

## TTIC 31230 Fundamentals of Deep Learning

### SGD Problems.

**Problem 1. Reformulating Momentum as a Running Average.** Consider the following running update equation.

$$\begin{aligned}y_0 &= 0 \\ y_t &= \left(1 - \frac{1}{N}\right) y_{t-1} + x_t\end{aligned}$$

(a) Assume that  $y_t$  converges to a limit, i.e., that  $\lim_{t \rightarrow \infty} y_t$  exists. If the input sequence is constant with  $x_t = c$  for all  $t \geq 1$ , what is  $\lim_{t \rightarrow \infty} y_t$ ? Give a derivation of your answer (Hint: you do not need to compute a closed form solution for  $y_t$ ).

**Solution:**

The limit  $y_\infty$  must satisfy

$$y_\infty = \left(1 - \frac{1}{N}\right) y_\infty + c$$

giving  $y_\infty = Nc$ .

(b)  $y_t$  is a running average of what quantity?

**Solution:** The update can be rewritten as

$$y_t = \left(1 - \frac{1}{N}\right) y_{t-1} + \frac{1}{N}(Nx_t)$$

so  $y_t$  is the running average of  $Nx_t$ .

(c) Express  $y_t$  as a function of  $\mu_t$  where  $\mu_t$  is defined by

$$\begin{aligned}\mu_0 &= 0 \\ \mu_t &= \left(1 - \frac{1}{N}\right) \mu_{t-1} + \frac{1}{N}x_t\end{aligned}$$

**Solution:**  $y_t$  is the running average of  $Nx_t$  which equals  $N$  times the running average of  $x_t$  so we have

$$y_t = N\mu_t$$

**Problem 2. Bias Correction** Consider the following update equation for computing  $y_1, \dots, y_t$  from  $x_1, \dots, x_t$ .

$$y_t = \left(1 - \frac{1}{\min(t, N)}\right) y_{t-1} + \frac{1}{\min(t, N)} x_t$$

If  $x_t = c$  for all  $t \geq 1$  give a closed form solution for  $y_t$ .

**Solution:** For  $t = 1$  we get  $y_1 = x_1 = c$ . We then get that  $y_{t+1}$  is a convex combination of  $y_t$  and  $x_t$  which maintains the invariant that  $y_t = c$ .

**Problem 3. Batch Size Coupling to RMSProp and Adam.** Consider the following for-loop representation of a batch of matrix-vector products.

$$\text{for } b, i, j \quad y[b, j] += W[j, i]x[b, i]$$

(a) Write the for-loop representation of back-propagation to  $W.\text{grad}$  following the convention that parameter gradients are averaged over the batch.

**Solution:**

$$\text{for } b, i, j \quad w.\text{grad}[j, i] += \frac{1}{B} y.\text{grad}[b, j]x[b, i]$$

(b) Write a for-loop representation for computing  $W.\text{grad}[b, i, j]$  where this is the derivative of loss with respect to  $W[i, j]$  for batch element  $b$ .

**Solution:**

$$\text{for } b, i, j \quad w.\text{grad}[b, j, i] += y.\text{grad}[b, j]x[b, i]$$

(c) Consider

$$W.\text{grad2}[j, i] = \frac{1}{B} \sum_b W.\text{grad}[b, j, i]^2$$

Is it possible to compute  $W.\text{grad2}[j, i]$  from  $W.\text{grad}[j, i]$ ? Explain your answer.

**Solution:** No.  $W.\text{grad2}[j, i]$  is the average over the batch of the square of the gradient. The average value does not determine the average square value — the average value does not determine the variance.

(d) Explain how your answer to (c) is related to batch size scaling of RMSProp and Adam.

**Solution:** Adam and RMSProp both compute a running average of  $\hat{g}[i]^2$  defined by

$$s_{t+1}[i] = \left(1 - \frac{1}{N_s}\right) s_t + \frac{1}{N_s} \hat{g}[i]^2$$

At batch sized greater than 1 this fails to take into account the variance of the gradients within the batch. This implies that  $s_t[i]$  will be reduced as the batch size increases and in the limit of large batches  $s_t[i]$  will converge to the mean squared rather than the second moment.