Assignment #2 CSC 2516

Parsa Yousefpour
1004749659

1. Optimization

1.1.1 Minimum Norm Solution - Mini batch SGD

$w_{t+1} \leftarrow w_t - \frac{\eta}{b} \sum_{x_j \in B} \nabla_{w_t} \mathcal{L}(x_j, w_t) \longrightarrow \mathcal{L}(x_j, w_t) = (w_t^T x_j - t_j)^2 \rightarrow \nabla_{w_t} \mathcal{L}(x_j, w_t) = 2(w_t^T x_j - t_j) x_j$

$w_{t+1} \leftarrow w_t - \frac{2\eta}{b} \sum_{x_j \in B} (w_t^T x_j - t_j) x_j$ Update step

Since we start at $w_0 = 0$ and the update step is a linear combination of vectors in the row space, Because x_j is a row of X therefore $(\hat{w}_t^T x_j - t_j) x_j$ and its linear combinations are in row space of X. Since we start with $\hat{w} = 0$ which is in the row space of X and updating it (subtracting) with vectors in the row space of X. $\hat{w}$ will always be in the row space of X.

Hence we can say $\hat{w} = X^T \alpha$ for some $\alpha \in \mathbb{R}^n$

if $\hat{w}$ is a solution $\rightarrow X\hat{w} = t \xrightarrow{\text{replace with } \hat{w} = X^T\alpha} X(X^T\alpha) = t \longrightarrow \alpha = (XX^T)^{-1} t$

since $\hat{w} = X^T \alpha \xrightarrow{\text{replace } \alpha} \hat{w} = X^T (XX^T)^{-1} t$ which is the same solution from 1.3.2 or w^* therefore $\hat{w} = w^*$

1.2.1 Minimum Norm Solution - Adaptive Methods

RMS Prop

$\mathcal{L}(w) = \frac{1}{2}(x_1^T w - t)^2$

$w_{i,t+1} = w_{i,t} - \frac{\eta}{\sqrt{v_{i,t}} + \epsilon} \nabla_{w_{i,t}} \mathcal{L}(w_{i,t})$

$\nabla_{w_i} \mathcal{L}(w_i) = (x_1^T w - t) x_{1,i}$

$v_{i,t} = \beta(v_{i,t-1}) + (1-\beta)(\nabla_{w_{i,t}} \mathcal{L}(w_{j,t}))^2$

$x_1 = \binom{2}{1}$ $w_0 = \binom{0}{0}$ $t = 2$ $\beta = 0.9$ $\eta = 0.1$ $\epsilon = 0.01$ $t = 0$ $v_{i,-1} = 0$ $n = 1$

$t = 0$

$\nabla w_1 L(w) = (2 \cancel{w_{1,0}}^{0} + \cancel{w_{2,0}}^{0} - 2) 2 = -4$ $\nabla w_2 L(w) = (2 \underset{0}{w_{1,0}} + \underset{0}{w_{2,0}} - 2) 1 = -2$

$V_{1,0} = 0.9(0) + (0.1)(-4)^2 = 1.6$ $V_{2,0} = 0.9(0) + (0.1)(-2)^2 = 0.4$

$w_{1,1} = 0 - \frac{0.1}{\sqrt{1.6} + 0.01}(-4) = 0.314$ $w_{2,1} = 0 - \frac{0.1}{\sqrt{0.4} + 0.01}(-2) = 0.311$ $w_1 = \begin{bmatrix} 0.314 \\ 0.311 \end{bmatrix}$

$t=0$

$$\nabla_{w_1} L(w) = (2\overset{0.314}{w_{1,1}} + \overset{0.311}{w_{2,1}} - 2)\,2 = -2.12 \qquad \nabla_{w_2} L(w) = (2\underset{0.314}{w_{1,1}} + \underset{0.311}{w_{2,1}} - 2)\,1 = -1.06$$

$$V_{1,1} = 0.9(1.6) + 0.1(-2.12)^2 = 1.89 \qquad V_{2,1} = 0.9(0.4) + 0.1(-1.06)^2 = 0.47$$

$$w_{1,1} = 0.316 - \frac{0.1}{\sqrt{1.89 + 0.01}}(-2.12) = 0.469 \qquad w_{2,1} = 0.311 - \frac{0.1}{\sqrt{0.47 + 0.01}}(-1.06) = 0.463 \qquad w_2 = \begin{bmatrix} 0.469 \\ 0.463 \end{bmatrix}$$

