Mechanistic Interpretability and Grokking

Gene Cooperman // gene@ccs.neu.edu

Oct. 6, 2023

Reviewing Two Papers

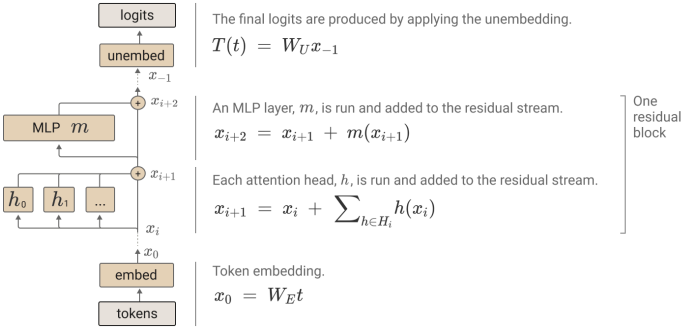
We review two papers, and skip a third paper:

1. "A Mathematical Framework for Transformer Circuits", `https://transformer-circuits.pub/2021/framework/index.html`
2. "Progress measures for grokking via mechanistic interpretability", `https://arxiv.org/abs/2301.05217`
3. "In-context Learning and Induction Heads", `In-contextLearningandInductionHeads`

The first paper is a prerequisite to understanding the terminology used for Mechanistic Interpretability, as used in the second paper. The third paper discussed the importance of "induction heads" (also described in the first paper), when scaling up. Hence, the name "in-context learning".

Outline of Talk

The LLM model (decoder only; no encoder)



The diagram from "A Mathematical Framework …" shows an easy-to-understand formalism for an LLM. Recall that an LLM is an autoregression (training by predicting the next token from the previous tokens), and using that training to update all weights using the classical backpropagation technique of an RNN (Recurrent Neural Net).

The LLM Model (cont.)

This includes an embedding and unembedding matrix (W_E and W_U) to embed the next block of tokens as activations in the first layer, and later unembed the internal representation as an output. There are then one or more blocks that include layers for attention heads, followed by layers for MLP.

MLP stands for “MultiLayer Perceptron”. An MLP is the classical neural net, with an input layer of neurons, one or more hidden layers and an output layer. The output of the MLP is compared with the “correct” output, and the activation weights on all previous layers are updated.

The Attention Heads

An LLM has one or more "attention heads". Multiple attention heads operate in parallel to update the next layer of neurons (n_{i+1}) based on the current layer of neurons (n_i).
An attention head is computed as a function from one layer to the next.
The attention head is normally formulated with weights involving a tensor, and is encoded using a framework such as PyTorch.
Next, we will look inside that function $h()$. In Mechanistic Interpretability, we reformulate the weights in $h()$ as an equivalent formulation involving:

1. a query and key matrix; and
2. a value and output matrix.

Mathematical Background: What is a tensor?

A tensor, like a matrix, is a linear function on its argument. A *linear function* is one in which $f(\alpha x) = \alpha f(x)$ and $f(x+y) = f(x) + y(x)$, where α is a scalar.

As an example, a matrix is a linear function from a vector to a vector. The argument to a tensor can be a vector, a matrix, or even a tensor with a lower number of dimensions. Just as we have the product of a matrix and vector (by reducing along one of the dimensions), or a product of a vector and vector (by reducing along the only dimension), or a product of a matrix and matrix (by reducing along two dimensions), a tensor product has a similar concept. Other analogous terms are *outer product* (multiply with no reduction) and *inner product* (reduction on all dimensions).

Looking Inside the Attention Heads

In main article, see: “Attention Heads as Information Movement”, and note the tensor products. The attention head function can be decomposed as:

$$h(x) = (Id \bigotimes W_O) \cdot (A \bigotimes Id) \cdot (Id \bigotimes W_V) \cdot x$$

x is the embedding of a sequence of the last few tokens, and can be viewed as a 2d matrix of dimension $n_{context} \times n_{model}$.

