28/8/20

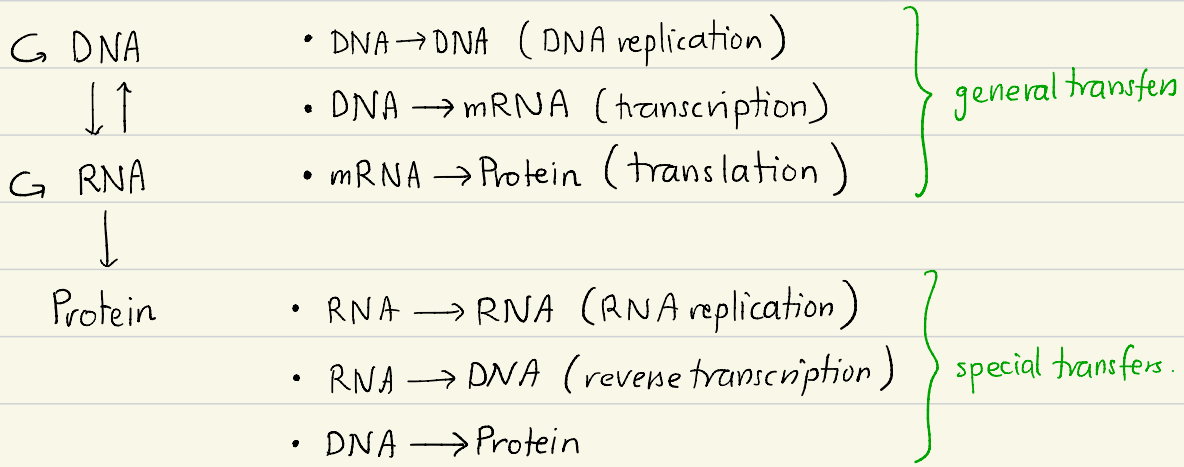
C|t|

(1)

Cell Learning Theory 1

Notes on joint work with Michael Chappuis. To set up the problem we follow the "connectivity map" paper [L] and consider cell lines, perturbagens and gene expression. A cell line is a culture of animal cells that can be propagated repeatedly (perhaps indefinitely) and are thus convenient for in vitro studies. Example: the immortalised mouse Sertoli cell line MSC-1. Cell lines are often used in place of primary cells to study biological processes, but some care is required when generalising to primary cells.

The central dogma of molecular biology (due to Crick) states that DNA/RNA are to proteins what syntax is to (operational) semantics. More concretely, generally speaking there are three classes of information carrying biopolymers in living organisms: DNA, RNA and proteins. The transfers of information that are known to occur:



It may be argued that computation is the most fundamental aspect of life [A]. Regarding the subset of computations taking place within cells, the physical mechanisms which support computation include gene regulatory networks. Such a network is a collection of regulators that govern the gene expression level of mRNA and proteins. These regulators can be DNA, RNA, protein and "complexes" of these.

for example some proteins serve to activate other genes, and these transcription factors (TFs) are the main players in regulatory networks. By binding to the promoter region at the start of other genes they turn them "on". More precisely, for transcription to take place the enzyme that synthesises RNA, RNA polymerase must attach to the DNA near a gene and it is the transcription factors which recruit RNA polymerase. A gene may be require multiple transcription factors. The same response element sequence may be located in the control regions of different genes, so that these genes may be stimulated simultaneously by a single transcription factor.

A transcription program is a set of functionally related and co-regulated genes [PM]. Such programs can be independently regulated using combinations of TFs. We can roughly consider four classes of TFs

- Class A : "housekeeping" expressed in most or all cell types.
- Class B : "signal dependent" present in latent form in un-stimulated cells. Upon activation these TFs quickly activate or repress their target genes. These TFs are pre-made (PRGs)
- Class C expression induced by Class B TFs, not pre-made (SRGs)
- Class D lineage-specific, express cell-type-specific genes.

Primary Response Genes (PRGs) can be induced rapidly and do not require de novo translation for their induction, whereas Slow Response Genes (SRGs) are induced slowly, require de novo translation for their induction and are regulated by PRG protein products [FSR].

Note To give an idea, in [MARS, p.8] there are ~20,000 genes and ~100,000 cells, and [N] estimates ~1600 TFs in the human genome.

