HW2

Faezeh Pouya Mehr

April 12, 2023

1 Question1

1.1 1.1

According to the mathematical induction we assume this equation holds for time step t-1 :

$$\sigma(h_{t-1}) = g_{t-1}$$

and now we want to prove this equation also holds for time step t:

$$\sigma(h_t) = g_t$$

according to the definition of h_t we have:

$$h_t = W\sigma(h_{t-1}) + Ux_t + b$$

now we apply σ on both side of the above equation:

$$\sigma(h_t) = \sigma(W\sigma(h_{t-1}) + Ux_t + b)$$

based on our induction assumption we know $\sigma(h_{t-1}) = g_{t-1}$ so we have :

$$\sigma(h_t) = \sigma(Wg_{t-1} + Ux_t + b) = g_t$$

so we prove that:

$$\sigma(h_t) = g_t$$

1.2 1.2

to show that the gradients of the hidden state will vanish over time if $\lambda_1(W^TW) \le \delta^2/\gamma^2$, we need to show that $|\partial g_T/\partial g_0| \to 0$ as $T \to \infty$.

We can use the chain rule to express the gradient of g_T with respect to g_0:

$$\frac{\partial g_T}{\partial g_0} = \frac{\partial g_T}{\partial g_{T-1}} \frac{\partial g_{T-1}}{\partial g_{T-2}} \cdots \frac{\partial g_1}{\partial g_0}$$

Using the recurrence relation for g:

$$g_T = \sigma(Wg_{T-1} + Ux_T + b)$$

We can compute the gradient of g_T with respect to g_{T-1} as:

$$\frac{\partial g_T}{\partial g_{T-1}} = \sigma'(Wg_{T-1} + Ux_T + b)W$$

Using the property of the L_2 operator norm $||AB|| \le ||A||\,||B||$, we can bound the norm of $\partial g_T/\partial g_{T-1}$ as:

$$||\frac{\partial g_T}{\partial g_{T-1}}|| \le ||\sigma'(Wg_{T-1} + Ux_T + b)||\,||W||$$

Since the derivative of σ is bounded by γ, we have:

$$||\frac{\partial g_T}{\partial g_{T-1}}|| \leq \gamma ||W||$$

Using the recurrence relation for g again, we can compute the gradient of g_{T-1} with respect to g_{T-2} as:

$$\frac{\partial g_{T-1}}{\partial g_{T-2}} = \sigma'(W g_{T-2} + U x_{T-1} + b)W$$

Similarly, we can bound the norm of $\partial g_{T-1}/\partial g_{T-2}$ as:

$$||\frac{\partial g_{T-1}}{\partial g_{T-2}}|| \leq \gamma ||W||$$

Continuing this process, we can bound the norm of the gradient of g_T with respect to g_0 as:

$$||\frac{\partial g_T}{\partial g_0}|| \leq \gamma^T ||W||^T$$

Using the property of the L_2 operator norm $||A|| = \sqrt{\lambda_1(A^\top A)}$, we have:

$$||W||^2 = \lambda_1(W^\top W)$$

Since $\lambda_1(W^\top W) \leq \delta^2/\gamma^2$, we can bound the norm of the gradient of g_T with respect to g_0 as:

$$||\frac{\partial g_T}{\partial g_0}|| \leq \gamma^T (\delta^2/\gamma^2)^{T/2}$$

$$||\frac{\partial g_T}{\partial g_0}|| \leq (\gamma \sqrt{\frac{\delta^2}{\gamma^2}})^T = (\delta)^T$$

As T approaches infinity, δ^T goes to 0 since $\delta < 1$. Thus we have shown that $||\frac{\partial g_T}{\partial g_0}|| \to 0$ as $T \to \infty$. This implies that the gradients of the hidden state will vanish over time.

1.3 1.3

If the largest eigenvalue of the weights is greater than δ^2/γ^2, gradients of hidden state may explode due to large eigenvalues in the weight matrix causing exponential growth. This is not the only factor for gradient explosion, as activation function, initial hidden state magnitude, and learning rate also contribute. Similarly, other factors such as long-term dependencies in input sequence or weight matrix initialization can cause gradient vanishing, not just the condition mentioned earlier

2 Question2

2.1 2.1

Recall the update rules for SGD with momentum and SGD with running average of gradients:

SGD with momentum:

$$v_t = \alpha v_{t-1} + \epsilon g_t \text{ and } \Delta\theta_t = -v_t \text{ where } \epsilon > 0 \text{ and } \alpha \in (0, 1).$$