Given x, compute the value vector v_i for each token x_i ($v_i = W_V x_i$). Then compute result vectors, $r_i = \sum_j A_{i,j} v_j$, which depend on the “history” of previous tokens. Then, for each result r_i, compute $h(x)_i = W_O r_i$.

So, r (for $r = h(x)$) is again a $n_{context} \times n_{model}$ matrix, and we have computed the contribution of one attention head, as part of one layer.

Looking Inside the Attention Heads (cont.)

If we followed the recipe of the previous slide literally, then at each layer, we would have to re-compute $v_i = W_V x_i$ for each x_i for each head, for each previous token. This would be wasteful.

So instead, we use a larger "space" of dimension $d_{model} = n_{heads} \times d_{head}$, in which we can store the previous v_i in its own subspace. If a given layer does not act upon a subspace containing v_i, then the value vector v_i is simply carried forward to the next layer. But the dimension of each value vector, v_i, must be kept small, in order to fit inside d_{model}. See: `https://transformer-circuits.pub/2021/framework/index.html#model-details`

As before, *backpropagation* for neural nets "trains" the weights W_V, A, W_O.

Looking Inside the Attention Heads: the Query-Key circuit

(See: “Splitting Attention Head terms into Query-Key and Output-Value Circuits”)

One computes a query $q_i = W_Q x_i$. One also computes keys $k_i = W_K x_i$ for each token of the history. One then takes a dot product to compute the activation energies applied to the next “neuron”: $A = \text{softmax}(q^T k)$.

More compactly, we have: $A = \text{softmax}(x^T W_Q^T W_K x)$.

Positional Attention Heads

For intuition, one can imagine first training the attention heads, and then freezing them, and later training the MLP layers.
We will see that attention heads come primarily in two flavors: *positional heads* and *induction heads*.

Some examples of large entries QK/OV Circuit for Primarily Positional Heads

Source Token	Destination Token	Out Token	Examples
" corresponding"	*Primarily Positional*	" to", "to", " for", "markup", " with"	" corresponding to", " correspoding with"
" coinc"	*Primarily Positional*	" with", " closely", "with", " con"	" coinc[ides] with", " coinc[ides] closely"
" couldn"	*Primarily Positional*	" resist", " compete", " stand", " identify"	" couldn['t] resist", " couldn['t] stand"
" shouldn"	*Primarily Positional*	" have", " be", " remain", " take"	" shouldn['t] have", " shouldn['t] be"

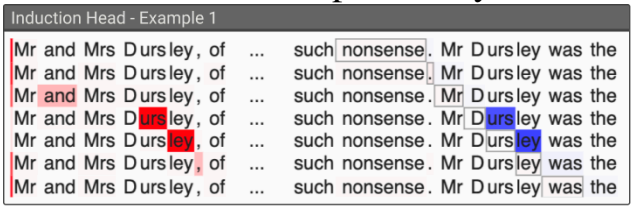
Summarizing OV/QK Matrices

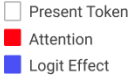
(Viewing OV/QK Matrices as bigrams (match of AB) or skip-trigrams ($A..BC$).
See article with "Interpretation as Skip-Trigrams".

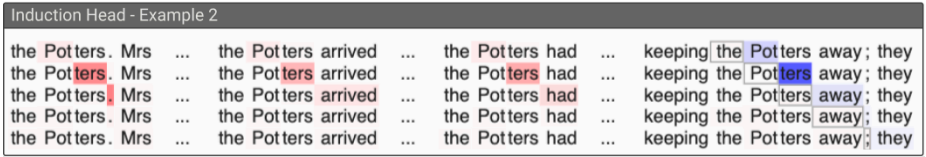
See article with "Summarizing OV/QK Matrices".
"The OV and QK matrices are extremely low-rank. They are $50,000 \times 50,000$ matrices, but only rank d_{model} (64 or 128). In some sense, they're quite small even though they appear large in their expanded form."

