ECE 046211 - Technion - Deep Learning

HW3 - Sequential Tasks and Training Methods

Part 1 - Theory

- You can choose whether to answser these straight in the notebook (Markdown + Latex) or use another editor (Word, LyX, Latex, Overleaf...) and submit an additional PDF file, **but no handwritten submissions**.
- You can attach additional figures (drawings, graphs,...) in a separate PDF file, just make sure to refer to them in your answers.
- $\LaTeX$ Cheat-Sheet (to write equations)
    - Another Cheat-Sheet

Question 1 - Transformer Encoding

In the following question we will assume we are given a Transformer capable of sentences of up to L tokens, where every token is represented by a d-dimensional vector.

1. Explain what a Positional Encoding is, why is it needed, and how it works.
2. One suggested encoding was to assign a number in range $[0, 1]$ to each word as follows: For a sentence of length $N \leq L$, add
$$\frac{t}{N-1}$$
to the t-th word. This means we add 0 to the first word and 1 to the final word. What issue can arise from this encoding? Note that the length of each sentence N can differ between sentences.
3. Another suggested encoding was to add 1 to the first word, 2 to the second and so on. Would this be a good encoding? Explain your answer.

From here on out, we will use the following encoding - let $0 \leq t < N, 0 \leq k < d$, we define

$$P_{t,k} = \begin{bmatrix} \sin(\omega_k t) \\ \cos(\omega_k t) \end{bmatrix}$$

where $\omega_k = 10000^{-2k/d}$. The encoding of word t is a d-dimensional vector of pairs $P_{t,i}$:

$$P_t = \begin{bmatrix} \sin(\omega_1 t) \\ \cos(\omega_1 t) \\ \vdots \\ \sin(\omega_{d/2} t) \\ \cos(\omega_{d/2} t) \end{bmatrix}$$

4. Explain why this gives a unique encoding for each word in the sentence regardless of its length N.
5. Show that we can linearly transform $P_{t,k}$ via offset, meaning that for any offset τ there is a matrix

$M_k^\tau \in \mathbb{R}^{2\times 2}$ such that

$$P_{t+\tau,k} = M_k^\tau P_{t,k}$$

Hint: remember that

$$\sin(\alpha+\beta) = \sin(\alpha)\cos(\beta) + \cos(\alpha)\sin(\beta)$$
$$\cos(\alpha+\beta) = \cos(\alpha)\cos(\beta) - \sin(\alpha)\sin(\beta)$$

7. Extend this to P_t, show that for $M^\tau \in \mathbb{R}^{d\times d}$

$$P_{t+\tau} = M^\tau P_t$$

Solution 1 - Transformer Encoding

1. Positional encodimgs assign a number (deterministically) to each word in a sentence corresponding to its position in the sentence. Without positional encoding, Transformers are invariant to the word location (unlike RNNs), and cannot distinguish between the locations of words.
2. The positional encoding is not a vector unlike the word embedding, but more importantly, there is no absolute meaning to the positional encoding, and the distance between positions is relative to N. This makes the network's role of understanding the meaning of the positional encoding difficult since the encoding for a word depends on sentence length.
3. This positional encoding does provide absolute values for each position, but natural numbers are unbounded, hurting the overall layer normalization, and we would have difficulties generalizing to unseen sentence lengths.
4. The encoding of the t-th word is a set of pairs, with each pair depending on the representation dimension d but not on sentence length N, and the encoding at position t will be the same no matter how long the sentence is.
5. Using the trignometric identities, it is easy to see that

$$P_{t,k} = \begin{bmatrix} \sin(\omega_k(t+\tau)) \\ \cos(\omega_k(t+\tau)) \end{bmatrix} = \begin{bmatrix} \sin(\omega_k t)\cos(\omega_k\tau) + \cos(\omega_k t)\sin(\omega_k\tau) \\ \cos(\omega_k t)\cos(\omega_k\tau) - \sin(\omega_k t)\sin(\omega_k\tau) \end{bmatrix} = \begin{bmatrix} \cos(\omega_k\tau) & \sin(\omega_k\tau \\ -\sin(\omega_k\tau) & \cos(\omega_k\tau \end{matrix}$$

And it is easy to see that

$$M_k^\tau = \begin{bmatrix} \cos(\omega_k\tau) & \sin(\omega_k\tau) \\ -\sin(\omega_k\tau) & \cos(\omega_k\tau) \end{bmatrix}$$

is independent of t. 6. Since M_k^τ is indepentent of t, we can just use a block diagonal matrix of M_k^τ:

$$M_k^\tau = \begin{bmatrix} M_1^\tau & 0 & \dots & 0 \\ 0 & M_2^\tau & \dots & 0 \\ \dots & \dots & \ddots & 0 \\ 0 & 0 & \dots & M_{d/2}^\tau \end{bmatrix}$$

Question 2 - Preventing Variance Explosion

This question relates to lectures 8-9 (from slide 7):

Find an initializtion scheme such that

$$\forall l, i, : (1)\ \mathbb{E}\left[F_l(u_l)|u_l\right] = 0,\ (2)\ Var(u_l[i]) = 1,$$

assuming skip connections: $u_{l+1} = u_l + F_l(u_l)$ with a single skip $F_l(u_l) = W_l \phi(u_l) + b_l$ and the activation is ReLU: $\phi(x) = \mathrm{ReLU}(x) = \max(0, x)$.

