CS446 hw3

Junsheng Huang

March 2024

1 Neural networks for simple functions

1.1

consider $w_0 = [1, -1]^T$ and $w_1 = [1, -1]^T$, we have $w_0 x = [x, -x]$, for $x > 0$, $w_1^T \sigma(w_0 x) = [1, -1] \cdot [x, 0]^T = x$, for $x < 0$, $w_1^T \sigma(w_0 x) = [1, -1] \cdot [0, -x]^T = x$, satisfy the requirement.

1.2

consider $w_0 = [m, -m]^T$, $b_0 = [b, -b]^T$ and $w_1 = [1, -1]^T$, we have $w_0 x = [mx + b, -mx - b]$, for $mx + b > 0$, $w_1^T \sigma(w_0 x) = [1, -1] \cdot [mx + b, 0]^T = mx + b$, for $mx + b < 0$, $w_1^T \sigma(w_0 x) = [1, -1] \cdot [0, -mx - b]^T = mx + b$, satisfy the requirement.

1.3

consider $w_0 = 0$, $w_1 = 2b$, thus we have $w_1^T \sigma(w_0 x) = w_1^T \sigma([0, 0]^T) = [b\frac{1}{2} = b$.

1.4

consider $f(x) = f_1(x) + f_2(x) + f_3(x)$, $f_1(x) = 3\sigma(x + 2)$, $f_2(x) = -6\sigma(x)$, $f_3(x) = 3\sigma(x - 2)$. Since that, we have:$w_0 = [1, 1, 1]^T$, $b = [2, -2, 0]^T$, $w_1 = [3, -6, 3]^T$.

1.5

Not exist. Using prove by contradiction to show: suppose $w_1 = [a_1, ...a_d]^T$, $w_0 = [b_1, ...b_d]^T$ satisfy the requirement. When $x = 2$, $f(x) = 4 = \sum_{i=1}^{d} a_i \sigma(2b_i)$. When $x = 4$,$f(x) = 16$, but $f(x) = \sum_{i=1}^{d} a_i \sigma(4b_i) = 8$, contradiction!

1.6

Not exist. The dimension of parameter is finite, so we can't approximate the nonlinear function for domain $\{x \in R\}$. In this case, we need infinite parameter to express our target function, which is contradicted with the finite restriction.

1.7

Exist. This time, the function has domain $\{x \in [0,1]\}$. From Universal approximate theorem, we can approximate our target function with finite parameters. The procedure is to divide f(x): $f(x) = f_1(x) + ... + f_n(x)$, each $f_i(x)$ is linear function and $\sum_{i=1}^{k} f_i(x)$ only need the k-dimensions in parameter w_0, w_1, b to approximate the domain $[\frac{k-1}{n}, \frac{k}{n}]$ with error within ϵ.(Which is similar to 1.4, using piece-wise continuous function to approximate)

2 Backpropagation through time(BPTT)

2.1

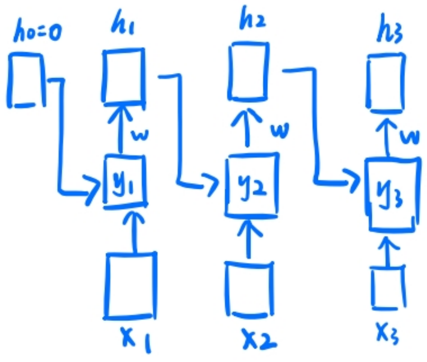

Figure 1: computation graph

2.2

$h_0 = 0$
$y_1 = x_1 + h_0 = x_1$
$h_1 = w(x_1 + h_0) = wx_1$
$y_2 = x_2 + h_1 = x_2 + wx_1$
$h_2 = wy_2 = wx_2 + w^2x_1$
$y_3 = x_3 + h_2 = x_3 + wx_2 + w^2x_1$
$h_3 = wy_3 = wx_3 + w^2x_2 + w^3x_1$

2.3

$$\begin{aligned}\frac{\partial h_3}{\partial w} &= \frac{\partial h_3}{\partial w} + \frac{\partial h_3}{\partial h_2}\frac{\partial h_2}{\partial w} + \frac{\partial h_3}{\partial h_2}\frac{\partial h_2}{\partial h_1}\frac{\partial h_1}{\partial w} \\ &= y_3 + wy_2 + w^2y_1 \\ &= x_3 + 2wx_2 + 3w^2x_1 \qquad (1)\end{aligned}$$

2.4

$$\begin{aligned}\frac{\partial f}{\partial h_1} &= \frac{\partial h_T}{\partial h_1} \\ &= \frac{\partial h_T}{\partial h_{T-1}}\frac{\partial h_{T-1}}{\partial h_{T-2}}\cdots\frac{\partial h_2}{\partial h_1} \\ &= w^{T-1}\prod_{i=2}^{T}\sigma'(x_i + h_{i-1}) \qquad (2)\end{aligned}$$

2.5

consider $w > 1$, after long-term $(T \to \infty)$,$w^{T-1} \to \infty$,thus exploding gradient. consider $w < 1$, after long-term $(T \to \infty)$,$w^{T-1} \to 0$,thus vanishing gradient.

3 Transformers

3.1

$$\alpha_i = \frac{exp(k_i^T q)}{\sum_j exp(k_j^T q)}$$

you can see numerator is contained in denominator and $exp(k_j^T q)$ always > 0. Thus, for one certain $k_i^T q \to \infty$, $\alpha_i \to 1$, can't be infinitely large. For one certain $k_i^T q \to \infty$, other$j \neq i$ has $\alpha_j \to 0$, so it could be zero.

3.2

In this case, q is highly similar to k_i and not related with other key k_j. $c = \sum_m \alpha_m v_m = v_i$ in this case.
This represent that when q "query" the "match key" in the sequence, k_i is the most "match key" while other is not matched. So the model will pay all attention to the value v_i.

