Homework 02 - Linear Algebra

Scott Chase Waggener
scw180000
CS6301.503 Redfern

January 30, 2019

As a Comet, I pledge honesty, integrity, and service in all that I do.

Problem 1

Complete

Problem 2

When multiplying MxK and KxN matrices we will produce a MxN output matrix where each element was the result of a linear combination of K elements. As such, we must compute M*N sums of K elements. Similarly, for memory operations we must bring in the original matrices and output the resulting matrix, leading to a sum over the dimensions of these matrices.

$$\texttt{Compute} = MNK \tag{1}$$
$$\texttt{Spatial} = KN + MK + MN \tag{2}$$

The proof of maximum intensity is as follows:

Proof. Let N be the largest term of N, M, K and let I denote the arithmetic intensity. Then we can express M, K in terms of a factor of N.

$$M = C_m N \qquad K = C_k N \qquad C_m, C_n \in \mathbb{R}^+ \tag{3}$$

Since we let N be the largest of the terms, C_m, C_k must be on the interval

$$0 < C_m, C_k \leq 1 \tag{4}$$

Then

$$\begin{aligned} I &= MNK + (KN + MK + MN) & (5) \\ &= C_m C_k N^3 + C_m C_k N^2 + C_m N^2 + C_k N^2 & (6) \end{aligned}$$

In this form it is clear that for $N \in \mathbb{N}^+$ and C_m, C_k on the interval described above that I will be maximized for

$$C_m = 1 \qquad\qquad C_k = 1 \tag{7}$$

So

$$M = C_m N = N \qquad\qquad K = C_k N = N \tag{8}$$

□

Problem 3

The given dense layer involves multiplying a $N_o \times N_i$ feature extractor matrix with a N_i vector. Under the BLAS notation we have

$$M = N_o = N_i \qquad\qquad K = N_i \qquad\qquad N = 1 \tag{9}$$

$$\tag{10}$$

We can then use the result of Problem 2 to compute the intensity

$$\begin{aligned} \texttt{MAC} &= MNK = N_i^2 & (11) \\ \texttt{Memory} &= (KN + MK + MN) = N_i^2 + 2N_i & (12) \\ & & (13) \end{aligned}$$

Vectorizing the given inputs prodcues vectors of the following lengths

$$
\begin{aligned}
\texttt{MNIST} &= (28 \times 28) & &= 784 & (14)\\
\texttt{CIFAR} &= (3 \times 32 \times 32) & &= 3072 & (15)\\
\texttt{ImageNet} &= (3 \times 224 \times 224) & &= 150528 & (16)\\
\texttt{Quasi 1/4 HD} &= (3 \times 512 \times 1024) & &= 1572864 & (17)\\
\texttt{Quadi HD} &= (3 \times 1024 \times 2048) & &= 6291456 & (18)\\
& & & & (19)
\end{aligned}
$$

Applying the derived total complexity

$$
\begin{aligned}
\texttt{MNIST} &= 2(784^2 + 784) = 1230880 & (20)\\
\texttt{CIFAR} &= 2(3072^2 + 3072) = 18880512 & (21)\\
\texttt{ImageNet} &= 2(150528^2 + 150528) = 4.53e10 & (22)\\
\texttt{Quasi 1/4 HD} &= 2(1572864^2 + 1572864) = 4.95e12 & (23)\\
\texttt{Quadi HD} &= 2(6291456^2 + 6291456) = 7.92e13 & (24)\\
& & (25)
\end{aligned}
$$

Problem 4

Problem 3 indicates that this approach will be limited by the square of memory movement. Given that memory movement is much less efficient than compute on modern hardware, practical implementations of this approach will be very inefficient. The number of learned weights for this layer will also scale by $N_o * N_i$, making it dependent on the input and output size.

This approach would also fail to provide the translational invariance of a CNN-style 2D convolution layer. Furthermore, vectorizing a 2D input discards potentially valuable information about the structure of the input data.

Problem 5

The dense layer would be incompatible with an input having N_i' greater than the trained input N_i. For an input having N_i' less than N_i it may be possible to pad the smaller input to fit inside a N_i vector, however this would likely produce poor results.

Problem 6

We defined a CNN-style 2D convolution layer to be of the form

$$\boldsymbol{Y}^{3D} = f\left(\boldsymbol{H}^{4D} \otimes \boldsymbol{X}^{3D} + \boldsymbol{V}^{3D}\right) \tag{26}$$

As in class, we can expand the inner product term for a single output feature map $n_o \in \{0, 1, ...N_o\}$. To do this, we first construct $\boldsymbol{H}$ (written as $\boldsymbol{H}^T$ to save space) as

$$\boldsymbol{H}(n_o, N_i, F_r, F_c)^T = \begin{bmatrix} \boldsymbol{H}(n_o, N_i, 0, 0) \\ \boldsymbol{H}(n_o, N_i, 0, 1) \\ \boldsymbol{H}(n_o, N_i, 1, 0) \\ ... \\ \boldsymbol{H}(n_o, N_i, F_r, F_c) \end{bmatrix} \tag{27}$$

This expresses $\boldsymbol{H}$ for a single output feature map as a vector where each entry represents filter coefficient f_r, f_c for each of N_i input feature maps.