The logic supporting the unique solution for minimum norm solution was that w was always in the span of X. However in this counter example neither weights are in the span of $\begin{bmatrix} 2 \\ 1 \end{bmatrix}$ hence RMSProp does not always obtain the minimum norm solution.

(expected to see a 2 to 1 relationship between x and y values. However, that is not the case hence not in the span of $\begin{bmatrix} 2 \\ 1 \end{bmatrix}$)

Gradient-based Hyper parameter Optimization

Optimal Learning Rate

2.1.1

$$\mathcal{L}_1 = \frac{1}{n}\| X w_1 - t \|_2^2 \qquad \nabla_{w_0} \mathcal{L} = \frac{2}{n} X^T (X w_0 - t)$$

$$w_1 = w_0 - \eta \nabla_{w_0} \mathcal{L} \qquad w_1 = w_0 - \frac{2\eta}{n} X^T (X w_0 - t)$$

$$w_1 = w_0 - \frac{2\eta}{n} X^T a \qquad a = X w_0 - t$$

$$\mathcal{L}_1 = \frac{1}{n}\left\| X\left(w_0 - \frac{2\eta}{n} X^T a\right) - t \right\|_2^2 = \frac{1}{n}\left\| X w_0 - \frac{2\eta}{n} X X^T a - t \right\|_2^2 = \frac{1}{n}\left\| X w_0 - t - \frac{2\eta}{n} X X^T a \right\|_2^2 = \frac{1}{n}\left\| a - \frac{2\eta}{n} X X^T a \right\|_2^2$$

$$\mathcal{L}_1 = \frac{1}{n}\left\| \left(I - \frac{2\eta}{n} X X^T\right) a \right\|_2^2 = \frac{1}{n} a^T \left(I - \frac{2\eta}{n} X X^T\right)^2 a$$

2.1.3

$$\frac{dL_1}{d\eta} = \frac{1}{d\eta}\left(\frac{1}{n}a^T\left(I - \frac{2\eta}{n}xx^T\right)^2 a\right) = \left(\frac{-2}{n}xx^T\right)\frac{2}{n}a^T\left(I - \frac{2\eta}{n}xx^T\right)a$$

Find η^*:

$$\frac{dL_1}{d\eta} = \left(\frac{-2}{n}xx^T\right)\frac{2}{n}a^T\left(I - \frac{2\eta}{n}xx^T\right)a = 0 \longrightarrow (xx^T)\,a^T\left(I - \frac{2\eta}{n}xx^T\right)a = 0$$

$$\frac{dL_1}{d\eta} = a^T(xx^T)a - \frac{2\eta}{n}a^Txx^Txx^Ta = 0 \longrightarrow \eta^* = \frac{n}{2}\cdot\frac{a^Txx^Ta}{a^Txx^Txx^Ta} = \frac{n}{2}\cdot\frac{(x^Ta)^2}{(xx^Ta)^2}$$

2.2 Weight decay and L_2 regularization

2.2.1

Regulized $\qquad \tilde{L} = \frac{1}{n}\|X\hat{\omega} - t\|_2^2 + \tilde{\lambda}\|\hat{\omega}\|_2^2 \qquad \nabla_{w_0}\tilde{L} = \frac{2}{n}x^T(xw_0 - t) + 2\tilde{\lambda}w_0$

$$w_1 = w_0 - \eta\nabla_{w_0}\tilde{L} \longrightarrow w_1 = w_0 - \eta\left(\frac{2}{n}x^T(xw_0 - t) + 2\lambda w_0\right) \longrightarrow w_1 = w_0 - \frac{2\eta}{n}x^T(xw_0 - t) - 2\eta\tilde{\lambda}w_0$$