Simple logic model Suppose G is a set of genes, and for each gene g let

$$\text{active}(g) = \text{gene } g \text{ is active}$$

Assume for each $g \in G$ we have an encoding $q(g)$ (say an integer) of the response element for g (assume there is only one per gene) and an encoding $k(g)$ of the transcription factor. Then the previous paragraph may be encoded by the axiom

$$\exists\, g' \in G \left(\text{active}(g') \wedge k(g') = q(g) \right) \Longrightarrow \text{active}(g). \qquad (*)$$

Suppose given some intial facts $F_0 = \{ \text{active}(s) \}_{s \in S}$ for $S \subseteq G$. which propositions $\text{active}(g)$ are logical consequences of these initial facts? If G is finite then the following iterative procedure (forward chaining [RN, §9.3]) reaches a fixed point: set F_n = all statements deducible from F_{n-1} and the axiom schema $(*)$ in one step.

$$\begin{array}{l} \text{active}(g_1) = \text{False} \\ \vdots \\ \text{active}(g_n) = \text{True} \end{array} \quad \xrightarrow{\text{iterative procedure}} \quad \begin{array}{l} \text{active}(s_1) = \text{True} \\ \vdots \\ \text{active}(s_n) = \text{True} \end{array}$$

Inference problem Suppose examples of the following form are given (g_0, Q) where $g_0 \in G$ and $Q \subseteq G$ is finite, to be read as "g_0 is initially active and at time ∞ the set Q (and possibly other genes) are active". In principle we can run fixed point iteration starting with $\{g_0\}$ and see if Q becomes active to derive a distribution over axioms $(*)$.

Assuming G is large this is impossible as the set of axioms is large. So instead we could formulate a nested inference problem: first given Q predict $P \subseteq G$ finite and run the fixed point iteration just for those axioms involving $\{g_0\} \cup P \supseteq Q$.

Simple vector model Suppose G is a set of genes and for each $g \in G$ we have an encoding or representation of the proposition active(g) of the following form: a fixed unit vector $T \in H$ and a vector $v(g) \in H$ such that our belief in active(g) is $|\langle \hat{v}(g), T \rangle| \in [0,1]$ where $\hat{v}(g) = v(g)/\|v(g)\|$. Here H is some Hilbert space.

Suppose also we have encodings $k(g), q(g) \in H$ so that $k(g) = q(g)$ is replaced by $\hat{k}(g) \approx \hat{q}(g)$. Then a "soft" encoding of (*) is

$$\exists g' \in G \left(\hat{v}(g') \approx T \wedge \hat{k}(g') \approx \hat{q}(g) \right) \Rightarrow \hat{v}(g) \approx T \qquad (3.1)$$

A simple analog of the fixed point iteration in the logic setting is the rule which starts off with a set $S \subseteq G$ of active genes, sets $v(s) = T$ for $s \in S$ and $v(g) = \bot$ for $g \notin S$ (here T is "true" and $\bot$ is "false" and we assume $\langle \bot, T \rangle = 0$). In the iterative step we look at each g, and if any g' with $\langle \hat{q}(g), \hat{k}(g') \rangle$ large has $\hat{v}(g') \approx T$ we should move g closer to T. One rule achieving this is

$$v(g) \longleftarrow v(g) + \sum_{g' \in G} \frac{e^{\langle q(g), k(g') \rangle}}{\sum_h e^{\langle q(g), k(h) \rangle}} v(g') \qquad (3.2)$$

That is,

$$\Delta v(g) \approx \frac{1}{Z} \sum_{v(g') \approx T} \delta\left(\hat{q}(g) \approx \hat{k}(g') \right) \cdot T + \frac{1}{Z} \sum_{v(g') \approx \bot} \delta\left(\hat{q}(g) \approx \hat{k}(g') \right) \bot \qquad (3.3)$$

Assuming some kind of normalisation, this has the same fixed point.

With some added complexity this "simple vector model" is a Transformer, with each iteration of the Transformer block representing one iteration. The genes or "entities" play the role of words in language modelling and we have the following dictionary

Transformer	Gene regulation
entity	gene
query	encoded response element
key	encoded transcription factor
value	encoded activation