Next we follow a similar process with $\boldsymbol{X}$.

$$\boldsymbol{X}(N_i, L_r, L_c) = \begin{bmatrix} \boldsymbol{X}(N_i, 0, 0) & & \boldsymbol{X}(N_i, L_r - F_r, L_c - F_c) \\ \boldsymbol{X}(N_i, 0, 1) & & \boldsymbol{X}(N_i, L_r - F_r, L_c - F_c + 1) \\ \boldsymbol{X}(N_i, 1, 0) & ... & \boldsymbol{X}(N_i, L_r - F_r + 1, L_c - F_c) \\ ... & & ... \\ \boldsymbol{X}(N_i, F_r, F_c) & & \boldsymbol{X}(N_i, L_r, L_c) \end{bmatrix} \tag{28}$$

This result expresses $\boldsymbol{X}$ as a matrix. Each column of this matrix represents one $F_r \times F_c$ window through N_i feature maps on which the feature extractor will act.
From this result we can see that the shape of $\boldsymbol{X}$ will be of the form

$$\boldsymbol{X} \in \mathbb{R}^{(F_r * F_c) \times ((L_r - F_r + 1) * (L_c - F_c + 1))} \tag{29}$$

$$M_r \times M_c = \big(L_r - F_r + 1\big) \times \big(L_c - F_c + 1\big) \tag{30}$$

Finally, we can convert this vector matrix multiplication to a matrix matrix multiplication by considering all output feature maps in N_o. We will then have a row in $\boldsymbol{H}$ for each $n_o \in N_o$.

Problem 7

Since we are considering a single output point, we can neglect the input feature map size. Following the proof in problem 6, we can express the CNN style 2D convolution in terms of $(F_r * F_c)$ matrix matrix multiplications. However, we can neglect the matrix dimension $(M_r * M_c)$ since we are considering a single output point.

$$\Bigg(N_o \times N_i\Bigg) \otimes \Bigg(N_i \times 1\Bigg) \tag{31}$$

This matrix matrix multiplication will require $N_o * N_i$ MACs, and we have a total of $(F_r * F_c)$ such multiplications to do. Thus for a single output point, the number of MACs is given by

$$\texttt{MACs} = F_r * F_c * N_o * N_i \tag{32}$$

Problem 8

First we will expand the result of problem 7 to all output points. This can be done by recognizing that we will have $M_r * M_c$ such output points on a feature map.

$$\begin{aligned} \texttt{MACs} &= F_r * F_c * N_o * N_i * M_r * M_c & (33)\\ &= F_r * F_c * N_o * N_i * \big(L_r - F_r + 1\big) * \big(L_c - F_c + 1\big) & (34) \end{aligned}$$

So for MACs we have the following scaling

$$(L_r * L_c) \to (L_r * L_c) \qquad \texttt{linear} \tag{35}$$
$$(F_r * F_c) \to (F_r * F_c)^2 \qquad \texttt{square} \tag{36}$$
$$(N_o * N_i) \to (N_o * N_i) \qquad \texttt{linear} \tag{37}$$

We follow a similar process for memory. The memory complexity for matrix matrix multiplication under BLAS notation is given by

$$\texttt{Memory} = (MK + KN + MN) \tag{38}$$

Under the lowering established in problem 6 we have

$$M = N_o \qquad K = N_i \qquad N = M_r * M_c \tag{39}$$

So for one of $F_r * F_c$ matrix matrix multiplications we have memory complexity

$$\texttt{Memory}_{f_r,f_c} = N_o * N_i + N_i * M_r * M_c + N_o * M_r * M_c \tag{40}$$

And in total

$$\texttt{Memory} = F_r * F_c\Bigg(N_o * N_i + N_i * M_r * M_c + N_o * M_r * M_c\Bigg) \tag{41}$$

Which gives the following scaling

$$(L_r * L_c) \to (L_r * L_c) \qquad \texttt{linear} \tag{42}$$
$$(F_r * F_c) \to (F_r * F_c)^2 \qquad \texttt{square} \tag{43}$$
$$(N_o * N_i) \to (N_o * N_i) \qquad \texttt{linear} \tag{44}$$

Problem 9

Padding input such that the output size is the same as the input size is half padding. For unit strides, half padding will require $P = \lfloor F/2 \rfloor$. For the given example we have

$$P_r = \left\lfloor \frac{F_r}{2} \right\rfloor \qquad P_c = \left\lfloor \frac{F_c}{2} \right\rfloor \tag{45}$$

Substituting the given cases we have

$$F_r = F_c = 0, \qquad P_r = P_c = \left\lfloor \frac{0}{2} \right\rfloor = 0 \tag{46}$$

$$F_r = F_c = 3, \qquad P_r = P_c = \left\lfloor \frac{3}{2} \right\rfloor = 1 \tag{47}$$

$$F_r = F_c = 5, \qquad P_r = P_C = \left\lfloor \frac{5}{2} \right\rfloor = 2 \tag{48}$$