$$w_1 = (1 - 2\eta\tilde{\lambda})w_0 - \frac{2\eta}{n}x^T(xw_0 - t)$$

UnRegulized + weigh decay $\qquad L = \frac{1}{n}\|X\hat{\omega} - t\|_2^2 \qquad \nabla_{w_0}\tilde{L} = \frac{2}{n}x^T(xw_0 - t)$

$$w_1 = (1 - \lambda)w_0 - \eta\frac{2}{n}x^T(xw_0 - t)$$

2.2.2

Regulized w_1:

$$w_1 = (1-2\eta\tilde{\lambda})w_0 - \frac{2\eta}{n} x^T(xw_0 - t)$$

Unregulized + weight decay w_1:

$$w_1 = (1-\lambda)w_0 - \eta \tfrac{2}{n} x^T(xw_0 - t)$$

$$(1-2\eta\tilde{\lambda})w_0 - \cancel{\frac{2\eta}{n} x^T(xw_0 - t)} = (1-\lambda)w_0 - \cancel{\eta \tfrac{2}{n} x^T(xw_0 - t)}$$

$$(1-2\eta\tilde{\lambda})\cancel{w_0} = (1-\lambda)\cancel{w_0} \longrightarrow 1-2\eta\tilde{\lambda} = 1-\lambda \longrightarrow \lambda = 2\eta\tilde{\lambda} \longrightarrow \tilde{\lambda} = \frac{\lambda}{2\eta}$$

Trading off Resources in Nueral NeT Training

3.1.1 Batch Size vs learning Rate

As you increase the batch size, the noise decreases. Because you have more samples you are avereging in a batch. With less noise the optimal Learning rate can be higher, because with less noise the direction of the gradient is more correct. Hence, as batch size increases the optimal Learning rate also increases.

3.1.2 Trainig Steps vs. Batch Size

a)

Option C: With a Batch size greater than C, we would be spending much more on Compute without gaining much speed in training time. As the number of training steps is not affected much from C to B. With a Smaller Batch size than C, we are significantly increasing our training time, which is inefficient. Hence, C's Batch size is the perfect balance between training time and Compute

b)

Point A Regime: noise dominated Potential ways to accelerate training: seek parallel Compute

Point A Regime: Curvature dominated Potential ways to accelerate training: use higher order optimizer.

3.2 Model size, dataset, and Compute

Option C: Increase the Model size

Reason: As shown in figure 2 for smaller models after some point the model stops improving significantly so an increase in the number of steps wouldn't help. Increasing batch size can't help as much as increasing model size. As show in figure 3, at the critical batch size, increasing the model size has the biggest effect on the test loss. Hence, Option C.

Programming Assignment

4.3

$\#\ \text{weights} = k^2 \times C_{in} \times C_{out}$ $\qquad$ $\#\ \text{outputs} = C_{out} \times W_{out} \times H_{out}$ $\qquad$ $\#\ \text{Connection} = k^2 C_{in} \times C_{out} \times H_{out} W_{out}$

For 32×32:

$$\text{Total weights} = \overset{9\,(\text{kernel size})}{\cancel{3\times 3}} \times NIC \times NF + 9 \times 2NF \times NF + 9 \times NF \times 2NF + 9 \times NF \times NC + 9NC^2 = 9 \times NIC \times NF + 36NF^2 + 9 \times NF \times NC + 9NC^2$$

$$\#\ \text{of Outputs} = \text{Conv2d\_1} + \text{Maxpool\_1} + \text{Conv2d\_2} + \text{Maxpool\_2} + \text{Conv2d\_3} + \text{Upsample\_3} + \text{Conv2d\_4} + \text{Upsample\_4} + \text{Conv2d\_5} =$$

$$NF \times 32 \times 32 + NF \times 16 \times 16 + 2NF \times 16 \times 16 + 2NF \times 8 \times 8 + NF \times 8 \times 8 + NF \times 16 \times 16 + NC \times 16 \times 16 + NC \times 32 \times 32 + NC \times 32 \times 32 = 2240NF + 2304NC$$

