Homework 01 – Linear Algebra

Arthur J. Redfern
arthur.redfern@utdallas.edu

---

0 Outline

1 Reading

1. Linear algebra
   Motivation: a xNN related linear algebra refresher
   https://github.com/arthurredfern/UT-Dallas-CS-6301-CNNs/blob/master/Lectures/xNNs_010_LinearAlgebra.pdf

Complete

2. A guide to convolution arithmetic for deep learning
   Motivation: an alternative presentation of CNN style 2D convolution; note that some of the notation used in this paper may differ than the notation used in class
   https://arxiv.org/abs/1603.07285

Complete

2 Theory

Matrix multiplication

3. [**2nd part is optional**] Part 1: What is the arithmetic intensity for matrix-matrix multiplication with sizes M, N and K (BLAS notation)? Part 2: Prove that arithmetic intensity for matrix-matrix multiplication is maximized when M = N = K (all matrices are square).

Part 1: Arithmetic intensity Ratio= MNK / (MN + MK + NK)

Part 2 (optional try): If M=N=K=x
   Arithmetic intensity = x^3/ $3x^2$ = x/3
   Maximum when all matrix are square

Dense layers

4. Consider a dense layer that transforms input feature vectors to output feature vectors of the same length ($N_o = N_i$). Ignoring the bias and pointwise nonlinearity, what is the complexity (MACs and memory) of this layer applied to inputs created from vectorized versions of the following:

MNIST:	1 x 28 x 28
CIFAR:	3 x 32 x 32
ImageNet:	3 x 224 x 224 (typical use)
Quasi 1/4 HD:	3 x 512 x 1024
Quasi HD:	3 x 1024 x 2048

Let N = Ni= N_o.
Memory = N x 1 input, N x N weight matrix and N x 1 output = N^2+ 2N elements (for large N this is ~ N^2)
MACs = matrix vector multiplication results in N^2

	N	MACs (=N^2)	Memory (=N^2+ 2N)
MNIST	28*28*1 = 784	782^2 = 614456	616224
CIFAR	32*32*3 = 3072	3072^2 = 9437184	9443328
ImageNet	3*224*224 = 150528	22658678784	22658979840
Quasi ¼ HD	3*512*1024 = 1572864	2473901162496	2473904308224
Quasi HD	3*1024*2048 = 6291456	39582418599936	39582431182848

5. In practice, why can't you flatten a quasi HD input image (3 x 1024 x 2048) to a 6291456 x 1 vector and use densely connected layers to transform from data to weak features to strong features to classes?

Flattening into a vector will lead to an impractical very high number of MACs and memory elements for current technology. This will likely be an insufficient amount of training data to effectively find that many coefficients.

6. Say I have trained a dense layer for an input of size 1024 x 1. Can this dense layer be applied to an input of size 2048 x 1? What about 512 x 1?

No, and No. A dense layer of 1024 X 1 **cannot** be applied to any other **input layer** because it must maintain compatible matrix vector multiplication dimensions.

CNN style 2D convolution layers

7. [**Optional**] Prove that CNN style 2D convolution with a N_o x N_i x F_r x F_c filter, N_i x L_r x L_c input and N_o x (L_r – F_r + 1) x (L_c – F_c + 1) output size can be lowered to the sum of (F_r*F_c) matrix matrix multiplications with N_o x N_i matrices made of filter coefficients where the elements of each matrix comes from a single $f_r \in \{0, ..., F_r – 1\}$ and single $f_c \in \{0, ..., F_c – 1\}$ index and N_i x (M_r*M_c) matrices made from inputs.
Not attempted

8. Consider a CNN style 2D convolution layer with filter size N_o x N_i x F_r x F_c. How many MACs are required to compute each output point?

MACs = N_i x F_r x F_c

9. How does CNN style 2D convolution complexity (MACs and memory) scale as a function of

Product of the image rows and cols (L_r*L_c)?

- MACs and feature map memory scale proportionally
- Filter memory is independent

Product of the filter rows and cols (F_r*F_c), assume N_i and N_o are fixed?

- MACs and filter memory scale proportionally
- Feature map memory is independent

Product of the number of input and output feature maps (N_i*N_o)?

- MACs and filter memory scale proportionally
- Feature map memory scales proportional to $(N_i*N_o)^{1/2}$ or ~ proportional to N_i or N_o

{

Assume L_r, L_c >> F_r, F_c

MACs = $N_o*(L_r - F_r + 1)*(L_c - F_c + 1)*N_i*F_r*F_c$

$\approx N_o*L_r*L_c*N_i*F_r*F_c$

Filter memory = $N_o*N_i*F_r*F_c$

Feature map memory = $N_o*(L_r - F_r + 1)*(L_c - F_c + 1) + N_i*L_r*L_c$

$\approx (N_i + N_o)*L_r*L_c$

}

10. Consider a CNN style 2D convolution layer with filter size N_o x N_i x F_r x F_c. How many 0s do I need to pad the input feature map with such that the output feature map is the same size as the input feature map (before 0 padding)? What is the size of the border of 0s for F_r = F_c = 1? What is the size of the border of 0s for F_r = F_c = 3? What is the size of the border of 0s for F_r = F_c = 5?

| Output feature map | = | Input feature map |

- Row padding of 0s = **F_r-1**
- Column padding of 0s = **F_c-1**

- Size of border of 0s for F_r = F_c = 1
  - **Zero**
- Size of border of 0s for F_r = F_c = 3
  - **Border of 1 pixel of 0s added on all sides**
- Size of border of 0s for F_r = F_c = 5
  - **Border of 2 pixel of 0x added on all sides**

11. Consider a CNN style 2D convolution layer with N_o x N_i x F_r x F_c filter, N_i x L_r x L_c input (L_r and L_c both even) and P_r = F_r – 1 and P_c = F_c – 1 zero padding. What is the size of the output feature map with striding S_r = S_c = 1 (no striding)? What is the size of the output feature map with striding S_r = S_c = 2? How does this change the shape of the equivalent lowered matrix equation?

