank, then the most obvious way is to start from the existing proof from for the transformer with the attention mechanism and try to extend it into the case where the degree is d instead of 3.

6.2 Proxy Measure

And for start let start with some proxy measure, of taking the degree of the final polynomial as some proxy for the complexity and maybe the expressiveness of the entire polynomial, then indeed,

$$degree\left(\mathcal{T}_p^d\right) = degree\left(L_p \circ ... \circ L_1\right) = degree\left(L_p\right) \times ... \times degree\left(L_1\right) = d^p \tag{16}$$

And since if N_{d_x} is the number of parameters in each specific layer then the total number of parameters in the network is $d \cdot p \cdot N_{d_x} = B$ then fixing the budget we have $p = \frac{B}{N_{d_x}} \cdot \frac{1}{d}$ combining with (17) we have

$$degree\left(\mathcal{T}_p^d\right) = \left(d^{\frac{1}{d}}\right)^{\frac{B}{N_{d_x}}} \tag{17}$$

Hence the total degree is maximized when d is taken to maximize the expression $d^{\frac{1}{d}}$ which results in $d = e$ when $d \in \mathbb{R}$ and $d = 3$ if we require $d \in \mathbb{N}$.

6.3 Separation Rank

Now let try to prove for the separation rank, in the same way like before, then we already have the proof for the case $p = 3$ and instead of attacking immediately the general case let start with the case $d = 2$ first

6.3.1 Warn-Up d=2

In this case we have

$$L_i^d(x) = x^T W_{i1}^T W_{i2} x \tag{18}$$

Or alternatively

$$L_i^d(x) = x^T W x \tag{19}$$

where $rankW = d_x$ i.e it can be factorized into the form $W = W_{i1} W_{i2}^T$ and just to feel how this formula behaves then

$$L_2 \circ L_1(x) = L_2(L_1(x)) = L_2\left(x^T W_1 x\right) = \left(x^T W_1 x\right) W_2 \left(x^T W_1 x\right) = \left(x^T W_1 x\right) W_2 \left(x^T W_1 x\right) \tag{20}$$

and all of this is after we forget the W_O in the end of each layer to normalize the dimensions.

6.3.2 Naive Approach

Then first each element in polynom of the form $q_{k_1,\ldots,k_d}(x_1,\ldots,x_d) := \prod_{j=1}^{d} x_j^{k_j}$

References

[1] Yoav Levine Noam Wies Daniel Jannai Dan Navon Yedid Hoshen Amnon Shashua. "THE INDUCTIVE BIAS OF IN-CONTEXT LEARNING: RETHINKING PRETRAINING EXAMPLE DESIGN". In: *CVPR 2021* 0.0 (2021), p. 40. DOI: `https://arxiv.org/pdf/2110.04541.pdf`.