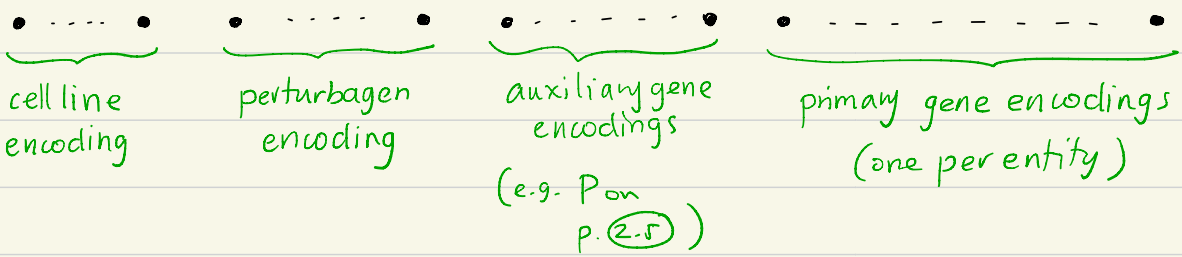
Some thoughts

- The "switch $\top$ to $\bot$ via iteration to a fixed point" makes each entity behave like a variable with "name" given by the query and contents by the value vector, and iteration simulates substitution (substitution of a complex term might be simulated using multiple heads)

Sketch of cellular learning model (v1)

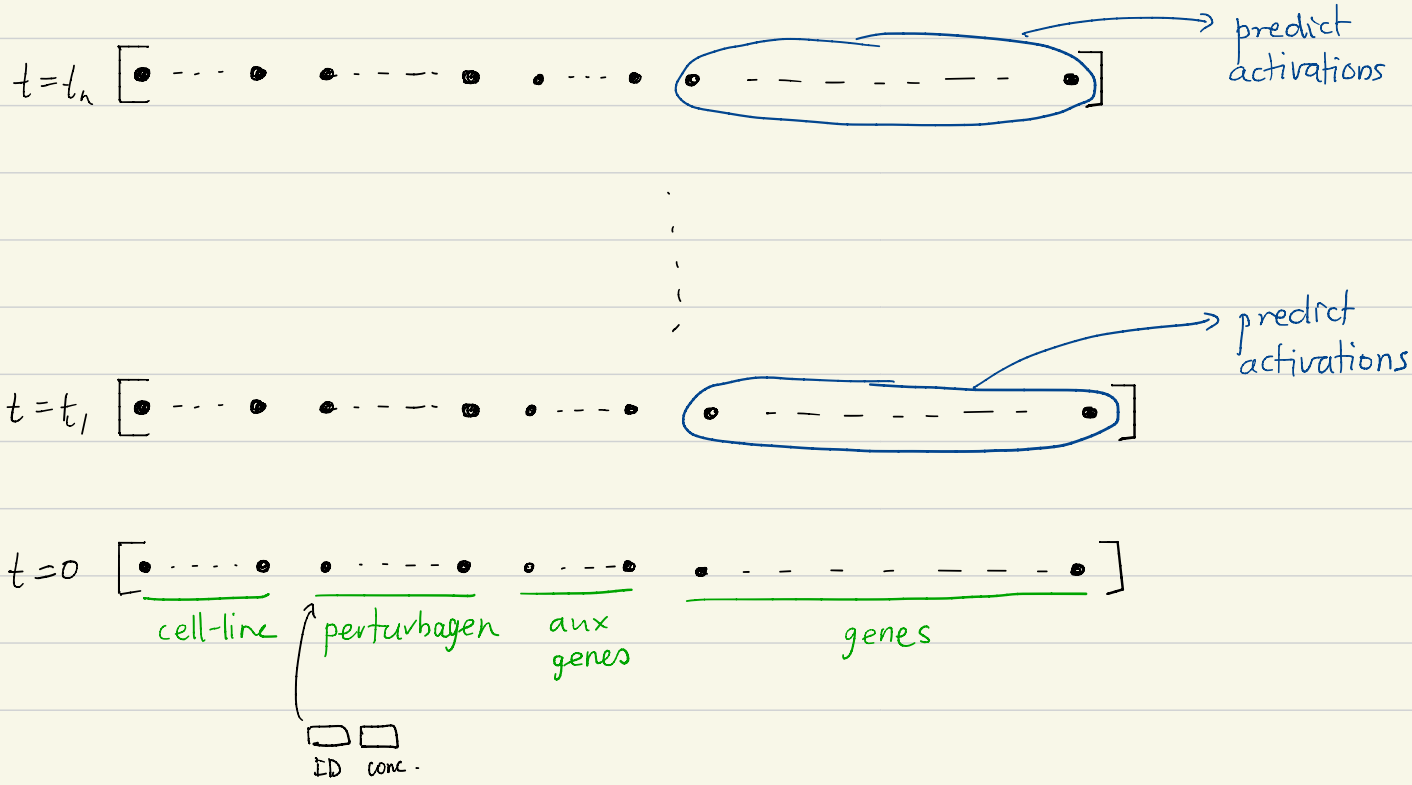
The task of the model is to learn the algorithm underlying the changes in mRNA and protein concentrations. We assume a correspondence between depth and wall time in the cell evolution, and suppose that each experiment involves a cell line, a perturbagen at some concentration, and measurements of mRNA populations at some collection of times $t_1 < \cdots < t_n$.

