HW3 - Handwritten Assignment

Convolution (1%)

As we mentioned in class, image size may change after convolution layers. Consider a batch of image data with shape $(B, W, H, input\_channels)$, how will the shape change after the convolution layer?

$$Conv2D\ (input\_channels,\ output\_channels,\ kernel\_size = (k_1,\ k_2),$$
$$stride = (s_1,\ s_2),\ padding = (p_1,\ p_2))$$

To simplify the answer: the padding tuple means that we pad p_1 pixels on both left and right side, and p_2 pixels for top and bottom

Sol: (A student who only answers W' and H' will get 0.75pt for not answering the change of channels)
$(B, W', H', output\_channels)$

$$W' = \lfloor \frac{W + 2{*}p_1 - k_1}{s_1} + 1 \rfloor$$

$$H' = \lfloor \frac{H + 2{*}p_2 - k_2}{s_2} + 1 \rfloor$$

Batch Normalization (1%)

Besides ***Dropout***, we usualy use ***Batch Normalization*** in training nowadays [ref] (https://arxiv.org/pdf/1502.03167.pdf). The trick is popular whithin the deep networks due to its convenience while training. It preverses the distribution within hidden layers and avoids gradient vanish.

The alogrithm can be written as below:

$Input:\ values\ of\ x\ over\ a\ mini-batch:\ B = \{x_{1..m}\};$
$Output: y_i = BN_{\gamma,\beta}(x_i)$

$Parameters\ to\ be\ learned: \gamma,\ \beta$

$\mu_B \leftarrow \frac{1}{m}\sum_{i=1}^{m} x_i \qquad //mini-batch\ mean$

$\sigma_B^2 \leftarrow \frac{1}{m}\sum_{i=1}^{m}(x_i - \mu_B)^2 \qquad //mini-batch\ variance$

$\hat{x_i} \leftarrow \frac{x_i - \mu_B}{\sqrt{\sigma_B^2 + \epsilon}} \qquad //normalize$

$y_i \leftarrow \gamma\hat{x_i}\ +\ \beta \equiv BN_{\gamma,\beta}(x_i) \quad //scale\ and\ shift$

How to update γ and β from the optimization process of loss?

Just try to derive $\frac{\partial l}{\partial \hat{x_i}}, \frac{\partial l}{\partial \sigma_B^2}, \frac{\partial l}{\partial \mu_B}, \frac{\partial l}{\partial x_i}, \frac{\partial l}{\partial \gamma}, \frac{\partial l}{\partial \beta}$ (get 1pt if not left blank, 0.15pt for each answer)

Sol:
$\frac{\partial l}{\partial \hat{x_i}} = \frac{\partial l}{\partial y_i}\gamma$

$\frac{\partial l}{\partial \sigma_B^2} = \sum_{i=1}^{m} \frac{\partial l}{\partial \hat{x_i}} * (x_i - \mu_B) * \frac{-1}{2}(\sigma_B^2 + \epsilon)^{-3/2}$

$\frac{\partial l}{\partial \mu_B} = (\sum_{i=1}^{m} \frac{\partial l}{\partial \hat{x}_i} * \frac{-1}{\sqrt{\sigma_B^2 + \epsilon}}) + \frac{\partial l}{\partial \sigma_B^2} \frac{\sum_{i=1}^{m} -2(x_i - \mu_B)}{m}$

$\frac{\partial l}{\partial x_i} = \frac{\partial l}{\partial \hat{x}_i} * \frac{1}{\sqrt{\sigma_B^2 + \epsilon}} + \frac{\partial l}{\partial \sigma_B^2} * \frac{2(x_i - \mu_B)}{m} + \frac{\partial l}{\partial \mu_B} * \frac{1}{m}$

$\frac{\partial l}{\partial \gamma} = \sum_{i=1}^{m} \frac{\partial l}{\partial y_i} * \hat{x}_i$

$\frac{\partial l}{\partial \beta} = \sum_{i=1}^{m} \frac{\partial l}{\partial y_i}$

Softmax and Cross Entropy (1%)

In classification problem, we use softmax as activation function and cross entropy as loss function.

