1. Neural networks for simple functions
2. One possible model is that . It means that will be . After passing ReLU, x remains unchanged if x>=0 and 0 remains 0. will scales to x after multiplying .
3. One possible model is that and . It means that will be . After passing ReLU, mx remains unchanged if mx>=0 and same for b. Next, . It means that the output of the network is the sum of the activations from the first and second neurons, which becomes mx+b.
4. and will be one possible model. sets for minimizing its influence on the variance of the sigmoid's output across different x values. And aligns with aiming for the sigmoid's output to be directly b.
5. . The slopes becomes 3 and -3 for the first 2 segments, which are 3x and -3x for -2<=x<=0 and 0<=x<=2. aligns with the 3x+6 and -3x+6. accordingly, which combine outputs from the first two neurons.
6. Since ReLU can only produce piecewise linear outputs, and is continuously non-linear, it's not possible to perfectly model using a finite combination of ReLU functions in such a single-layer network.
7. Yes. We can use approximation theorem to compute the w0, w1, and b0 to a high accuracy according to the procedure below:
8. We can choose a compact subset interval to approximate .
9. We can divide the interval to smaller sections, since we want to mimic use as much segments as possible.
10. Allocate a neuron for each segment to construct a linear piece that approximates that segment, which each neurons can be active or inactive based on x. We can adjust w0 to define the slope of each segment's approximation and adjust biases b0 to control when each neuron activates.
11. The output layer w1 combines the contributions of each neuron to form the overall approximation. We can modify w1 such that the linear approximations from each neuron sum up to closely follow the curve within the chosen range.
12. We need to use gradient descent to iteratively adjust w0, w1, and b0 to minimize the difference over the subset we chosed.
13. Yes. The procedures are basically the same as last problem:
14. We can divide [0,1] into smaller segments and set a neuron for each segments with w0 and b0.
15. After passing ReLU function, each segment will contribute to approximate .
16. Use gradient descent to minimize the errors generated by the model by adjusting parameters w0, w1, b0.
17. Backpropagation through time (BPTT)
18. A diagram of a graph

    Description automatically generated

.

1. For t=T, . And we need to apply this recursively to find . Thus, , which reflects the chain of dependencies through the sequence of backpropagation.
2. Exploding gradient problem occurs because of the huge magnitude of gradients. As shown of the equation in the last problem, if the weight parameter w is greater than 1, the multiple iterations will cause the terms grows exponentially. It will finally lead to unstable training dynamics where the model parameters may diverge and render the training process ineffective. Conversely, vanishing gradients occur when the absolute values of gradients become excessively small, approaching zero. From the equation, it happens because w is less than 1 and after multiple iterations, the terms will be approaching to zero. Therefore, the model may fail to capture important information from the earlier parts of the input sequence, limiting its ability to make accurate predictions based on the full context of the data.
3. Transformers
4. First, it cannot be an infinitely large value since it’s a softmax value which means it’s between 0 and 1. Second, it cannot be 0 because any input to exponential function is positive, but it can be very close to 0.
5. q and ki are in the same direction or close to it in the vector space, and the magnitude of q and ki is such that when they are normalized or when their dot product is computed, is maximized relative to other . Also, q should be less aligned with all other key vectors kj for leading to lower dot product values and, consequently, leading to smaller softmax scores.

Consequently, since the attention weights sum up to 1, if one weight is close to 1, the others must be close to 0. Hence, since .

This scenario represents a case of attention that the model focuses almost entirely on a specific piece of information or text in the sequence. When considering NLP, the word in the sentence might be crucial for understanding the sentence.

1. In order to find a q that make the contribution of v1 and v2 equally to the output c, we can render q be a vector that equally aligned with k1 and k2 and orthogonal to all other ks. Thus, . Thus, and are maximized and roughly equal because of q and after softmax’s normalization, for . Thus, it renders .
2. We can expand the RBF kernel:

.

Since , .

When , we can simplify the kernel: , which is same as the dot product softmax similarity since can act as a scaling factor.

1. First, computing involves a multiplication of an n\*d matrix with a d\*n matrix, resulting in an n\*n matrix. Since for each of the n\*n output elements, we perform d multiplications, it requires time complexity.

Then, as we apply the softmax to each row of , requires time complexity.

Then, the n\*n matrix is multiplied by an n\*d matrix, which also requires time complexity that’s similar to the first step.

Next, we apply the SVD properties:

Since W is n\*k and is d\*n, we can compute first and multiply by W.

Next, multiply the diagonal matrix with the matrix of k\*d from the last step requires time complexity.

Finally, the matrix U of n\*k multiplies with the matrix k\*d of the last step. This requires time complexity.

1. Resnet

2. The purpose of the bias term in the convolutional layer is for shifting the output. Thus, after normalization by BatchNorm2d, the effect of this shift is nearly useless because the batch normalization process centers the data around zero. Also, the shift factor in the BatchNorm2d can provide the necessary bias for the output, which is the same effect of what the bias in the convolution layer can do.

1. Image overfitting

6. Laplacian formula provides that for a two-dimensional function . The ReLU function gives max(0, x) which is a piecewise linear function. Therefore, when x>0, the second derivative is 0 for ReLU and thus the Laplacian will be 0 which leads to large flat areas in the Laplacian where the network's output changes linearly with respect to the input. As for x<0, the ReLU function outputs 0 for second derivative and thus for Laplacian output, which leads to a relative more flat area in the plot. For x=0, it’s not differentiable thus for Laplacian result it’s sort of like abrupt changes in the plot.

7. Learning rate scheduling might be a good method to improve the model. The method is basically reducing the learning rate periodically according to a pre-schedule or a pre-defined formula.

8. Without manually setting the initial weight, the initial parameters may be far away from the optimal solution, which leads to a very slow convergence and testing period. Second, if the initial random weight is too small or too large, there might be problems of vanishing or exploding gradients. On the other hand, without normalizing the input coordinates will also cause problems. First, the gradient descent might not be useful to find the optimal path for the final solution since the whole landscape is skewed by the input. Also, gradient descent might goes in other wrong directions which gets far away from the optimal solution.