State There is a sequence of entities, divided into three groups



Note that the perturbagen entities have a learned encoding of (perturbagen, concentration) pairs.

Loss Assume some mapping F from times to depths $F(t_1) < \cdots < F(t_n)$



The auxiliary genes are meant to be "prepared" by the context of cell-line and perturbagen to include whichever genes are computationally relevant.

Remarks (i) We bin the concentration levels and represent them by some number of tokens $c_1, \ldots, c_n$.

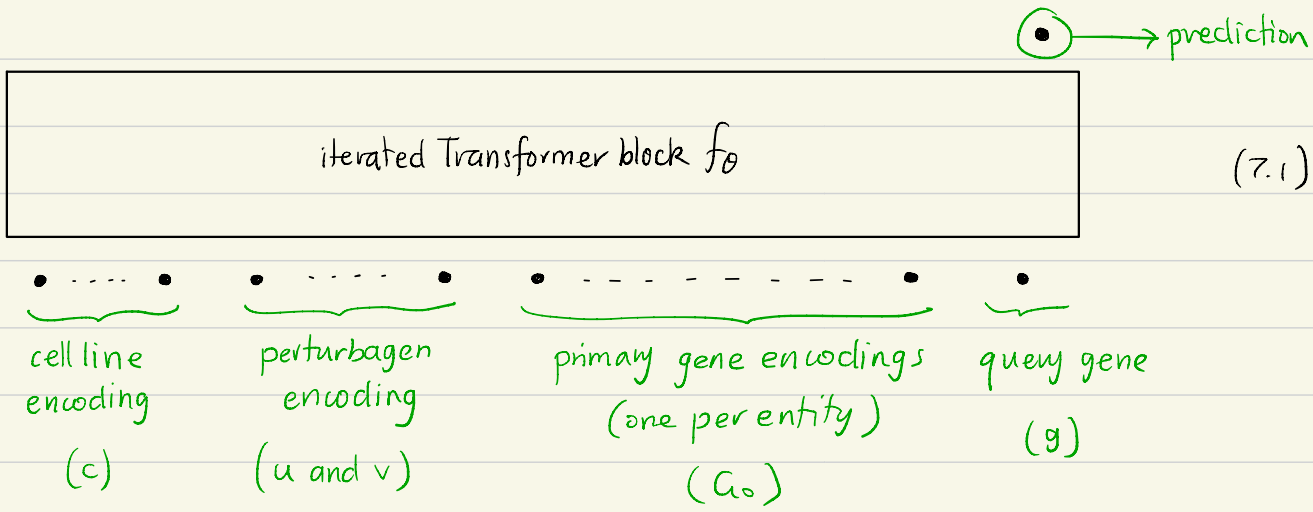
Sketch of a cellular learning model (v2)

The key difference between modelling gene regulation programs and natural language is that the probability distribution over possible next words in a sentence (activation of a gene) depends on the other words in the context (active genes) but not on other words (inactive genes). However given some initial set G_0 of active genes, the probability that a given gene g will be active at time t depends <u>indirectly</u> on G_0 via genes $h \notin G_0$ which are induced by the genes in G_0.

If we accept the basic model outlined above, this means the "new problem" beyond language modelling to be solved is to predict the "auxiliary genes" P determined by G_0 (that is, so that message-passing/attention on $P \cup G_0$ predicts well the activation level of any g). To this end we divide into two networks (partly inspired by the original AlphaGo architecture and its bootstrap value function).

Static network (f_0)

Given a cell-line c, a perturbagen u, a concentration v, a list of active genes G_0, and a gene g, predicts the probability that when the given perturbagen is introduced to the given cell-line at the given concentration in a state where the genes in G_0 are active, at <u>some time</u> between this moment and the end of measurements g will be active.