Problem 10

Given an input of shape $L_r \times L_C$ will produce an output with N_o number of $M_r \times M_c$ feature maps where

$$M_r = L_r - F_r + 1 \qquad M_c = L_c - F_c + 1 \tag{49}$$

If we pad the input with

$$P_r = F_r - 1 \qquad P_c = F_c - 1 \tag{50}$$

Then we will have an input of shape

$$\begin{aligned} N_o \times (L_r + 2P_r) \quad &\times (L_c + 2P_c) && (51) \\ N_o \times (L_r + 2F_r - 2) &\times (L_c + 2F_c - 2) && (52) \\ & && (53) \end{aligned}$$

Based on this, we have

$$\begin{aligned} M_r' &= L_r' - F_r + 1 & M_c' &= L_c' - F_c + 1 \\ M_r' &= (L_r + 2F_r - 2) - F_r + 1 & M_c' &= (L_c + 2F_c - 2) - F_c + 1 \\ M_r' &= L_r + F_r - 1 & M_c' &= L_c + F_c - 1 \end{aligned} \tag{54}$$

For a stride of 1 the size of a single output feature map $n_o \in N_o$ will be

$$\big(L_r + F_r - 1\big) \times \big(L_c + F_c - 1\big) \tag{55}$$

For a stride of 2 we will take $\lfloor (L_r - F_r + 2p)/2 \rfloor$ steps plus 1 for the initial position to traverse a row. This leads to

(56)

Problem 11

In the case of a $3 \times 512 \times 1024$ input it would be possible to upsample this input to the expected $3 \times 1024 \times 2048$ via interpolation without compromising the original input.

For a $3 \times 512 \times 512$ input the results would be questionable as the original image is not in the same aspect ratio as the expected input.

Problem 12

The RNN layer has the form

$$\begin{align}
\boldsymbol{y}_t &= f(\boldsymbol{H}\boldsymbol{x}_t + \boldsymbol{G}\boldsymbol{y}_{t-1} + \boldsymbol{v}) \tag{57}\\
\boldsymbol{H} &\in \mathbb{R}^{M \times K} \tag{58}\\
\boldsymbol{x} &\in \mathbb{R}^{K \times 1} \tag{59}\\
\boldsymbol{G} &\in \mathbb{R}^{M \times M} \tag{60}\\
\boldsymbol{y} &\in \mathbb{R}^{M \times 1} \tag{61}
\end{align}$$

First recognize that in this form, the input to the activation function will be that of a dense layer plus some update vector. Now consider the case when the state update matrix $\boldsymbol{G}$ is not constrained to a diagonal. In this case each element of the update vector will be a linear combination of outputs from the previous iteration, ie

$$\boldsymbol{G}\boldsymbol{y}_{t-1} = \begin{bmatrix} g_{11} & g_{12} & \cdots & g_{1M} \\ g_{21} & g_{22} & \cdots & g_{2M} \\ \vdots & \vdots & \ddots & \vdots \\ g_{M1} & g_{M2} & \cdots & g_{MM} \end{bmatrix} \begin{bmatrix} y_1^{t-1} \\ y_2^{t-1} \\ \vdots \\ y_M^{t-1} \end{bmatrix} \tag{62}$$

$$= \begin{bmatrix} g_{11} * y_1^{t-1} + g_{12} * y_2^{t-1} + \cdots + g_{1M} * y_M^{t-1} \\ g_{21} * y_1^{t-1} + g_{22} * y_2^{t-1} + \cdots + g_{2M} * y_M^{t-1} \\ \vdots \\ g_{M1} * y_1^{t-1} + g_{M2} * y_2^{t-1} + \cdots + g_{MM} * y_M^{t-1} \end{bmatrix} \tag{63}$$

When $\boldsymbol{G}$ is constrained to the diagonal we retain only one term in the linear combination that compries each vector element, giving

$$\boldsymbol{G}\boldsymbol{y}_{t-1} = \begin{bmatrix} g_{11} * y_1^{t-1} \\ g_{22} * y_2^{t-1} \\ \vdots \\ g_{MM} * y_M^{t-1} \end{bmatrix} \tag{64}$$

The intuition behind this is that by choosing a diagonal $\boldsymbol{G}$ we can force output feature $m_i \in \boldsymbol{y}$ in iteration t to have a memory dependency only on the value of m_i in iteration $t-1$, rather than depending on all output features $m \in \boldsymbol{y}$ in iteration $t-1$.

Problem 13

In this example we have an input of $N_i = 1024$ feature maps, each of size 16×32. Global average pooling layers output a single average value for each feature map, so the size of the output will be a single vector of length $N_i = 1024$.

Computing the output of the global pooling layer will require MACs equal to the product of the input dimensions, ie

$$\begin{aligned} \texttt{MACs} &= N_i * L_r * L_c \qquad (65) \\ &= 1024 * 16 * 32 \qquad (66) \\ &= 524288 \qquad (67) \end{aligned}$$