Solution 2 - Preventing Variance Explosion

We can use zero-initilization with some added noise to the dynamics (not to the initialization, e.g., Dropout during forward pass) to break symmetry such that $\mathbb{E}[W_l], \mathbb{E}[b_l] = 0$.

Question 3 - Recurrent Neural Networks

You are given a recurrent/feedback neural network with LReLU activations $\phi(u) = \max[pu, u]$, with input x_t and a representation $v_t \in \mathbb{R}^d$ that is updated as follows:

$$\forall \tau = 1, 2, \dots t\ : v_\tau = \phi(u_\tau)\,, u_\tau = W v_{\tau-1} + B x_\tau,$$

from initialization v_0, and outputs $\hat{y}_t = C v_t$. The network is trained with GD on a single long series $\{x_\tau, y_\tau\}_{\tau=1}^t$ with a cost function $\ell(y_t, \hat{y}_t)$ over the last term in the series.

1. Calculate the exact gradient $\frac{\partial \ell}{\partial W[i,j]}$ using Backpropagation through time (BPTT).
2. Recall that calculating the gradient using the method in the previous section there are two issues for $t \to \infty$: (1) the required computational resources grow indefinitely, and (2) the gradients explode or vanish. For each problem: explain it, provide an example for a method to alleviate it and describe any limitations of this method.

Solution 3 - Recurrent Neural Networks

1. First, let's calculate v_τ and u_τ in forward pass:

$$\forall \tau = 1, 2, \dots t\ : v_\tau = \phi(u_\tau)\,, u_\tau = W v_{\tau-1} + B x_\tau,$$

Then we perform a backward pass. Since the matrices are: Forward pass equation:

$$\forall\tau = 1, 2, \ldots t : v_\tau = \phi(u_\tau), u_\tau = Wv_{\tau-1} + Bx_\tau$$

Backward pass equations:

$$\frac{\partial}{\partial W[i,j]}\ell(y_t, \hat{y}_t) = \frac{\partial \ell(y_t, \hat{y}_t)}{\partial \hat{y}_t}\frac{\partial \hat{y}_t}{\partial v_t}\sum_{t'=1}^{t}\left(\prod_{\tau=t'+1}^{t}\frac{\partial v_\tau}{\partial v_{\tau-1}}\right)\frac{\partial v_{t'}}{\partial u_{t'}[i]}v_{t'-1}[j]$$

$$= \left[\frac{\partial \ell(y_t, \hat{y}_t)}{\partial \hat{y}_t}C\sum_{t'=1}^{t}\left(\prod_{\tau=t'+1}^{t}\mathrm{Diag}(\phi'(u_\tau))W\right)\mathrm{Diag}(\phi'(u_{t'}))\right][i]v_{t'-1}[j]$$

The backward pass is performed as follows:

$$\bar{v}_t = C^T\frac{\partial \ell(y_t, \hat{y}_t)}{\partial \hat{y}_t}$$

$$\forall\tau = 1, 2, \ldots t : \bar{v}_{\tau-1} = W^T\mathrm{Diag}(\phi'(u_\tau))\bar{v}_\tau$$

and the update:

$$\frac{\partial}{\partial W}\ell(y_t, \hat{y}_t) = \sum_{t'=1}^{t}\mathrm{Diag}(\phi'(u_{t'}))\bar{v}_{t'}v_{t'-1}^T$$

2. The first problem is that in the forward pass, we need to store (or recalculate) the values of v_τ, u_τ and $\bar{v}_\tau$ $\forall\tau = 1, 2, \ldots t$, which can be problematic when t tends to infinity, as this requires infinite resources. A good way to solve this problem is to perform the backward pass only for a fixed number of steps back. This creates a slight inaccuracy in the gradient calculation that can affect performance in certain tasks. The second problem is that the Jacobian:

$$\prod_{\tau=t'+1}^{t}\frac{\partial v_\tau}{\partial v_{\tau-1}} = \prod_{\tau=t'+1}^{t}\mathrm{Diag}(\phi'(u_\tau))W$$

can explode or vanish as t tends to infinity. This can affect the final gradient calculation $\frac{\partial}{\partial W}\ell(y_t, \hat{y}_t)$, because of numerical issues (it's difficult to perform precise operations on very large or very small numbers).

Credits

- Icons made by Becris from www.flaticon.com
- Icons from Icons8.com - https://icons8.com
- Datasets from Kaggle - https://www.kaggle.com/