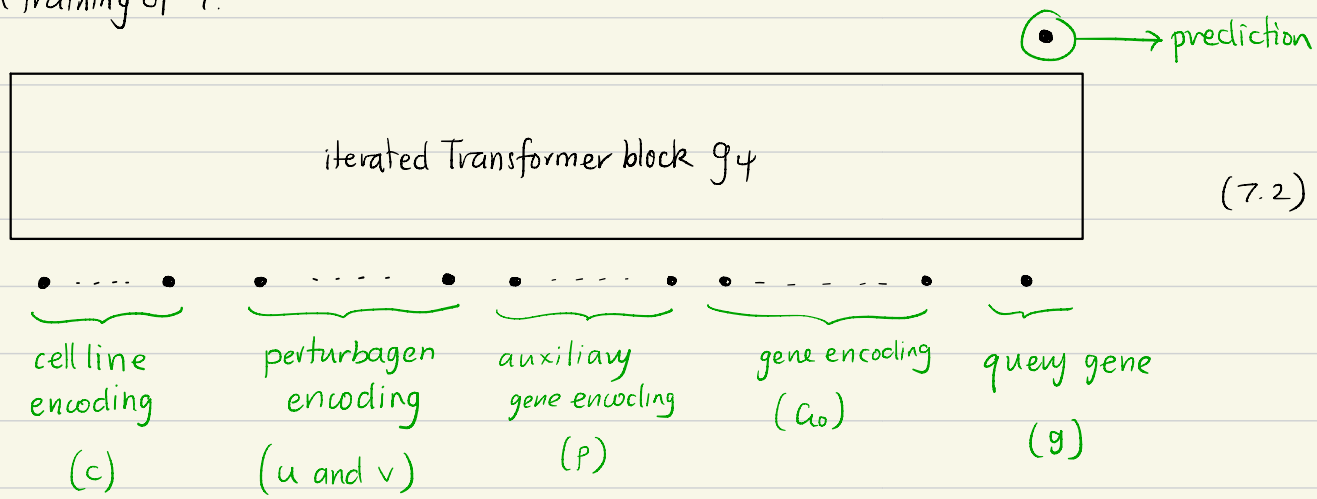
(7.1)

We "pretrain" this static network on our dataset, in the process generating embeddings for cell lines, perturbagens, concentrations and genes. We call these the static network embeddings. The network f_θ learns which genes are relevant to the dynamics, but not the dynamics itself.

Note: probably we want f_θ to be a generative model not a classifier, see usage below. In this case we would use the Transformer in its standard encoder-decoder form and predict a set of activated genes.

Dynamic network (g_ψ)

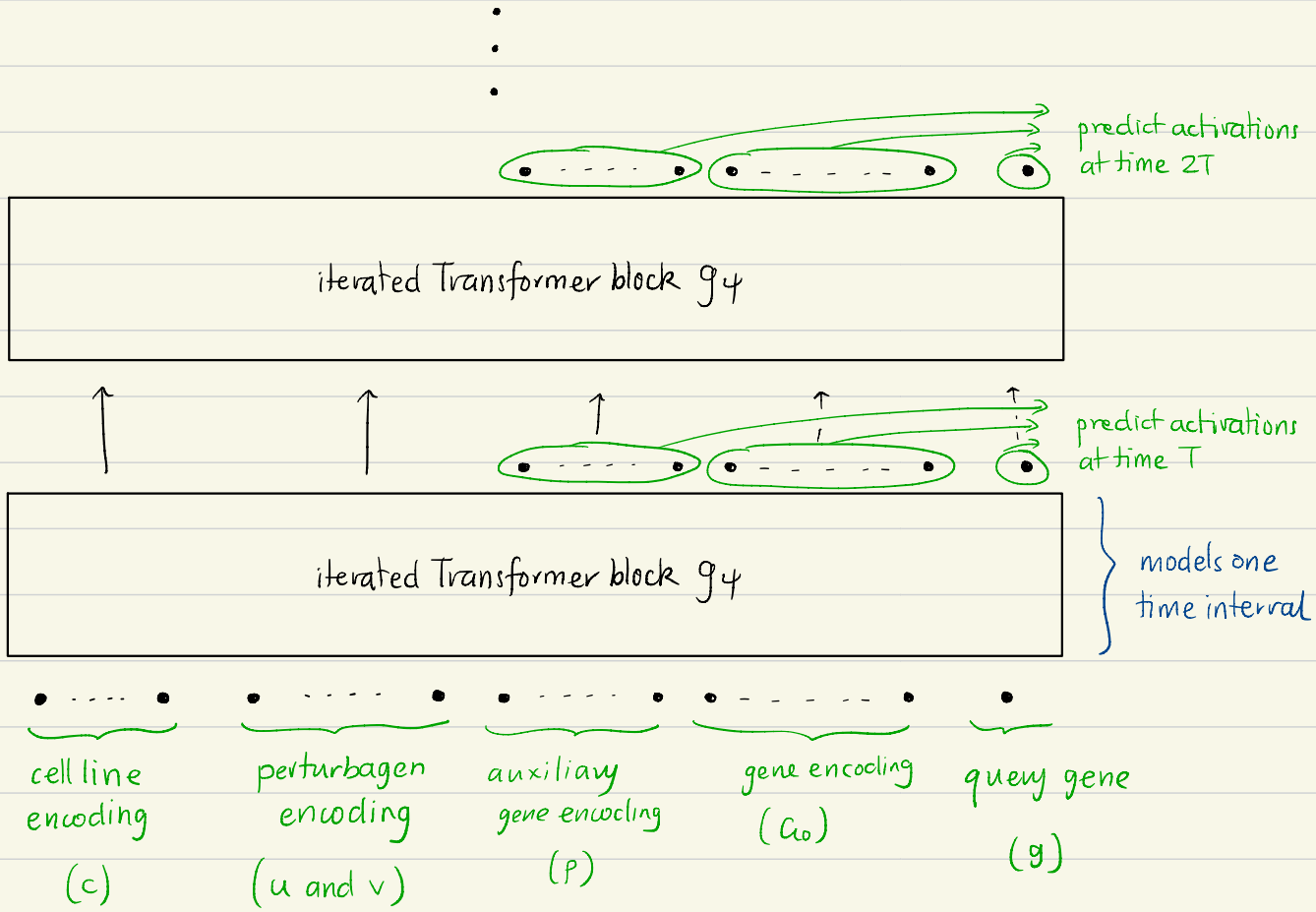
Given a cell-line c, perturbagen u and concentration v, a list of active genes G_0 and a "query" gene g, predict the activation level of g at some particular time. The embeddings are initialised to the static embeddings, but are allowed to vary with training of ψ.



