Swetha Karthikeyan

Data C182

1. Implementing RNNs (and optionally, LSTMs)
2. Expected: 0.1973753273487091, got: 0.1973753273487091, max error: 0.0

Max error all\_h: 5.0008296966552734e-05

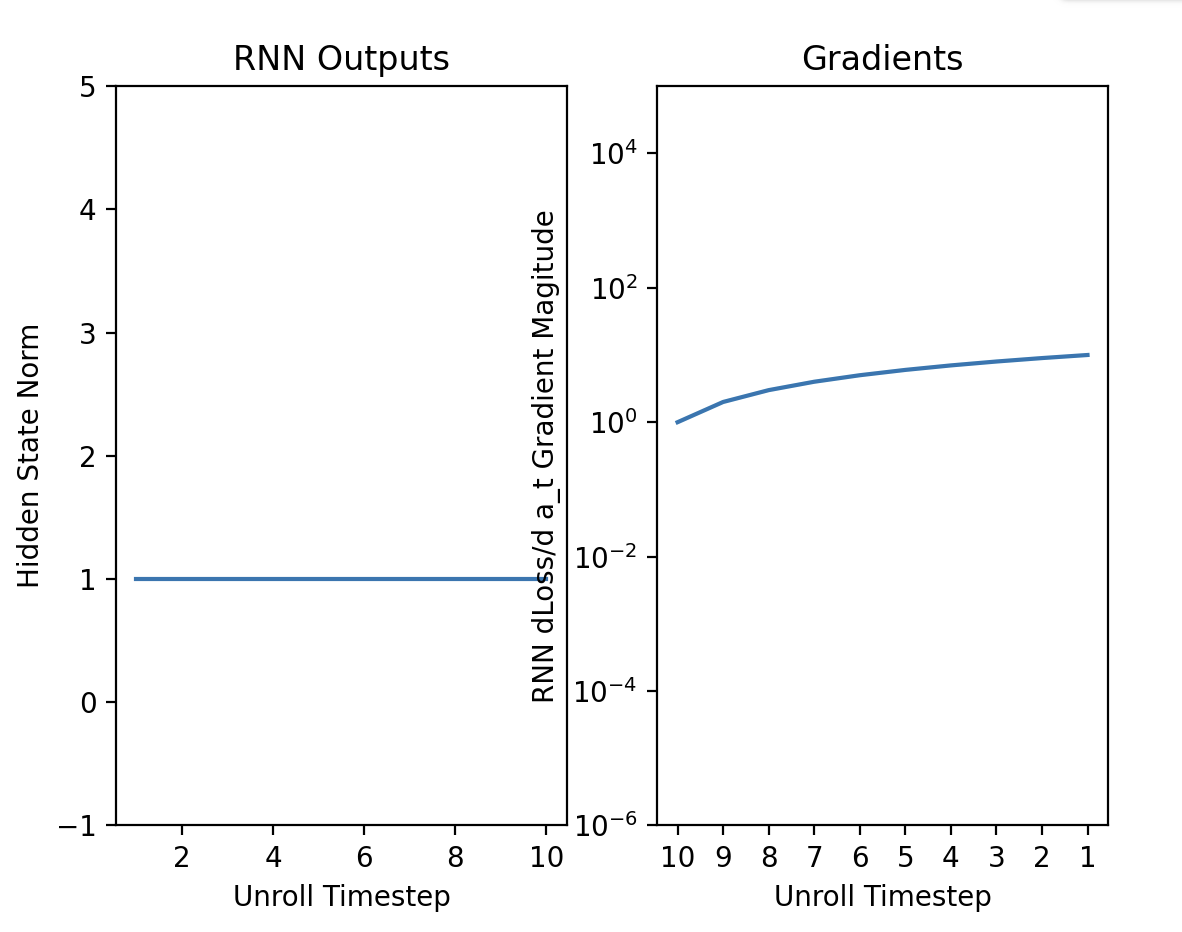
Max error last\_h: 2.498924732208252e-05

1. Max error all\_h: 4.699826240539551e-05

Max error last\_h: 4.312396049499512e-05

1. Max error loss\_all: 3.314018249511719e-05

Max error loss\_last: 2.384185791015625e-07

1. 
2. If the network has no nonlinearities, under what conditions would you expect the exploding or vanishing gradients with for long sequences? Why? (Hint: it might be helpful to write out the formula for ∂L/∂ht and analyze how this changes with different t). Do you see this pattern empirically using the visualization tool in Section 1.D in the notebook with last\_step\_only=True?

I would expect the exploding gradient if Wh is larger than 1, and the vanishing gradient if Wh is less than 1. Using the visualization tool, I can see exploding gradients when I increase the weight\_val, and I can see vanishing gradients when I decrease weight\_val.

∂L/∂ht = ∂L/∂hT \* ∏(t to T-1) (Wh^T)

1. Compare the magnitude of hidden states and gradients when using ReLU and tanh nonlinearities in Section 1.D in the notebook. Which activation results in more vanishing and exploding gradients? Why? (This does not have to be a rigorous mathematical explanation.)

When using ReLU, there tends to be more exploding gradients, and when using tanh, there tends to be a greater risk of vanishing gradients. This is because ReLu has a constant gradient of 1, while tanh generally has smaller gradients.

1. What happens if you set last\_target\_only = False in Section 1.D in the notebook? Explain why this change affects vanishing gradients. Does it help the network’s ability to learn dependencies across long sequences? (The explanation can be intuitive, not mathematically rigorous.)

When last\_target\_only = False, we find the loss function during all steps. This gives the network more gradient signals to update weights, and thus decreases the vanishing gradient. It helps the ability to learn dependencies across long sequences because the network doesn’t have to only rely on the long sequences to get update information.

1. RNNs for Last Name Classification
2. Although the neural network you have trained is intended to predict the language of origin for a given last name, it could potentially be misused. In what ways do you think this could be problematic in real-world applications?

It could be misused in job or college applications that aren’t supposed to discriminate based on race, gender, etc. The potential employer or college could use the neural network to figure out the language origin of an applicant and let their background influence their decision to accept them or not.