SGD with running average of g_t:

$$v_t = \beta v_{t-1} + (1-\beta) g_t \text{ and } \Delta\theta_t = -\delta v_t \text{ where } \beta \in (0, 1) \text{ and } \delta > 0.$$

Now we can express the two update rules recursively:

For SGD with momentum, we have:

$$\begin{aligned} v_t &= \alpha v_{t-1} + \epsilon g_t \\ \Delta\theta_t &= -v_t \\ \Delta\theta_t &= -(\alpha v_{t-1} + \epsilon g_t) \\ \Delta\theta_t &= \alpha \Delta\theta_{t-1} - \epsilon g_t \end{aligned}$$

For SGD with running average of g_t, we have:

$$\begin{aligned} v_t &= \beta v_{t-1} + (1-\beta) g_t \\ \Delta\theta_t &= -\delta v_t \\ \Delta\theta_t &= -\delta(\beta v_{t-1} + (1-\beta) g_t) \\ \Delta\theta_t &= -\delta\beta v_{t-1} - \delta(1-\beta) g_t \\ \Delta\theta_t &= \beta \Delta\theta_{t-1} - \delta(1-\beta) g_t \end{aligned}$$

Comparing the two expressions for $\Delta\theta_t$, we see that they are equivalent if we set:

$$\alpha = \beta \text{ and } \epsilon = \delta(1-\beta)$$

Therefore, the two update rules are equivalent and we have shown that estimating the first moment of the gradient using an exponential running average is equivalent to using momentum.

2.2 2.2

Recall the update rule for SGD with running average of gradients:

$$v_t = \beta v_{t-1} + (1-\beta) g_t$$

Expanding v_t recursively, we have:

$$\begin{aligned} v_t &= \beta v_{t-1} + (1-\beta) g_t \\ &= \beta(\beta v_{t-2} + (1-\beta) g_{t-1}) + (1-\beta) g_t \\ &= \beta^2 v_{t-2} + \beta(1-\beta) g_{t-1} + (1-\beta) g_t \\ &= \beta^3 v_{t-3} + \beta^2(1-\beta) g_{t-2} + \beta(1-\beta) g_{t-1} + (1-\beta) g_t \\ &= \beta^4 v_{t-4} + \beta^3(1-\beta) g_{t-3} + \beta^2(1-\beta) g_{t-2} + \beta(1-\beta) g_{t-1} + (1-\beta) g_t \\ &\vdots \\ &= \beta^{t-1} v_1 + \beta^{t-2}(1-\beta) g_2 + \beta^{t-3}(1-\beta) g_3 + \cdots + \beta(1-\beta) g_{t-1} + (1-\beta) g_t \\ &= \beta^{t-1}(\beta * v_0 + (1-\beta) g_1) + \beta^{t-2}(1-\beta) g_2 + \beta^{t-3}(1-\beta) g_3 + \cdots + \beta(1-\beta) g_{t-1} + (1-\beta) g_t \\ &= \beta^t v_0 + \beta^{t-1}(1-\beta) g_1 + \beta^{t-2}(1-\beta) g_2 + \beta^{t-3}(1-\beta) g_3 + \cdots + \beta(1-\beta) g_{t-1} + (1-\beta) g_t \\ &= \beta^t v_0 + \sum_{i=1}^{t} (\beta^{t-i}(1-\beta) g_i) \end{aligned}$$

since $v_0 = 0$ we have

$$v_t = \sum_{i=1}^{t} (\beta^{t-i}(1-\beta) g_i)$$

Therefore, we have expressed v_t as a linear combination of g_i's ($1 \le i \le t$), where the coefficients depend on β and t. This shows that the running average update rule implicitly computes a weighted sum of the past gradients, with exponentially decaying weights that depend on β. This allows the algorithm to place more emphasis on recent gradients while still considering information from past gradients.

2.3 2.3

from the previous part we found that :

$$v_t = \sum_{i=1}^{t} (\beta^{t-i}(1-\beta)g_i)$$

taking expectation from both side and using the fact that the expected value is linear we have:

$$E[v_t] = E[\sum_{i=1}^{t} (\beta^{t-i}(1-\beta)g_i)] = (1-\beta)\sum_{i=1}^{t} \beta^{t-i}E[g_i]$$

since g_t has a stationary distribution independent of t $E[g_i]$ for i=1 to t are all equall. so we can rewrite the above equation as:

$$E[v_t] = E[g_t](1-\beta)\sum_{i=1}^{t} \beta^{t-i} = E[g_t](1-\beta)\frac{1-\beta^t}{1-\beta} = E[g_t] * (1-\beta^t)$$

as it can been seen from the above equation $E[v_t]$ is a biased estimator and for having an unbiased estimator we have to define new $v'_t = \frac{v_t}{1-\beta^t}$