(7.2)

Here we use the network f_θ to sample a set P of auxiliary/relevant genes. To predict activation of g at time NT we use N iterations of g_ψ. The idea is that the Q, K, V matrices associated to each gene are a "microscopic program" and these microcodes are made to interact via self-attention in g_ψ. To some extent this interaction must model the actual gene expression program if g_ψ can predict the activation levels of arbitrary genes.

To make full use of the training data, and to realise the idea that the attention microcodes should model the underlying gene expression program, we ask the network to predict not just the final activation of the query gene but final and intermediate activations of all genes in $P \cup G_0 \cup \{g\}$ at some standard time points corresponding to measurements.



Contrast to the literature

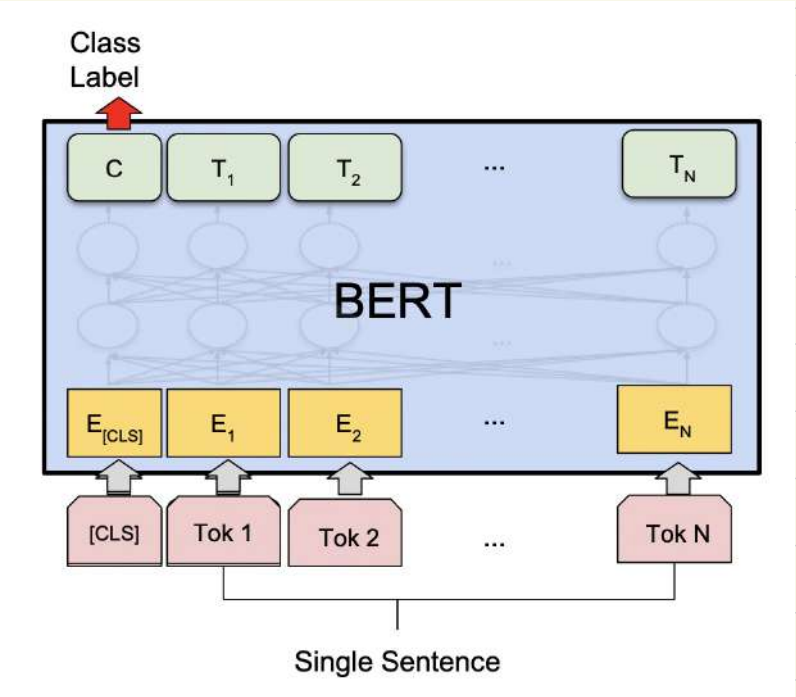
We take a "program synthesis" point of view, which seeks to model gene activations over time as measurements of an ongoing computational process. The model is the time-evolving representations in a Transformer network, thought of as "registers" (genes, entities) interacting via attention (message passing). This is in contrast to papers like [MARS] which are focused on classification on the basis of learned representations of individual gene-cell activation matrices.