$softmax(z_t) = \frac{e^{z_t}}{\sum_i e^{z_i}}$

$cross\_entropy = L(y, \hat{y}) = -\sum_i y_i log \hat{y}_i$

$cross\_entropy = L_t(y_t, \hat{y}_t) = -y_t log \hat{y}_t$

$\hat{y}_t = softmax(z_t)$

Derive that $\frac{\partial L_t}{\partial z_t} = \hat{y}_t - y_t$

Sol:
In binary case (y_t = 1) :

$\frac{\partial L_t}{\partial z_t} = -\frac{\partial y_t log \hat{y}_t}{\partial z_t} = -y_t \frac{\partial log \hat{y}_t}{\partial z_t} = -y_t \frac{1}{\hat{y}_t} \frac{\partial \hat{y}_t}{\partial z_t} = -y_t \frac{1}{\hat{y}_t}(\hat{y}_t - \hat{y_t}^2) = y_t \hat{y}_t - y_t$

Adaptive learning rate based optimization (1%)

Adam optimizer is commonly used in deep learning applications. It combines two tricks, momentum and learning rate adaptation by gradients. Most of the time, Adam shows good results empirically. Here we show the updating schemes of AdaGrad, Adam as follows:

AdaGrad

$w^t = w^{t-1} - \frac{\eta}{\sqrt{\sum_{i=0}^{t} (g^i)^2}} g^t$, where w^i and g^i represent the model weights and the gradients in the i -th step; η represent the learning rate which is usually assigned as a constant(termed as η_0), that is, $\eta = \eta_0$.

Adam

$m^t = \beta_1 \cdot m^{t-1} + (1 - \beta_1) \cdot g^t$
$v^t = \beta_2 \cdot v^{t-1} + (1 - \beta_2) \cdot (g^t)^2$

$\hat{m^t} = \frac{m^t}{1 - \beta_1^t}$
$\hat{v^t} = \frac{v^t}{1 - \beta_2^t}$

$w^t = w^{t-1} - \frac{\eta}{\sqrt{\hat{v^t}}}\hat{m^t}$, where w^i and g^i represent the model weights and the gradients in the i-th step; β_1 and β_2 are the constants standing for the momentum estimates' exponetial decay rate. Before training, we initially set $m_0 = 0$ and $v_0 = 0$.

(a) Please rewrite m^t, v^t into the formation $m^t = A\sum_{i=1}^{t} B_i \cdot g^i$, $v^t = C\sum_{i=1}^{t} D_i \cdot (g^i)^2$ and derive the corresponding values of A, B_i, C, and D_i in terms of β_1 and β_2. Note that A, B_i, C and D_i are all scalars. As you derive this result, we expect you to learn that how β_1 and g_i affect the change of weights :P.

(b) Under the sitution that we set a scheduling scheme for the Adam optimizer such that $\eta = \eta_0 \cdot t^{-\frac{1}{2}}$, Adam corresponds to a version of AdaGrad(with $\eta = \eta_0$) if $\beta_1 = 0$ and $\beta_2 \to 1$. Please show the detailed derivation. (PS: If you are not able to prove it, you can still give the intuitive expression about this. Based on your answer, we might provide partial credits.)

Sol: (0.5pt for each sub-question)

(a) By induction, we can get $m^t = (1-\beta_1)\sum_{i=1}^{t} \beta_1^{t-i} \cdot g^i$; and on this basis, we can get $v^t = (1-\beta_2)\sum_{i=1}^{t} \beta_2^{t-i} \cdot (g^i)^2$. Thus, we can dervie $A = (1-\beta_1)$, $B_i = \beta_1^{t-i}$, $C = (1-\beta_2)$, and $D_i = \beta_2^{t-i}$.

(b) If we set $\beta_1 = 0$ and $\beta_2 \to 1$, Adam will behave the same with AdaGrad. Based on the result of (a), setting $\beta_1 = 0$ means that we don't take momentum for updating parameters, leading to $\hat{m^t} = g^t$. On the other hand, $(1-\beta_2^t)$ can be rewrited as $\frac{\sum_{i=1}^{t}\beta_2^{t-i}}{(1-\beta_2)}$, which can be substituted into $\hat{v^t}$. Then, we can get $\hat{v^t} = \frac{\sum_{i=1}^{t}\beta_2^{t-i}\cdot(g^i)^2}{\sum_{i=1}^{t}\beta_2^{t-i}}$. As we set $\beta_2 \to 1$, the result becomes $\hat{v^t} = \frac{\sum_{i=1}^{t}(g^i)^2}{t}$. Finally, we back to the definition of Adam and substitue $\eta = \eta_0 \cdot t^{-\frac{1}{2}}$, $\hat{v^t} = \frac{\sum_{i=1}^{t}(g^i)^2}{t}$, and $\hat{m^t} = g^t$ into it. Thus, we can derive that $w^t = w^{t-1} - \frac{t^{-\frac{1}{2}}}{t^{-\frac{1}{2}}} \cdot \frac{\eta 0}{\sqrt{\sum_{i=1}^{t}(g^i)^2}} g^t = w^{t-1} - \frac{\eta 0}{\sqrt{\sum_{i=0}^{t}(g^i)^2}} g^t$