Looking Inside the Attention Heads

Induction heads save previously “learned” information in a safe place:

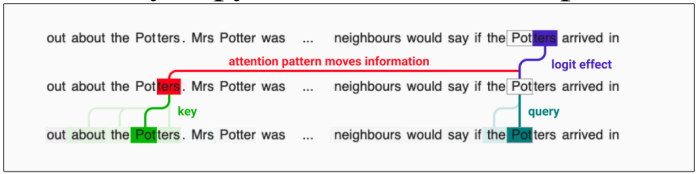


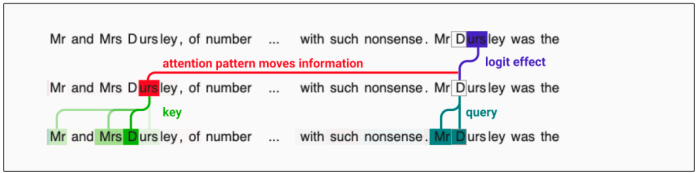




Looking Inside the Attention Heads

Induction heads move previously “learned” information, so as to be used later. They copy to an available subspace





Outline of Talk

Reviewing Transformers: Assimilating earlier lessons (cf. original article)

1. Attentions heads are independent operations adding tnto a "residual stream", $x_0, \ldots, x_n$. Each x_i has dimension d_{model}. And $x_0 = W_E t$ is the embedding of the input token t, and $T(t) = W_U x_{n-1}$ is the activation for the output token (the model's prediction of the next token). In between, there are multiple residual blocks. Each residual block takes input x_i, applies all attention heads to produce x_{i+1}, and then applies the MLP (input/hidden/output) layers to produce x_{i+2}.
2. The original transformer paper presented the circuits as: $A : x_i \to x_{i+1}$. (A "circuit" is a nonlinear "matrix", due to the logit function.) More intuitively, we can write $A = Id + W$ ("$x_{i+1} = A(x_i) = x + W(x_i)$'), where Id is the identity matrix.

Reviewing Transformers (cont.)

3. If one freezes the attention patterns (freezes the training), then attention-only models are linear paths from input to output.
4. Multiplying matrices/tensors for zero, one, or two layers reveals insights for interpretability. Instead of multiplying $A \cdot A'$, we analyze $(Id + W) \cdot (Id + W') = Id + W + W' + f(W, W')$, where $f(\cdot)$ is analyzed.
5. Attention heads include *independent* QK (query-key) and OV (output-value) circuits. QK determines the which previous token blocks to pay attent to, while OV determines the output and the subspace where it's stored if attended to. The OV circuit allows us to "copy" tokens or interences from tokens, so that the next layer needs only to refer to the immediately preceding layer.

Reviewing Transformers (cont.)

6. The matrices $W_Q^T W_K$ and $W_O W_V$ are of dimension $n_{vocab} \times n_{vocab}$, where a "vocabulary" for $W_Q^T W_K$ (and also something large for $W_O W_V$) might include 50,000 tokens. If these were of full rank, then the LLM would be "memorizing" probabilities for arbitrary pairs of tokens. This is too large to handle. LLMs guarantee that $W_Q^T W_K$ and $W_O W_V$ are of low rank by having an intermediate dimension $n_{context}$ (e.g., 128 keys or queries).
7. The components of a transformer communicate with each other by reading and writing to different subspaces of the residual stream. This provides communication channels among components such as token embedding, attention head, MLP, unembedding).

Outline of Talk

1. A Mathematical Framework for Transformer Circuits
2. A Mathematical Framework for Transformer Circuits: Takeaways
3. Progress measures for grokking via mechanistic interpretability

New section: a+b mdulo P

PROBLEM: For a fixed prime P, and two inputs, a and b, predict the output, $a + b \mod P$.

Experimental setup for $a + b \mod P$

a and b are encoded by P-dimensional one-hot vectors. (A *one-hot* vector means that a can take on any of the values $0, 1, \ldots, P-1$ by setting the appropriate one neuron out of P possibilities.) The input is a, b and $=$. (The $=$ input allows us to read the output c.) We take $P = 113$ and a one-layer ReLU transformer. (The Rectified Linear Unit is the activation function.) The token embeddings use $d = 128$, one has learned positional encodings, 4 attention heads of dimension $d/4 = 32$., and $n = 412$ hidden units in the MLP.