Tokenisation

This model faces a similar problem to Transformers in NLP, namely there are too many genes (words). If the number of genes is on the order of 50,000 then this is compatible with standard Transformers on modern hardware (GPT has ~40,000 tokens) but if it is larger then strategies like BPE may be used (https://huggingface.co/transformers/tokenizer_summary.html).

Transformers do classification

To predict the activation level of a gene we use the final hidden state of the rightmost "query" entity, which is fed to a classification layer. This follows standard practice [BERT]:



Position encoding

Transformers only "see" the order of their inputs via position encoding, and we choose these so that the network is invariant to permutations <u>within</u> the groups of input entities in (7.1), (8.1) but so that the groups appear in the specified order.

Higher-order interactions

The analogies between the attention mechanism and "logic" models at the beginning of this note, and elaborated further below, suggest that the architecture is a good fit for genes requiring a single TF to be induced. For genes requiring multiple TFs we are in the situation of [25] where there is preliminary evidence that higher-order attention can help Transformers model higher-order interactions.

Reasoning with Transformers

In many ways the quest to realise reasoning within artificial neural networks parallels Boole's attempt to model human reasoning by algebra, which was the beginning of modern logic [Boole]. This influence is strong (and perhaps unconscious) in e.g. [H].

> 6. It is designed, in the next place, to give expression in this treatise to the fundamental laws of reasoning in the symbolical language of a Calculus. Upon this head it will suffice to say, that those laws are such as to suggest this mode of expression, and to give to it a peculiar and exclusive fitness for the ends in view. There is not only a close analogy between the operations of the mind in general reasoning and its operations in the particular science of Algebra, but there is to a considerable extent an exact agreement in the laws by which the two classes of operations are conducted. Of course the laws must in both cases be determined independently; any formal agreement between them can only be established *à posteriori* by actual comparison. To borrow the notation of the science of Number, and then assume that in its new application the laws by which its use is governed will remain unchanged, would be mere hypothesis. There exist, indeed, certain general principles founded in the very nature of language, by which the use of symbols, which are but the elements of scientific language, is determined.
>
> [Boole]

> Ultimately, major progress in artificial intelligence will come about through systems that combine representation learning with complex reasoning. Although deep learning and simple reasoning have been used for speech and handwriting recognition for a long time, new paradigms are needed to replace rule-based manipulation of symbolic expressions by operations on large vectors[101]. ■
>
> LeCun, Bengio, Hinton [LBH].

There is a way of thinking about attention and Transformers which draws out the continuities with Boole's work, emphasising the central role of <u>algebra in spaces of representations</u>. In Boole's work a proposition p about the world is an element of an algebra (thought of as an operator on some hypothetical state which projects onto configurations satisfying p) and conjunction is multiplication. In Transformers a relationship between entities is represented by a "head" [2S, Remark 2.1] which is a transformation of entities to vectors in an inner product space H.

The <u>Clifford algebra</u> $Cl(H)$ is an associative unital $\mathbb{R}$-algebra generated by H subject to the relations $xy + yx = 2\langle x, y\rangle \cdot 1$ where $\langle , \rangle$ is the pairing (e.g. dot product on $H = \mathbb{R}^d$). The underlying vector space of $Cl(H)$ has a $\mathbb{Z}$-grading in the sense that

$$Cl(H) = Cl(H)_0 \oplus Cl(H)_1 \oplus \cdots$$

$$\psi$$

$$A = [A]_0 + [A]_1 + \cdots \qquad 1 \in Cl(H)_0, \quad [A]_i \in Cl(H)_i$$

<u>Example</u> $H = \mathbb{R}^2$, $\langle , \rangle$ is the dot product then for $x = x_1 e_1 + x_2 e_2$, $y = y_1 e_1 + y_2 e_2$ in H, with $x_i, y_i \in \mathbb{R}$ for $1 \le i \le 2$, in $Cl(H)$

$$\begin{aligned} xy &= (x_1 e_1 + x_2 e_2)(y_1 e_1 + y_2 e_2) \\ &= x_1 y_1 e_1^2 + x_1 y_2 e_1 e_2 + x_2 y_1 e_2 e_1 + x_2 y_2 e_2^2 \end{aligned}$$

But $2e_1^2 = 2\langle e_1, e_1\rangle \cdot 1 = 2 \cdot 1$ so $e_1^2 = e_2^2 = 1$, and

