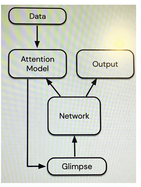
Deepmind - attention & memory

Sunday, 14 March 2021 21:50

- Deep nets naturally learn a form of *implicit attention* where they respond more strongly to some parts of the data than others.
  - → We can visualize this by looking at the network Jacobian; the sensitivity of the network w.r.t. the inputs: $J_{ij} = \frac{\partial y_i}{\partial x_j}$
  - e.x. Dueling network for DRL
- Attention and memory in RNN:
  - Sequential attention shows which past inputs the RNN remember when predicting current outputs.
  - Sequential Jacobian is the set of derivatives of one network output w.r.t. all the inputs (it is a time series): $J_k^t = \left(\frac{\partial y_k^t}{\partial x^1}, \frac{\partial y_k^t}{\partial x^2}, \dots \right)$
- Advantages of explicit attention:
  1. Computational efficiency.
  2. Scalability.
  3. Sequential processing of static data.
  4. Easier to interpret.
- Neural attention models:
  - The network produces an extra output vector used to parametrise an attention model.
  - The attention model then operates on data to create a fixed-size glimpse vector that is passed to the network as input at the next time step.
    - → the complete system is recurrent even if the network isn't.
  - Attention models generally work by defining a probability distribution over glimpses g of the data x given a set of attention outputs a from the network: $Pr(g|a)$



- Attention with RL:
  - We can treat the distribution over glimpses g as a stochastic policy π_a, sample from it, and use REINFORCE (with reward $R = L$ task loss induced by the glimpse) to train the attention model:
    $\pi_a = Pr(g_k|a)$; $R = E_{g\sim\pi_a}[\log \pi_a L(g)]$; $\nabla_a R = E_{g\sim\pi_a}[\nabla_a \log \pi_a L(g)]$
  - In general we can use RL methods for supervised tasks any time some modules in the networks is non-differentiable.
- Soft (differentiable) attention:
  - Soft attention can be trained end-to-end with backprob.
  - → we use attention parameters to determine a distribution $Pr(g|a)$ as before, only now we take an expectation over all possible glimpses instead of a sample: $g = \sum_{g\in x} g' Pr(g'|a)$
  - → differentiable w.r.t: $\nabla_a g = \sum_{g\in x} g' \nabla_a Pr(g'|a)$
- Associative Attention:
  - Instead of attending by position, we can attend by content: a key vector k is compared to all x_i in the data using some similarity function S which used to define $w_i = \frac{\exp S(k,x_i)}{\sum_j \exp S(k,x_j)}$
    S can be learned or fixed (dot product / cosine)
    - → yields to Multidimensional, feature-based lookup
  - Given w_i we can sum over the data directly to get an attention readout v: $v = \sum_i w_i x_i$
- Introspective Attention:
  - It's useful to attend to the network's internal state or memory: Memory = attention through time.
  - With internal information we can do selective writing as well as reading allowing the network to iteratively modify its state.

- Selective attention:
  - We want to focus on the parts of the memory the network will read and write to: need an introspective attention model.
  - We use the controller outputs to parametrise a distribution (weighting) over the rows (locations) in the memory matrix.
  - The weighting is defined by two main attention mechanisms: one based on content and one based on location.
  1. Addressing by content:
     A key vector k is emitted by the controller and compared to the content of each memory location $M[i]$ using a similarity measure $S(\cdot,\cdot)$ (e.g. cosine distance) the normalized with a softmax.
     A sharpness β is used to narrow the focus.
     Finds the memories closest to the key: $w[i] = \frac{\exp(\beta S(k, M[i]))}{\sum_j \exp(\beta S(k, M[j]))}$
  2. Addressing by location:
     The controller outputs a shift kernel s (e.g. a softmax on $[-n,n]$) which is convolved with a weighting w to produce a shifted weighting $\hat{w}$: $\hat{w}[i] = \sum_j w[j]\, s(i-j)$
  - → Data structure and accessors:
    The combination of addressing mechanisms allows the controller to interact with the memory in several distinct modes, corresponding to different data structures and accessors.

Content key only	— memory is accessed like an associative map
Content and location	— key finds an array, shift indexes into it
Location only	— shift iterates from the last focus

  - Reading and Writing:
    Once the weighting are defined, each read head returns a read vector r as input to the controller at the next time step
    $r = \sum_i w[i] M[i]$
    Each write head recieves an erase vector e and an add vector a from the controller and resets then writes to modify the memory (like LSTM):
    $M[i] \leftarrow M[i]\,(1 - w[i]e) + w[i]a$
- Differentiable Neural Computers:
  DNC is a successor architecture to Neural Turing Machine with new attention mechanisms for memory access.
- Self attention:
  Transformer Networks: instead of a controller emitting a query, every vector in the input sequence is compared with every other
  Anarchist Attention Attention $(Q,k,V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V$
  Attention is All You Need [Vaswani '17]
  The Annotated Transformer nlp.seas.harvard
  - Universal Transformers [Dehghani '19]
    Tying the weights at each transform makes the system like an RNN in depth instead of time: variable runtime, recursive transform

Summary

- Selective attention appears to be as useful for deep learning as it is for people
- Neural nets always have **implicit attention**, but we can also add **explicit attention** mechanisms
- These can be **stochastic** and trained with **reinforcement learning**
- Or **differentiable** and trained with ordinary **backdrop**
- We can use attention to attend to **memory** as well as directly to **data**
- Many types of attention mechanism (**content, spatial, visual, temporal..**) can be defined
- Can get great results in sequence learning *just using attention* (**transformers**)