3.3

$$q = 100d \cdot k_1 + 100d \cdot k_2$$

In this case, for $i = 1, 2$: $k_i^T q = 100d \cdot k_i^T k_i = 100d$, for $i = 3, 4...d$: $k_i^T q = 0$. So, for $i = 1, 2$: $\alpha_i = \frac{exp(100d)}{2exp(100d)+d-2} \approx \frac{1}{2}$, for $i = 3, 4...d$: $\alpha_i = \frac{1}{2exp(100d)+d-2} \approx 0$. Thus, $c \approx \frac{1}{2}(v_1 + v_2)$

3.4

$$y_i = exp(q^T k_i)$$

$$\begin{aligned} z_i &= exp(\frac{-\|q-k_i\|_2^2}{2\sigma}) \\ &= exp(-\frac{\|q\|_2^2 - 2q^T k_i + \|k_i\|_2^2}{2\sigma}) \\ &= exp(-\frac{\|q\|_2^2 + 1}{2\sigma}) y_i \end{aligned} \tag{3}$$

$$\begin{aligned} \alpha_i &= \frac{exp(k_i^T q)}{\sum_j exp(k_j^T q)} \\ &= \frac{y_i}{\sum_j y_j} \\ &= \frac{z_i}{\sum_j z_j} \end{aligned} \tag{4}$$

So we can rewrite the dot product softmax similarity with the RBF kernel.

3.5

$$P = softmax(QK^T) = A\Sigma B^T$$

P has rank k means Σ has only k elements on the diagonal and we can rewrite PV as: (subscript means the shape of the matrix)

$$PV = A_{nxm}\Sigma_{nxn}B_{nxn}V_{nxd} = A_{nxk}\Sigma_{kxk}B_{kxn}V_{nxd}$$

Computing BV takes $O(nkd)$ time.

$$PV = A_{nxk}\Sigma_{kxk}C_{kxd} = A_{nxk}D_{kxd}$$

Computing ΣC and AD takes $O(nkd)$ time for each.
In total, attention can be computed in $O(nkd)$ time.

4 Resnet

BatchNorm2d layer will centring (minus the median) and zoom (divide the standard deviation), so adding bias in **Conv2d** layer is meaningless.

5 Image overfitting

5.1 ReLU

epochs: 300 learning rate: 0.001 batch size: 64

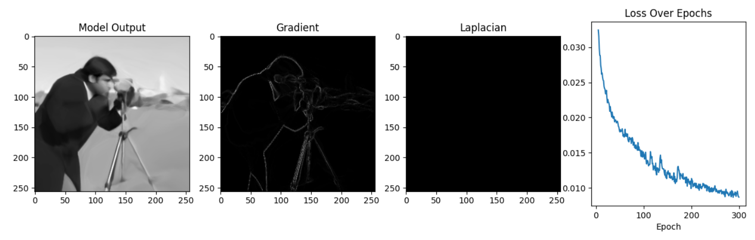

Figure 2: ReLU

5.2 Tanh

epochs: 300 learning rate: 0.001 batch size: 100

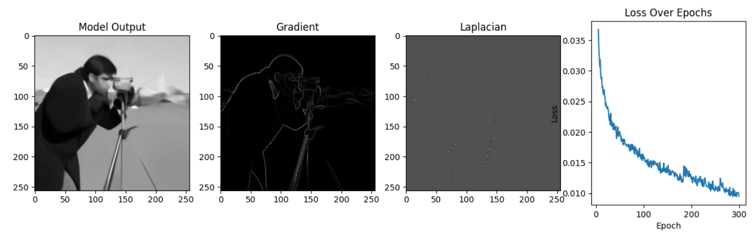

Figure 3: Tanh

5.3 Sin

epochs: 200 learning rate: 0.0001 batch size: 2048

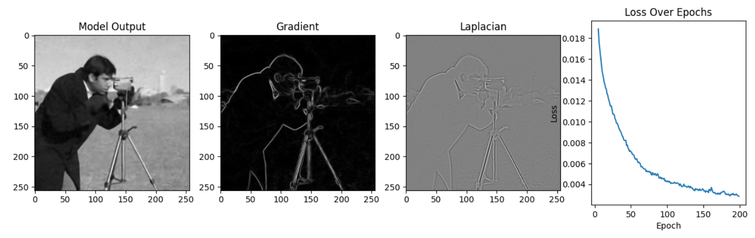

Figure 4: Sin

5.4 Q6

we have:

$$\frac{d^2}{dz^2}RELU(z) = \begin{cases} 0 & z \neq 0 \\ undefined & z = 0 \end{cases}$$

So when we are using ReLU as the activation, the graph of the Laplacian is all black. The Laplacian of the output with respect to the input is zero except for input is 0 (which is non-differentiable).

5.5 Q7

We should reduce the learning rate and probably add more epochs. This strategy is called learning rate decay.

5.6 Q8

Do not manually set the initial weights for sin: Every weight should be default initialization in this case, the loss stuck at 0.32 and cannot go down anymore. Thus the output is totally noise.

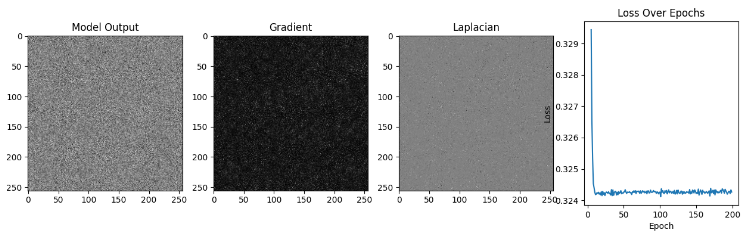

Figure 5: No Init