3 Question 3

3.1 3.1

After one-step gradient descent with the gradient g and learning rate ϵ, we update the current point x_0 to x_1 as:

$$x_1 = x_0 - \epsilon g$$

We can substitute this updated value of x_1 in the second-order Taylor expansion of f at x_0 to obtain an approximation of the function after the update. Specifically, we have:

$$\hat{f}x_0(x_1) = f(x_0) + (x_1 - x_0)^T g + \frac{1}{2}(x_1 - x_0)^T H(x_1 - x_0)$$

Substituting $x_1 = x_0 - \epsilon g$, we get:

$$\hat{f}x_0(x_1) = f(x_0) + (-\epsilon g)^T g + \frac{1}{2}(-\epsilon g)^T H(-\epsilon g)$$

Simplifying this expression, we get:

$$\hat{f}x_0(x_1) = f(x_0) - \epsilon g^T g + \frac{1}{2}\epsilon^2 g^T H g$$

Therefore, the value of $\hat{f}x_0(\cdot)$ after the update is given by the above expression.

3.2 3.2

There are three terms involved in calculating the optimal step size for gradient descent: the original value of the function, the expected improvement due to the slope of the function, and the correction we must apply to account for the curvature of the function. However, when the correction term is too large, the gradient descent step may actually move uphill. If $g^T Hg$ is zero or negative, the Taylor series approximation predicts that increasing the step size will lead to a decrease in the function value. However, for large step sizes, the Taylor series may no longer be accurate, and one must use more heuristic choices of step size. On the other hand, if $g^T Hg$ is positive, the optimal step size that decreases the Taylor series approximation of the function most can be found by solving for it:

$$\epsilon* = \frac{g^T g}{g^T Hg}$$

3.3 3.3

To derive a new optimization algorithm based on setting the gradient of $\hat{f}x_0(\cdot)$ to zero, we can first compute the gradient of $\hat{f}x_0(\cdot)$ with respect to x and set it to zero. Specifically, we have:

$$\nabla \hat{f}_{x_0}(x) = g + H(x - x_0) = 0$$

Solving for x, we get:

$$x = x_0 - H^{-1}g$$

This expression gives the optimal point x that minimizes the second-order approximation of f around x_0. Therefore, we can use this expression as the update rule for the optimization algorithm. This algorithm is known as Newton's method and involves computing the inverse of the Hessian matrix at each iteration, which can be computationally expensive for large-scale problems but speeds up the convergence of the algorithm.

4 Question 4

4.1 4.1

In the case of a single output layer conditioned on one input feature x, the equation for Weight Normalization (WN) is:

$$y = \sigma\left(W^T x + b\right)$$

Assuming that x is whitened to be independently distributed with zero mean and unit variance, the equation for Batch Normalization (BN) is:

$$y = \gamma \cdot x_{\text{norm}} + \beta$$

where x_{norm} is the normalized version of x, given by:

$$x_{\text{norm}} = \frac{x - \text{mean}(x)}{\sqrt{\text{var}(x) + \text{eps}}}$$

where mean(x) and var(x) are the mean and variance of x, respectively, and eps is a small constant for numerical stability.

To show that WN is equivalent to BN in this case, we can start by substituting the reparameterized weight vector $w = \frac{g}{||u||}u$into the equation for y:

$$y = \sigma\left(\left(\frac{g}{||u||}\right) u^\top x + b\right)$$

Next, we can rewrite $u^T * x$ in terms of the normalized version of x:

$$u^\top x = ||u|| \left(\frac{u}{||u||}\right)^\top x = ||u|| \left(\frac{u}{||u||}\right)^\top \left(\sqrt{\text{var}(x) + \text{eps}} \cdot x_{\text{norm}} + \text{mean}(x)\right)$$

substitute

$$x_{\text{norm}} = \frac{x - \text{mean}(x)}{\sqrt{\text{var}(x) + \text{eps}}}$$

and simplify

Substituting this expression back into the equation for y, we get:

$$\begin{aligned}
y &= \sigma\left(\frac{g}{||u||}\left(||u||\left(\frac{u}{||u||}\right)^T \sqrt{\text{var}(x) + \epsilon}\, x_{\text{norm}} + ||u||\left(\frac{u}{||u||}\right)^T \text{mean}(x)\right) + b\right) \\
&= \sigma\left(g\left(\frac{u}{||u||}\right)^T \sqrt{\text{var}(x) + \epsilon}\, x_{\text{norm}} + g\left(\frac{u}{||u||}\right)^T \text{mean}(x) + b\right) \\
&= \sigma\left(\gamma^{'}\ x_{\text{norm}} + \beta^{'}\right)
\end{aligned}$$

