Data 188 Introduction to Deep Learning
Spring 2026 Eric Kim

Discussion 03

This week we will cover Optimization and Initialization. You'll practice calculating the parameter update of different optimization algorithms on a simple example, and have a closer look at the dead ReLU problem.

1. Optimization Update Rules

Consider a simple scalar-based one layer network with input $x \in \mathbb{R}$, weight $w \in \mathbb{R}$, bias $b \in \mathbb{R}$, and label $y \in \mathbb{R}$ given by the function $\hat{y} = \text{ReLU}(wx + b)$. Assume our loss function is $L(\hat{y}, y) = \frac{1}{2}\|\hat{y} - y\|_2^2$.

For reference, we provide the following derivatives that you may find useful:

$$\frac{\partial L}{\partial w} = (\hat{y} - y)\text{ReLU}'(wx + b) \cdot x$$
$$\frac{\partial L}{\partial b} = (\hat{y} - y)\text{ReLU}'(wx + b) \cdot 1$$
$$\text{ReLU}'(z) = \begin{cases} 1 & \text{if } z > 0 \\ 0 & \text{otherwise} \end{cases}$$

Update rules for different optimizers: At each update, we stochastically sample a batch of data points, compute the average loss L on that batch, and compute the gradient of the loss with respect to every parameter θ (which are w and b in our case). Also note that for Momentum and Adam, we need to keep track of the (first/second) moment u, v, which has the same shape as θ, for each parameter. The update rules for some optimizers are as follows:
SGD (Stochastic Gradient Descent):

$$\theta_{t+1} = \theta_t - \alpha \nabla_\theta L(\theta_t)$$

SGD with Momentum:

$$\begin{aligned} u_{t+1} &= \beta u_t + (1 - \beta)\nabla_\theta L(\theta_t) \\ \theta_{t+1} &= \theta_t - \alpha u_{t+1} \end{aligned}$$

Adam:

$$\begin{aligned} u_{t+1} &= \beta_1 u_t + (1 - \beta_1)\nabla_\theta L(\theta_t) \\ v_{t+1} &= \beta_2 v_t + (1 - \beta_2)(\nabla_\theta L(\theta_t))^2 \\ \theta_{t+1} &= \theta_t - \alpha \frac{u_{t+1}}{\sqrt{v_{t+1}} + \epsilon} \end{aligned}$$

CAUTION: We are using a ReLU activation function in this question just for the sake of practice. When building an actual neural network in real life, it is important to choose the right final layer activation function. For example, when outputting a probability distribution, we should use sigmoid or softmax. For regression problems, usually we do not use any activation function at the final layer. If we use ReLU as the final layer activation function, then our neural network's output is always non-negative.

We are interested in computing the parameter update for w and b using SGD, SGD with Momentum, and Adam. For simplicity, we set the batch size to be 1.

Assume we have some data points (x_i, y_i). We initialize $w = 1$ and $b = -1$. We also set the learning rate $\alpha = 0.5$, momentum parameter $\beta = 0.5$, Adam parameters $\beta_1 = 0.5$, $\beta_2 = 0.5$, and $\epsilon = 10^{-8}$. For convenience, we initialize the (first and second) moment $u_0 = 0$ and $v_0 = 0$ to be zero vectors for all parameters. Note that there are a lot of variants to Adam/Momentum with small correction terms. For simplicity and consistency with lecture, we will ignore those correction terms in this question.

(a) Why do we have an ϵ term in the Adam update rule? What would happen if we set $\epsilon = 0$?

