Lecture 13: Neural Networks and Transformers

CS4787/5777 — Principles of Large-Scale ML Systems

Review: Linear models and neural networks. From the homeworks and projects you should all be familiar with the notion of a linear model hypothesis class. For example, for multinomial logistic regression, we had the hypothesis class

$$h_W(x) = \operatorname{softmax}(Wx).$$

This is a specific example of a more general linear model of the form

$$h_W(x) = \sigma(Wx)$$

for some inputs $x \in \mathbb{R}^d$, matrix $W \in \mathbb{R}^{D \times d}$, and function $\sigma : \mathbb{R}^D \to \mathbb{R}^D$. Many important methods in machine learning use linear model hypothesis classes, including linear regression, logistic regression, and SVM.

One naive way that we can combine two hypothesis classes is by *stacking* or *layering* them. If I have one class of hypotheses $h^{(1)}_{W_1}$ that maps from $\mathbb{R}^{d_0}$ to $\mathbb{R}^{d_1}$ and a second class of hypotheses $h^{(2)}_{W_2}$ that maps from $\mathbb{R}^{d_1}$ to $\mathbb{R}^{d_2}$, then I can form the layered hypothesis class

$$h_{W_1,W_2}(x) = h^{(2)}_{W_2}(h^{(1)}_{W_1}(x))$$

that results from first applying $h^{(1)}$ and then applying $h^{(2)}$. Intutively, we're first having $h^{(1)}$ make a prediction and then using the result of that prediction as an input to $h^{(2)}$ to make our final prediction. If both our consituent hypothesis classes are linear models, we can write this out more explicitly as

$$h_{W_1,W_2}(x) = \sigma_2(W_2 \cdot \sigma_1(W_1 x)).$$

Of course, we don't need to limit ourselves to layering just two linear classifiers. We could layer as many as we want. For example, if we had $\mathcal{L}$ total layers, then our hypothesis would look like

$$h_{W_1,W_2,\ldots,W_l}(x) = \sigma_l(W_l \cdot \sigma_{l-1}(W_{l-1} \cdots \sigma_2(W_2 \cdot \sigma_1(W_1 x)) \cdots)).$$

We can write this out more generally and explicitly in terms of a recurrence relation.

Typical runtime cost:

$$o_0 = x$$

$$\forall l \in \{1, \ldots, \mathcal{L}\}, \quad a_l = W_l \cdot o_{l-1} + b_l$$

$$\forall l \in \{1, \ldots, \mathcal{L}\}, \quad o_l = \sigma_l(a_l)$$

$$h_{W_1,b_1,W_2,b_2,\ldots,W_l,b_l}(x) = o_{\mathcal{L}}.$$

where $a_l, o_l \in \mathbb{R}^{d_l}$, and here we've also added an explicit *bias* parameter $b_l \in \mathbb{R}^{d_l}$ to each layer. This type of model is called a *multilayer perceptron* (MLP), *artificial neural network* (ANN), or *deep neural network* (DNN). (Specifically, it's a type of deep neural network called a *feedforward neural network*.) Here, the functions σ_l are called the *activation functions* and are almost always chosen to **operate independently along each dimension**; that is (with abuse of notation)

$$(\sigma_l(x))_i = \sigma_l(x_i).$$

Note that this is not true for the softmax, but it's true about pretty much every other major activation function.

Variants of neural networks:

- *Residual neural networks* include feedback connections in which the outputs of the model are fed back into itself.
- *Convolutional neural networks* restrict some of the linear transformations W_l to be members of some subset of linear transformations, typically convolutions with some filter.
- *Recurrent neural networks* repeat the same layers to process a sequence.
- *Transformers* use attention blocks to process sequences and spatially/temporally structured data in a unified way.

Transformers. Designed to process sequential data, but can generalize to any sort of structured data.

Represents an example as a matrix in $\mathbb{R}^{n \times d}$ where n is the *sequence length* (a.k.a. n "tokens") and d is the *representation dimension*. Most characteristic layer: *attention layer* (more formally, "Scaled Dot-Product Attention"). Given input activation matrices $Q \in \mathbb{R}^{n \times d_k}$ (the "query" matrix), $K \in \mathbb{R}^{n \times d_k}$ (the "key" matrix), and $V \in \mathbb{R}^{n \times d_v}$ (the "value" matrix), the attention layer outputs

$$\mathrm{Attention}(Q, K, V) = \mathrm{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V,$$

where this softmax applies along the rows of the matrix (i.e. each row of $\mathrm{softmax}(\cdot)$ sums to 1). You can think of this as a "soft" or "weighted" lookup. This formulation lets every token (every sequence element) look up into every other one: if we want to restrict this, we can use an *attention mask* $M \in \mathbb{R}^{n \times n}$, usually with elements in $\{-\infty, 0\}$, and set

$$\mathrm{MaskedAttention}(Q, K, V) = \mathrm{softmax}\left(\frac{QK^T}{\sqrt{d_k}} + M\right)V.$$

This "zeros out" the entries of $\mathrm{softmax}(\cdot)$ for which $M_{ij} = -\infty$.

Multiple attention layers are combined together to form a *multi-head attention layer*. Such a layer with h "heads" takes as input tensors $Q \in \mathbb{R}^{n \times h \times d_k}$, $K \in \mathbb{R}^{n \times h \times d_k}$, and $V \in \mathbb{R}^{n \times h \times d_v}$, and outputs a tensor of size $(n \times h \times d_v)$ such that

$$\mathrm{MultiHeadAttention}(Q, K, V)_{:,i,:} = \mathrm{MaskedAttention}(Q_{:,i,:}, K_{:,i,:}, V_{:,i,:});$$

that is, it's just h attention layers running in parallel along the head dimension.

A typical multi-head attention block with representation dimension d and number of heads h (where h evenly divides d) has $d_k = d_v = d/h$ and is parameterized by four matrices: $W_K \in \mathbb{R}^{d \times d}$, $W_Q \in \mathbb{R}^{d \times d}$, $W_V \in \mathbb{R}^{d \times d}$ and $W_O \in \mathbb{R}^{d \times d}$. Given input $X \in \mathbb{R}^{n \times d}$, it outputs

$$\mathrm{MultiHeadAttention}(XW_Q^T, XW_K^T, XW_V^T)W_O^T$$

where here we reshape $\mathrm{MultiHeadAttention}$ to operate on matrices like $Q \in \mathbb{R}^{n \times hd_k}$ rather than on tensors.

Let's draw a block diagram of a transformer block.