Basic Machine Learning Subsystems

Contents

1. Introduction

As the quantity and complexity of operations commonly used in Machine Learning continues to grow, the possibility for *incorrectness* increases. In order to manage that complexity, BMLS provides formal, peer-reviewed, unit tested and benchmarked implementations of these operators. This paper will provide the formal definitions for the subsystems provided by BMLS.

BMLS is *not* a machine learning library, it is a *math* library. It is intended for library implementers to ensure correctness and performance in their code, such as for reverse-mode automatic differentiation.

1.1. Features

1. Element-wise subsystems
2. Higher-order subsystems
3. Activations
4. Loss functions
5. Optimizer functions

1.2. Design Patterns

The inputs to BMLS subsystems are `&[f32]` or `&mut [f32]`. These slices are assumed to be row-major tensors laid out in NCHW format, where N is the batch size, C is the channels, H is the height, and W is the width. To ensure maximum performance, when iterating the W axis, the elements should be sequential. This is the standard convention used by most tensor libraries.

BMLS provides the `BMLSError` type, which is returned from every subsystem. Subsystems can fail in this way if the lengths of the provided slices is not what was expected, if the shape is invalid, or if other parameters are invalid.

1.3. Contributors

1. Rylan W. Yancey

1.4. Peer Reviewers

1. J. Leon Ballentine

2. Matmul

The Matmul operator computes the dot product of two matrices **A** and **B**. The dot product of two matrices is valid when **A** is an M x N matrix and **B** is an N x P matrix. This will produce an output **C** with the shape M x P. To compute the dot product, apply the following formula:

$$\sum_{i=0}^{M} \sum_{j=0}^{P} C_{i,j} = \sum_{k=0}^{N} A_{i,k} \times B_{k,j}$$

2.1. Proposition 1

The definition of an element $C_{i,j}$ is the vector dot product of row A_i and column B_j. The gradient of the vector product with respect to some factor is the factor it is multiplied by. We can extend this formula for a given element $A_{i,k}$ and $B_{k,j}$ by summing the gradients of the vector-vector products.

$$\frac{\delta C}{\delta A_{i,k}} = \sum_{j=0}^{P} B_{k,j}$$

$$\frac{\delta C}{\delta B_{k,j}} = \sum_{i=0}^{M} A_{i,k}$$

2.2. Proposition 2

Because the definition of the gradient of $C_{i,j}$ is defined in proposition 1, we can apply this formula to each element in **C**, summing the gradients for each factor. Assuming that δC is an identity matrix, the formula for the gradients w.r.t. **A** and **B** is defined as:

$$\frac{\delta C}{\delta A} = \sum_{i=0}^{M} \sum_{k=0}^{N} \frac{\delta C}{\delta A_{i,k}} = \sum_{j=0}^{P} \delta C_{i,j} \times B_{k,j}^{T}$$

$$\frac{\delta C}{\delta B} = \sum_{k=0}^{N} \sum_{j=0}^{P} \frac{\delta C}{\delta B_{k,j}} = \sum_{i=0}^{M} A_{i,k}^{T} \times \delta C_{i,j}$$

The gradient of some element $A_{i,k}$ becomes the sum of its multiplications in the forward operation. The same can be said for some element $B_{k,j}$, which becomes the sum of its multiplications as well.

2.3. Proposition 3

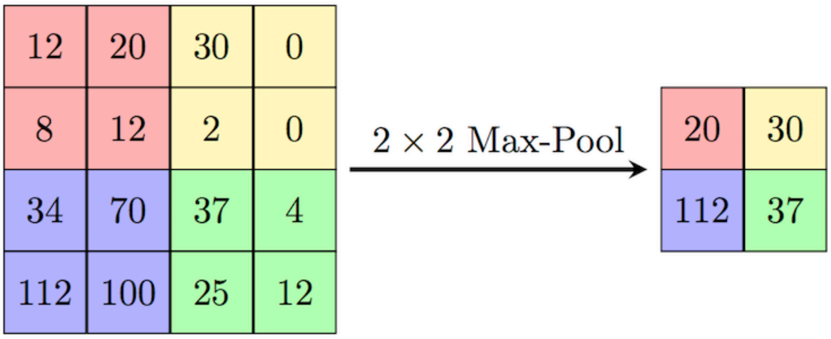
Based on the definition of the gradient defined in proposition 2, we can conclude that $\frac{\delta C}{\delta A}$ is the matrix dot product of δC and B^T, and that $\frac{\delta C}{\delta B}$ is the matrix dot product of A^T and δC.

$$\frac{\delta C}{\delta A} = \delta C \cdot B^T$$

$$\frac{\delta C}{\delta B} = A^T \cdot \delta C$$

3. Max Pool

The Max Pool operator finds the maximum value in a kernel, or subsection, of some matrix A. The operation requires a kernel size, a stride, a height padding, and a width padding.