$e_1 e_2 + e_2 e_1 = 2\langle e_1, e_2\rangle \cdot 1 = 0$ so

$$xy = \underbrace{(x_1 y_1 + x_2 y_2) \cdot 1}_{[xy]_0 \text{ is the dot product of } x, y \text{ in } Cl(H)_0 \cong \mathbb{R}.} + (x_1 y_2 - x_2 y_1) e_1 e_2$$

Let x, y be learned representations of entities in H. The Transformer represents the inductive bias that a relationship between these entities exists to the extent that the dot product $\langle x, y \rangle$ is large and positive. We may think of this as a property of the product xy in $Cl(H)$, so that measuring the degree of relation between the entities is factored into several steps

$$\begin{array}{ccccccc} H \otimes H & \hookrightarrow & Cl(H) \otimes Cl(H) & \xrightarrow{\text{product}} & Cl(H) & \xrightarrow{[-]_0} & Cl(H)_0 \cong \mathbb{R} \\ x \otimes y & & x \otimes y & & xy & & [xy]_0 = \langle x, y \rangle \cdot 1 \end{array} \qquad (13.1)$$

embed into an algebra — multiply in that algebra — extract measure of relationship

Of course this is unnecessarily complex in the case of binary relations, but just as in Boole's work the virtue of an algebraic perspective on reasoning becomes more apparent when more entities are involved. In [25] it is explained how to use the same idea to model three-way (and higher-order) interactions in the setting of Transformers.

Transformers learn to route and transform information using the geometric algebra of dot products, and the success of these models across many difficult tasks (including reaction to units in AlphaStar and protein folding in AlphaFold) makes it clear that this algebra of attention supports some form of reasoning with representations (one definition of reasoning is deductions made on the basis of algebraic operations on a representation of existing knowledge [Bo]. Indeed that is precisely Boole's idea!). This arguably goes towards realising the replacement of "rule-based symbolic expressions by operations on large vectors" as in [LBH]. More complex algebras may support more complex reasoning.

This concludes the sketch on our approach to unifying logic with deep learning via algebra. Under the Curry-Howard correspondence this also relates to Transformers modelling code (be it computer code or gene regulation programs).

References

[L] J. Lamb et al "The connectivity map : using gene-expression signatures to connect small molecules, genes and disease" Science 2006.

[A] H. Akhlaghpour "A theory of natural universal computation through RNA" arXiv: 2008.08814, 2020.

[RN] S. J. Russell, P. Norvig "Artificial intelligence" 3rd edition

[PM] S. D. Pope and R. Medzhitov "Emerging principles of gene expression programs and their regulation" Molecular Cell 71, 2018.

[FSR] T. Fowler, R. Sen, A. L. Roy "Regulation of primary response genes" Mol. Cell 2011.

[MARS] M. Brbic et al "MARS : discovering novel cell types across heterogeneous single-cell experiments" Nature Methods

[N] A. H. M. Ng et al "A comprehensive library of human transcription factors for cell fate engineering" Nature biotechnology.

[BERT] J. Devlin et al "BERT: pre-training of deep bidirectional Transformers for language understanding" 2018.

[2S] J. Clift, D. Doryn, D. Murfet and J. Wallbridge "Logic and the 2-simplicial Transformer" ICLR 2020.

[C] J. Clauwaert, W. Waegeman "Novel Transformer networks for improved sequence labeling in genomics" preprint Nov 2019.

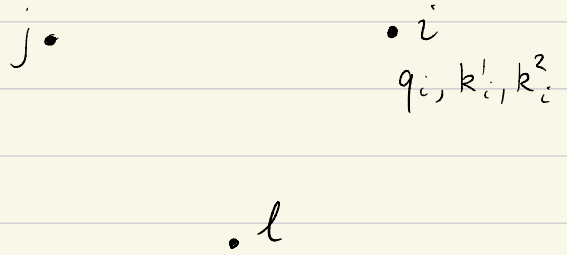
[LBH] Y. LeCun, Y. Bengio and G. Hinton "Deep learning" Nature 2015.

[Bo] L. Bottou, "From machine learning to machine reasoning" arXiv: 1102.1808.

[H] G. Hinton "Aetherial symbols" talk 2015.

[Boole] G. Boole "An investigation of the laws of thought" 1854.

Q/ — Time points? Adaptive computation time



$q_i k^1_j k^2_l \in Cl(H)$ — multiplication in $Cl(H)$

η → scalar

$\eta : Cl(H) \longrightarrow \mathbb{R}$

$\eta(q_i k^1_j k^2_l)$ is the analogue of $\langle q_i, k_j \rangle$ in standard attention