Suppose for the first step, we sample the first data point $(x = 3, y = 0)$. Compute the parameter update for w and b, as well as relevant optimizer state update, using SGD, SGD with Momentum, and Adam. For convenience, we have provided the gradient formulae.

$$\begin{aligned}\hat{y} &= \text{ReLU}(wx + b) = \text{ReLU}(1 \cdot 3 - 1) = \text{ReLU}(2) = 2\\ \nabla_w L &= (\hat{y} - y)\text{ReLU}'(wx + b) \cdot x = (2 - 0) \cdot 1 \cdot 3 = 6\\ \nabla_b L &= (\hat{y} - y)\text{ReLU}'(wx + b) \cdot 1 = (2 - 0) \cdot 1 \cdot 1 = 2\end{aligned}$$

(b) For the first iteration, calculate the parameter updates for w, b using SGD:

(c) For the first iteration, calculate the parameter updates for w, b and moment update for u^w, u^b using SGD with Momentum, where u^w and u^b are the momentum terms for w and b respectively:

(d) For the first iteration, calculate the parameter updates for w, b, first moment update for u^w, u^b, and second moment update v^w, v^b using Adam, where u^w, u^b are the first moment terms and v^w, v^b are the second moment terms for w and b respectively:

Building on the first step update, suppose for the second step we sample the second data point $(x = -3, y = 0)$. Compute the parameter update for w and b, as well as relevant optimizer state update, using SGD, SGD with Momentum, and Adam.

(e) For the second iteration, calculate the parameter updates for w, b using SGD. For convenience, we have provided the parameters after iteration 1 and the new gradients:

$$\begin{aligned}
w &= -2\\
b &= -2\\
\hat{y} &= \text{ReLU}(wx+b) = \text{ReLU}(-2\cdot -3 - 2) = \text{ReLU}(4) = 4\\
\nabla_w L &= (\hat{y} - y)\text{ReLU}'(wx+b)\cdot x = (4-0)\cdot 1 \cdot -3 = -12\\
\nabla_b L &= (\hat{y} - y)\text{ReLU}'(wx+b)\cdot 1 = (4-0)\cdot 1 \cdot 1 = 4
\end{aligned}$$

(f) For the second iteration, calculate the parameter updates for w, b and moment update for u^w, u^b using SGD with Momentum, where u^w and u^b are the momentum terms for w and b respectively. For convenience, we have provided the parameters and moments after iteration 1 as well as the new gradients:

$$\begin{aligned}
u^w &= 3\\
u^b &= 1\\
w &= -0.5\\
b &= -1.5\\
\hat{y} &= \text{ReLU}(wx+b) = \text{ReLU}(-0.5\cdot -3 - 1.5) = \text{ReLU}(0) = 0\\
\nabla_w L &= (\hat{y} - y)\text{ReLU}'(wx+b)\cdot x = (0-0)\cdot 0 \cdot -3 = 0\\
\nabla_b L &= (\hat{y} - y)\text{ReLU}'(wx+b)\cdot 1 = (0-0)\cdot 0 \cdot 1 = 0
\end{aligned}$$

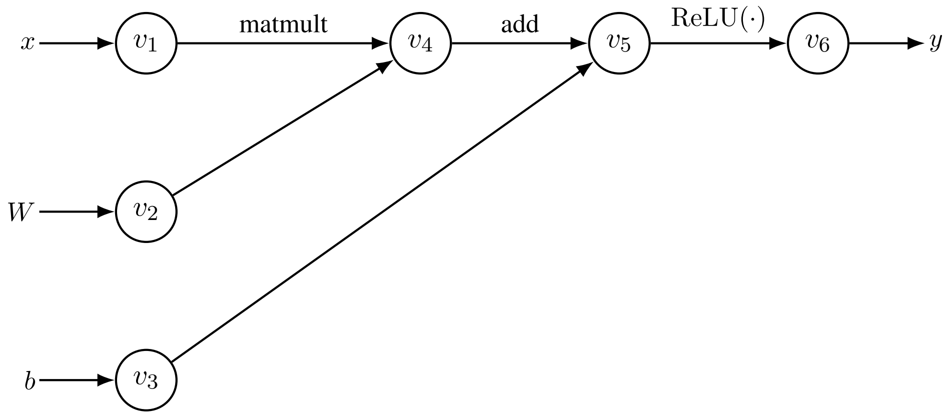
(g) For the second iteration, calculate the parameter updates for w, b and moment update for u^w, u^b, v^w, v^b using Adam, where u^w, u^b are the first moment terms and v^w, v^b are the second moment terms for w and b respectively. For convenience, we have provided the parameters, first moments, and second moments after iteration 1 as well as the new gradients.