12	20	30	0
8	12	2	0
34	70	37	4
112	100	25	12

20	30
112	37

Figure 1: Max Pooling with kernel size $[2, 2]$, stride $[2, 2]$, hpad $[0, 0]$, and wpad $[0, 0]$

In terms of BMLS data formats, `bmls::max_pool` will operate on the HW dimensions, treating it as a HxW matrix. When N and/or C are greater than 1, the operation remains unchanged, but is performed for NxC times, once for every HW. The resulting tensor B will have the shape:

1. B.N: A.N
2. B.C: A.C
3. B.H: ((A.H - kernel[0] + (hpad[0] + hpad[1])) / stride[0]) + 1
4. B.W: ((A.W - kernel[1] + (wpad[0] + wpad[1])) / stride[1]) + 1

3.1. Gradient

Any given $B_{i,j}$ is defined as the maximum value in the corresponding subsection of A. We can find the index of the top-left corner of the subsection by applying the formula $A_i = ((B_i * S_h) + K_h) - P_{h_0}$ and $A_j = \left(\left(B_j * S_w\right) + K_w\right) - P_{w_0}$. We can then iterate the subsection using the kernel height and width.

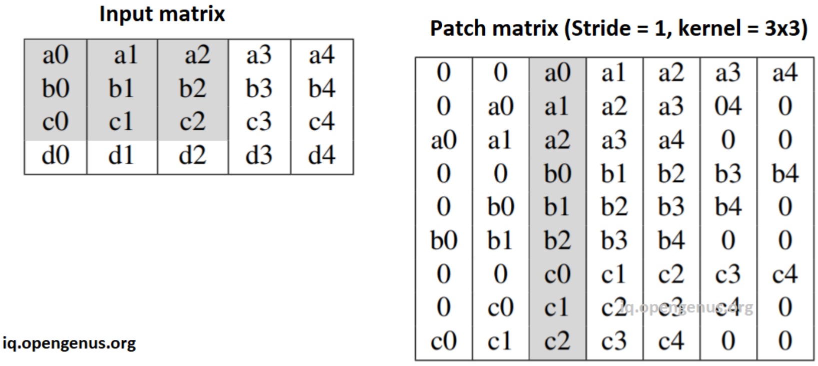
$$\sum_{h=0}^{B.h}\sum_{w=0}^{B.w} B_{h,w} = \max\left(\sum_{k_h=0}^{K_h}\sum_{k_w=0}^{K_w} A_{h*S_h+k_h-P_{h_0},w*S_w+k_w-P_{w_0}}\right)$$

The gradient of the max value function for a vector or a matrix (a.k.a. a subsection in our case) is 1 at the index of the max value, and 0 otherwise. Therefore, assuming δB is an identity matrix, the value of some element $\delta A_{i,j}$ is the number of times that element was the max value of a kernel. For example, if an element was the max value for 3 kernels, the value would be 3. In the case of Figure 1, the values at (0,1), (0,2), (3,1), and (2,2) would all be 1.

In the case that δB is an input gradient, we add the value $\delta B_{i,j}$ to the index of the maximum value in the corresponding kernel. Take Figure 1 for example, assuming that the gradient w.r.t. the output = `[1,2,3,4]`, (0,1) would be 1, (0,2) = 2, (3,1) = 3, and (2,2) = 4.

4. Im2Col

The Im2Col operator converts a 4-dimensional (batched, NCHW) image A into a 2-dimensional (NC) column matrix B, where each column in B corresponds to a kernel patch in A. The operation requires a filter size, a stride, a height padding, and a width padding.



Input matrix

a0	a1	a2	a3	a4
b0	b1	b2	b3	b4
c0	c1	c2	c3	c4
d0	d1	d2	d3	d4

Patch matrix (Stride = 1, kernel = 3x3)

0	0	a0	a1	a2	a3	a4
0	a0	a1	a2	a3	04	0
a0	a1	a2	a3	a4	0	0
0	0	b0	b1	b2	b3	b4
0	b0	b1	b2	b3	b4	0
b0	b1	b2	b3	b4	0	0
0	0	c0	c1	c2	c3	c4
0	c0	c1	c2	c3	c4	0
c0	c1	c2	c3	c4	0	0

iq.opengenus.org

Figure 2: Im2Col with filter size $[1, 1, 3, 3]$, stride $[1, 1]$, hpad $[0, 0]$, and wpad $[0, 0]$,

Bibliography