$$\#\ \text{of Connections} = \text{Conv2d\_1} + \text{Maxpool\_1} + \text{Conv2d\_2} + \text{Maxpool\_2} + \text{Conv2d\_3} + \text{Upsample\_3} + \text{Conv2d\_4} + \text{Upsample\_4} + \text{Conv2d\_5} =$$

$$9 \times NIC \times 32 \times 32 \times NF + NF \times 16 \times 16 \times 4 + 9N_F \times 2N_F \times 16 \times 16 + 2N_F \times 16 \times 16 + 9(2NF) \times NF \times 8 \times 8 + N_F \times 16 \times 16 + 9N_F \times N_C \times 16 \times 16 + N_C \times 32 \times 32 + 9N_C \times N_C \times 32 \times 32 =$$

$$9216\,NIC\,NF + 5760N_F^2 + 1792N_F + 2304N_FN_C + 1024N_C + 9216N_C^2$$

For 64×64:

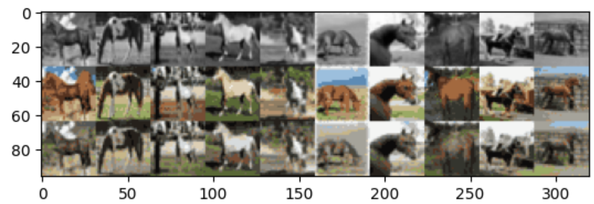
$\#\ \text{Total weights} = 9 \times NIC \times NF + 36NF^2 + 9 \times NF \times NC + 9NC^2$ Same as 32×32 as the input H and W have no effect on # weight

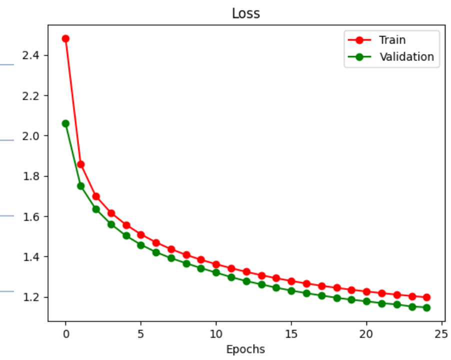
Since the input dimentions affect the # of outputs and # of Connection and we are doubling the width and height, the result is quadriple of the result in 32×32.

$\#\ \text{of output} = 8960\,NF + 9216\,NC$

$\#\ \text{of connections} = 36864\,NIC\,NF + 23040N_F^2 + 7168N_F + 9216N_FN_C + 4096N_C + 36864N_C^2$

5.2





5.3

The section 4 Model (PoolUpsampleNet) achieved an accuracy of 41.1% (Val. loss 1.588) whereas ConvTransposeNet achieved an accuracy of 55.4% (Val. loss 1.1468) hence it performed better than the first model.

The reason for ConvTransposeNet's better performance can be that Contraspose has more learnable parameters compared to poolupsample.

Another reason can be that the maxpool in PoolUpSampleNet is discarding information which is not being discarded in ConvTransposeNet.

5.4

Convolution: $Dim_{out} = \left\lfloor \frac{Dim_{in}+2P-k}{S} + 1 \right\rfloor$ Transpose Conv.: $Dim_{out} = (Dim_{in}-1) \times S - 2P + k + P_{out}$

$k = 4$

$16 = \frac{32+2P-4}{2} + 1 \longrightarrow P = 1$ $32 = (16-1) \times 2 - 2 \times 1 + 4 + P_{out} \longrightarrow P_{out} = 0$

Kernal size 4 padding = 1 output padding = 0

$k = 5$

$16 = \frac{32+2P-5}{2} + 1$ $P = 1.5 \approx 2$ $32 = (16-1) \times 2 - 2 \times 2 + 5 + P_{out} \longrightarrow P_{out} = 1$

Kernal size 5 padding = 2 output padding = 1