- $S_r = S_c = 1$: the output feature map is of size N_o x L_r x L_c and the equivalent lowered matrix equation has BLAS notation dimensions of
  $M = N_o$
  $N = Lr*Lc$
  $K = Ni*Fr*Fc$

- $S_r = S_c = 2$: the output feature map is of size N_o x $(L_r/2)$ x $(L_c/2)$ and the equivalent lowered matrix equation has BLAS notation dimensions of
  $M = N_o$
  $N = (L_r/2) * (L_c/2) = L_r*L_c/4$
  $K = N_i*F_r*F_c$

- As such, for $S_r = S_c = 2$ the matrices $\mathbf{Y}^{2D}$ and $\mathbf{X}^{2D}$ have 1/4 the number of columns vs the $S_r = S_c = 1$ case.

12. Say I have trained a CNN style 2D convolution layer for an input of size 3 x 1024 x 2048. Can this CNN style 2D convolution layer be applied to an input of size 3 x 512 x 1024? What about 3 x 512 x 512?

Yes, and yes. A CNN style 2D convolution layer works for inputs with arbitrarily sized rows and cols but expects a specific number of input channels N_i to maintain compatible matrix-matrix multiplication dimensions.

RNN layers

13. In a standard RNN, if the state update matrix is constrained to a diagonal, what does this do for the mixing of the previous state with new inputs?

Let $\mathbf{y}_t = f(\mathbf{H}\,\mathbf{x}_t + \mathbf{G}\,\mathbf{y}_{t-1} + \mathbf{v})$ and constrain $\mathbf{G} = \mathrm{diag}(g(0), g(1), \ldots, G(M-1, M-1))$. The m^{th} element of the M x 1 output vector can be written as $y_t(m) = f(\mathbf{H}(m, :)\,\mathbf{x}_t + g(m)\,y_{t-1}(m) + v(m))$. The output feature at the previous time step $y_{t-1}(m)$ only interacts with the corresponding feature at the current time step via the scale $g(m)$. There is not mixing from multiple previous features to current feature.

Attention layers

14. Consider single headed self-attention where input $\mathbf{X}^T$ is a M x K matrix composed of M input vectors with K features per vector, $\mathbf{A}_i^T$ is a M x M attention matrix where each element is non negative and each row sums to 1 (think each row is a pmf), $\mathbf{W}_{v,i}$ is a K x L weight matrix and output $\mathbf{Y}_i^T$ is a M x L matrix composed of M output vectors with L features per vector

$$\mathbf{Y}_i^T = \mathbf{A}_i^T\,\mathbf{X}^T\,\mathbf{W}_{v,i}$$

Can $\mathbf{A}_i^T$ be computed once and then used for all inputs?

No, because $\mathbf{A}_i^T = \mathrm{softmax}_{row}(\mathbf{X}^T\,\mathbf{W}_{q,i}\,\mathbf{W}_{k,i}^T\,\mathbf{X} / P^{1/2})$ is a function of each input $\mathbf{X}^T$. Note that it may be beneficial to pre compute the $\mathbf{W}_{q,i}\,\mathbf{W}_{k,i}^T$ term depending on the matrix dimensions.

What does it mean for the output if A_i^T is an identity matrix?

If A_i^T is an identity matrix then each output vector is only a function of the corresponding input vector (no mixing across input vectors to create output vectors, very similar to a dense layer).

What does it mean for the output if $W_{v,i}$ is an identity matrix?

If $W_{v,i}$ is an identity matrix then each output feature is only a function of the corresponding input feature (no mixing across input features to create output features).

Average pooling layers

15. The size of the input to a global average pooling layer is 1024 x 16 x 32. What is the size of the output? What is the complexity (operations) of the layer?

Size of output = 1024 x 1.
Each element of the vector output requires summing the 16*32 = 512 elements of the corresponding feature map and dividing the result by 16*32 = 512. Calling these 512 operations per feature map, there are 1024*512 = 524288 operations total.

3 Practice

16. For network specification, training and evaluation, this class will use PyTorch (https://pytorch.org) running in Google Colab. Begin to familiarize yourself with PyTorch via the following tutorials:

- https://pytorch.org/tutorials/beginner/deep_learning_60min_blitz.html
- https://pytorch.org/tutorials/beginner/ptcheat.html
- https://pytorch.org/tutorials/recipes/recipes_index.html

Complete

17. The features of PyTorch are easier to understand and the problems are easier to debug with small datasets and simple networks (re: Andrej Karpathy's blog post you read for the previous homework assignment). This coding example will use MNIST (60k training and 10k testing 1 x 28 x 28 images of {0, …, 9} digits) with a few layer neural network for classification. Understand all lines of code in the following example (https://github.com/arthurredfern/UT-Dallas-CS-6301-CNNs/blob/master/Code/xNNs_Code_011_MNIST.py) and run it in Google Colab. Note how the code specifies a graph comprised of:

- Data
  - Source
  - Transformation
- Forward path

    - Encoder
    - Decoder
- Loss
- Backward path
    - Implicit
- Weight update

Feel free to modify the code and experiment (e.g., increase the level 0 and / or level 1 blocks, modify the number of channels in a level, try a different optimizer, change the batch size, add drop out, …). Experimentation will help you gain intuition.

Complete

Submitted by Kapil Gautam (KXG180032)