Comparing this equation with the equation for BN, we can see that they are equivalent, with the scaling and shifting factors for BN (γ and β) replaced by $g * (u/||u||) * \sqrt{\mathrm{var}(x) + \epsilon}$ and $g * (\frac{u}{||u||})^T \mathrm{mean}(x) + b$, respectively. Therefore, in this simple case, WN is equivalent to BN, ignoring the learned scale and shift terms for both BN and WN.

4.2 4.2

To derive the expression for the gradient of L with respect to u, we first express w in terms of g and u as follows:

$$w = g\frac{u}{\|u\|}$$

Now, let's compute the gradient of L with respect to u using the chain rule:

$$\nabla_u L = \frac{\partial L}{\partial w}\frac{\partial w}{\partial u}$$

Using the definition of w and the chain rule, we can write:

$$\frac{\partial w}{\partial u} = \frac{\frac{\partial gu}{\partial u}\|u\| - \frac{\partial \|u\|}{\partial u} * gu}{\|u\|^2}$$

we know that The gradient of the norm of a vector u with respect to u is given by:

$$\nabla_u ||u|| = \frac{u}{||u||}$$

so we have :

$$\frac{\partial w}{\partial u} = g\frac{1}{\|u\|}\left(I - \frac{uu^T}{\|u\|^2}\right)$$

where I is the identity matrix.
Now, let's compute $\frac{\partial L}{\partial w}$. Using the chain rule, we can write:

$$\frac{\partial L}{\partial w} = \frac{\partial L}{\partial y}\frac{\partial y}{\partial w}$$

$$\frac{\partial y}{\partial w} = \sigma'(w^T x + b)x$$

$$\frac{\partial L}{\partial w} = \frac{\partial L}{\partial y}\sigma'(w^T x + b)x$$

where $y = \sigma(w^T x + b)$ and σ' is the derivative of the activation function σ.
Therefore,

$$\nabla_u L = \frac{\partial L}{\partial y}\sigma'(w^T x + b)xg\frac{1}{\|u\|}\left(I - \frac{uu^T}{\|u\|^2}\right)$$

Now, we can simplify this expression by defining $s = \frac{\partial L}{\partial y}\sigma'(w^T x+b)g\frac{1}{\|u\|}$ and $W^* = I - \frac{uu^T}{\|u\|^2}$, which is the orthogonal complement projection matrix. Then, we get:

$$\nabla_u L = sW^* x$$

Therefore, we have shown that the gradient of L with respect to the new parameters u can be expressed in the form $sW^*\nabla_w L$, where s is a scalar and W^* is the orthogonal complement projection matrix. This means that the gradient is projected onto the subspace orthogonal to the direction of w, which helps to condition the landscape over which we want to optimize.

4.3 4.3

As weight updates progress in a neural network that uses weight normalization with standard gradient descent and without momentum, the norm of parameter u steadily increases due to projecting away from w. To update the parameter, we use steepest ascent/descent, denoted as $u' = u + \Delta u$, where Δu is proportional to the gradient of the loss function with respect to u, denoted as $\nabla_u L$. As we calculate $\nabla_u L$, we project away from the current weight vector w, making Δu perpendicular to w. Since u is proportional to w, the update is also perpendicular to v and increases its norm via the Pythagorean theorem. If $||\Delta u||/||u|| = c$, the new weight vector's norm $||u'|| = \sqrt{||u||^2 + c^2||u||^2} = \sqrt{1 + c^2}||u|| \geq ||u||$. The increase rate will depend on the weight gradient variance. High noise in the gradients results in a high c, causing the norm of v to rise quickly and decreasing the scaling factor $g/||u||$. If the gradients' norm is small, we get $\sqrt{1 + c^2} \approx 1$, stopping the norm of u from rising. This mechanism leads to the scaled gradient self-stabilizing its norm. Empirically, we find that the ability to increase the norm $||u||$ makes weight normalization optimization in neural networks highly robust to changes in learning rate. If the learning rate is too high, the norm of the unnormalized weights will rise swiftly until an appropriate effective learning rate is reached. Once the weights' norm has become large concerning the update norm, the effective learning rate stabilizes. ref

References