$$\begin{aligned}
u^w &= 3 \\
u^b &= 1 \\
v^w &= 18 \\
v^b &= 2 \\
w &= 0.65 \\
b &= -1.35 \\
\hat{y} &= \text{ReLU}(wx+b) = \text{ReLU}(0.65 \cdot -3 - 1.35) = \text{ReLU}(-3.3) = 0 \\
\nabla_w L &= (\hat{y} - y)\text{ReLU}'(wx+b) \cdot x = (0-0) \cdot 0 \cdot -3 = 0 \\
\nabla_b L &= (\hat{y} - y)\text{ReLU}'(wx+b) \cdot 1 = (0-0) \cdot 0 \cdot 1 = 0
\end{aligned}$$

2. Dead ReLU at Bad Initialization

Consider one linear layer with ReLU activation function in a larger neural network. Assume the input to this layer is $x \in \mathbb{R}^{1\times n}$, the weight matrix is $W \in \mathbb{R}^{n\times m}$, the bias is $b \in \mathbb{R}^{1\times m}$, and the output of this layer is given by the function $y = \text{ReLU}(xW + b) \in \mathbb{R}^{1\times m}$. In addition, the layer's output y can also be fed into other components of the neural network, and thus the loss L depends on y.

For convenience, the computation graph of this layer is shown below.



The node definitions are as follows: $v_1 = x, v_2 = W, v_3 = b, v_4 = xW, v_5 = xW + b, v_6 = y = \text{ReLU}(xW+b)$. For convenience, the adjoints of all the nodes are provided as follows (though we encourage you to derive them yourself!):

$$
\begin{aligned}
\bar{v}_6 &= \frac{\partial L}{\partial v_6} = \frac{\partial L}{\partial y} \in \mathbb{R}^{1\times m} \\
\bar{v}_5 &= \frac{\partial L}{\partial v_5} = \frac{\partial L}{\partial v_6} \cdot \frac{\partial v_6}{\partial v_5} = \bar{v}_6 \cdot \frac{\partial}{\partial v_5}\text{ReLU}(v_5) \in \mathbb{R}^{1\times m} \\
\bar{v}_4 &= \frac{\partial L}{\partial v_4} = \frac{\partial L}{\partial v_5} \cdot \frac{\partial v_5}{\partial v_4} = \bar{v}_5 \cdot 1 = \bar{v}_5 \in \mathbb{R}^{1\times m} \\
\bar{v}_3 &= \frac{\partial L}{\partial v_3} = \frac{\partial L}{\partial v_5} \cdot \frac{\partial v_5}{\partial v_3} = \bar{v}_5 \cdot 1 = \bar{v}_5 \in \mathbb{R}^{1\times m} \\
\bar{v}_2 &= \frac{\partial L}{\partial v_2} = \frac{\partial L}{\partial v_4} \cdot \frac{\partial v_4}{\partial v_2} = x^\top \bar{v}_4 \in \mathbb{R}^{n\times m} \\
\bar{v}_1 &= \frac{\partial L}{\partial v_1} = \frac{\partial L}{\partial v_4} \cdot \frac{\partial v_4}{\partial v_1} = \bar{v}_4 \cdot W^\top \in \mathbb{R}^{1\times n}
\end{aligned}
$$

Where we have already reshaped the adjoint $\frac{\partial L}{\partial v_2}$ into the same shape as W at the last step of its calculation for convenience.

Suppose we initialize each component of the weight matrix independently according to Kaiming initialization $W_{ij} \sim \text{N}(0, \frac{2}{n})$, but we initialize the bias to be a large negative vector (e.g. $b = -10^6 \cdot \mathbf{1}$). Why would this be problematic? What would happen to the gradients of W and b during backpropagation?

Contributors:

- Zekai Wang.