EECS 182 Deep Neural Networks
Fall 2023 Anant Sahai

Discussion 8

1. Inductive Biases in Standard Deep Learning Building Blocks

In this question, we will review inductive biases that standard deep learning building blocks have. This question is inspired by [1], so we highly recommend you read through this paper.

Inductive biases are any assumptions that learners utilize to learn the world and predict the output. These assumptions can either be related to the date-generation process, the functional family space chosen, or the space of the solutions. Inductive biases could be a regularization term added for better generalization or preventing overfitting, and they could also be embedded in the model function family or model architecture (e.g. CNN vs MLP). Inductive biases generally reduce the amount of data needed to fit the model while constraining the model's flexibility. Ideally, inductive biases will improve training efficiency (small data points, small gradient steps, faster optimization), while maintaining performance and generalizing well. However, if inductive biases are mismatched with the problem domain, this will lead our solutions to be sub-optimal. Another way to think about this is through the lens of the bias-variance tradeoff: inductive biases induce a large bias term in approximation error.

The first example of inductive bias we'll look at is Ridge Regression. **What inductive biases lead us to introduce a L2 Penalty on the ordinary least squares objective?**

Now let's discuss the inductive biases of deep learning building blocks. **Fill in the table below.**

Blocks	Form of Invariance	Inductive Bias
Fully Connected	No invariance due to fully-connected nature	
Convolutional		
Recurrent		

2. Backprop through a Simple RNN

[1] Battaglia, Peter W., et al. "Relational inductive biases, deep learning, and graph networks." (2018)

Consider the following 1D RNN with no nonlinearities, a 1D hidden state, and 1D inputs u_t at each timestep. (Note: There is only a single parameter w, no bias). This RNN expresses unrolling the following recurrence relation, with hidden state h_t at unrolling step t given by:

$$h_t = w \cdot (u_t + h_{t-1}) \tag{1}$$

The computational graph of unrolling the RNN for three timesteps is shown below:

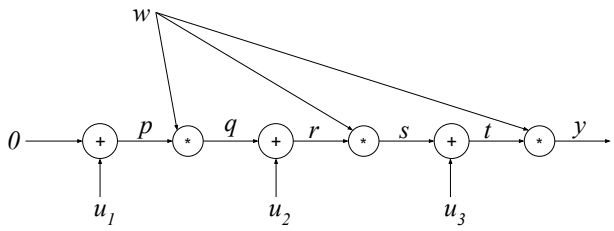


Figure 1: Illustrating the weight-sharing and intermediate results in the RNN.

where w is the learnable weight, u_1, u_2, and u_3 are sequential inputs, and p, q, r, s, and t are intermediate values.

(a) **Fill in the blanks for the intermediate values during the forward pass, in terms of w and the u_i's:**

$$p = u_1 \qquad q = w \cdot u_1 \qquad r = u_2 + q = u_2 + w \cdot u_1$$

$$s = w \cdot r = w \cdot u_2 + w^2 \cdot u_1$$

$$t = \_\_\_\_\_$$

$$y = \_\_\_\_\_$$

(b) **Using the expression for y from the previous subpart, compute $\frac{dy}{dw}$.**

(c) **Fill in the blank for the missing partial derivative of y with respect to the nodes on the backward pass.** You may use values for p, q, r, s, t, y computed in the forward pass and downstream derivatives already computed.

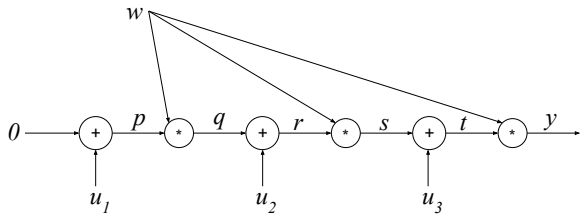
$$\frac{\partial y}{\partial t} = w \qquad \frac{\partial y}{\partial s} = w \qquad \frac{\partial y}{\partial r} = \frac{\partial y}{\partial s} \cdot w \qquad \frac{\partial y}{\partial q} = \frac{\partial y}{\partial r} \cdot 1$$

$$\frac{\partial y}{\partial p} = \_\_\_\_\_$$

(d) **Calculate the partial derivatives along each of the three outgoing edges from the learnable w in Figure 1, replicated below.** (e.g., the right-most edge has a relevant partial derivative of t in terms of how much the output y is effected by a small change in w as it influences y through this edge. You need to compute the partial derivatives for the other two edges yourself.)

You can write your answers in terms of the p, q, r, s, t and the partial derivatives of y with respect to them.

Use these three terms to find the total derivative $\frac{dy}{dw}$.



(HINT: You can use your answer to part (b) to check your work.)

Contributors:

- Anant Sahai.
- Kumar Krishna Agrawal.
- Matthew Lacayo.
- Suhong Moon.
- Shivam Singhal.
- Bryan Wu.
- Saagar Sanghavi.