Logit (or Logistic) Function for Neuron Activations

The logistic function is of the form:

$$p(x) = \frac{1}{1 + e^{-(x-\mu)/s}}$$

where μ is a location parameter (the midpoint of the curve and s is a scale parameter.
See `https://en.wikipedia.org/wiki/Logistic_regression` from Wikipedia. See also: `https://en.wikipedia.org/wiki/Softmax_function#Neural_networks` and `https://en.wikipedia.org/wiki/Softmax_function#Neural_networks` for the *softmax* function used in neural net activation functions.

Trigonometric Identities

$$e^{i\theta} = \cos(\theta) + i\sin(\theta)$$
$$e^{i(a+b)} = \cos(a+b) + i\sin(a+b)$$
$$\cos(a+b) + i\sin(a+b) =$$
$$(\cos(a)\cos(b) - \sin(a)\sin(b)) + i(\sin(a)\cos(b) + \cos(b))$$

So (taking real and imaginary parts):

$$\cos(a+b) + i\sin(a+b) =$$
$$(\cos(a)\cos(b) - \sin(a)\sin(b)) + i(\sin(a)(\cos(b) + \cos(b)))$$

Basic Fourier/Harmonic Analysis

We all know about vectors and linear algebra. For example, a three-dimensional space has three basis vectors: $(1,0,0)$, $(0,1,0)$, and $(0,0,1)$. We can take inner products, project onto subspaces, etc.

A space of periodic functions (e.g., $a \mod P$) can be decomposed into an infinite number of basis functions: $\cos(\theta)$, $\sin(\theta)$, $\cos(2\theta)$, $\sin(2\theta)$, $\cos(3\theta)$, $\sin(3\theta)$, etc. Alternatively, if we allow complex numbers, then the basis functions are $e^{i\theta}$, $e^{i2\theta}$, $e^{i3\theta}$, etc. The corresponding inner product of two functions $f(\theta)$ and $g(\theta)$ is:

$$\frac{\int_0^{2\pi} f(\theta)\, g(\theta)\, d\theta}{2\pi}$$

Harmonic Analysis: A Real-World Example

[Click on links for deeper reading.]

- The Tacoma Bridge Collapse
- Meanwhile, Professor F. B. Farquharson continued wind tunnel tests. He concluded that the “cumulative effected of undampened rhythmic forces” had produced “intense resonant oscillation”.

In fact, the physics analysis later revealed a different reason for the collapse, but resonant frequency and the speed of the wind still made a nice story:

- Science Busts The Biggest Myth Ever About Why Bridges Collapse

Mechanistic Interpretation: Algorithm of one-layer transformer

Given two numbers a and b, the model projects each point onto a corresponding rotation using its embedding matrix. Using its attention and MLP layers, it then composes the rotations to get a representation of $a+b$ mod P. Finally, it "reads off" the logits for each $c \in \{0,1,\ldots,P-1\}$, by rotating by $-c$ to get $\cos(\omega(a+b-c))$, which is maximized when $a+b \equiv c \mod P$ (since ω is a multiple of $2\pi/P$).

Evidence for Given Mechanistic Interpretation

(1) the network weights and activations exhibit a consistent periodic structure;

(2) the neuron-logit map W_L is well approximated by a sum of sinusoidal functions of the key frequencies, and projecting the MLP activations onto these sinusoidal functions lets us "read off" trigonometric identities from the neurons;

Evidence for Given Mechanistic Interpretation (cont.)

(3) the attention heads and MLP neuron are well approximated by degree-2 polynomials of trigonometric functions of a single frequency; and

(4) ablating key frequencies used by the model reduces performance to chance, while ablating the other 95% of frequencies slightly improves performance.

Definition: Ablating a frequency means removing that frequency in the Fourier decomposition.