YAMML: Yet-Another-Matrix-Manipulation-Language

Name: Bill Chen, Janet Zhang, James Xu, Kent Hall, Doria Chen
Uni: gc2677, jz2896, jrx2000, dc3267

Feb 3, 2020

Introduction

The rise of machine learning has seen a rise in need of matrix based computations. Neural Networks used in deep learning, in essence, boils down to a series of matrix operations. Hence, a language that simplifies matrix operations can drastically simplify the workflow ML engineers and architects, allowing for fast prototyping, ease of use, and high likelihood of correctness. With that goal in mind, we aim to create an imperative language that has a familiar syntax to existing languages such as java and C++, while adding built-in support for matrix creating and a series of common matrix operations.

Language Definition

Our language has C-like syntax with strongly typing and curly brackets. Matrices are defined with square brackets and all the values having the same type. Rows are delineated with semicolons and columns are delineated with commas.

Supported Data Types

Our language supports basic primitive data types including int, float, string, boolean with the addition of a matrix structure which is written similar to an array in C. A matrix is defined as a rectangular 2d array that has a width w and a height h.

Examples

```
int i = 5;
float f = 1.6;
string s = "matrix";
boolean b = True;

int[] empty = int[3,3] // a 3x3 empty matrix
int[] m = [1,2,3;4,5,6;7,8,9];
int[] vertical = [1;2;3;4;5;6;7;8;9];
int[] horizontal = [1,2,3,4,5,6,7,8,9];

// for floating point precision, matrix should be declared as float types

float[] fm = [1,2,3;4,5,6;7,8,9];

// if an int[] matrix is declared with float values, the floating point values are automatically
    cast to int.

int[] m = [1,2.2]; // [1,2]
```

Supported Operations

Our language supports the basic C operations on integers and floats plus a few matrix operations that can be done in Python and MATLAB, such as matrix addition, subtraction, multiplication and transposition.

Basic arithmetic operations on integers and floats are equivalent to existing C++ operations. Additionally, our language supports the following matrix operations:

- + matrix addition
- - matrix subtraction
- * matrix multiplication
- .* element-wise matrix multiplication
- ./ element-wise matrix division
- M[i,j]: Indexing matrix
- M[i:j,k:l]: Slicing matrix

Examples

```
int[] A = [1,2,3;4,5,6;7,8,9]; // Declaring matrices
int[] B = [1,2,3;4,5,6;7,8,9];

C = A * B // [30,36,42;66,81,96;102,126,150];

D = 5.5 * A // Multiply each element of A by 5.5

a = A[0,0]; // 1

E = A[0:2, 0:2] // [1,2;4,5]
```

Control Flow

Our language supports if/else, for, and while, similar to the C language.

```
int greater_than_x(int l, int x) {
       if (l > x){
           return l;
       }
       else {
           while (l <= x){
               print (l);
               l++;
           }
           return l;
       }
   }
}
```

Built-in Functions

We include several basic matrix operations essential in implementing basic matrix algorithms.

- int printm(matrix m): print the matrix
- matrix size(matrix m): returns [height, width]
- float sum(matrix m): sum of all the elements in the matrix
- float mean(matrix m): average of all the elements in the matrix
- matrix trans(matrix m): transpose matrix
- matrix conv(matrix m, matrix k): convolve matrix with kernel
- matrix pad(matrix m, int k): pad the input matrix with 0, return the new matrix.

Example Code

```
float[] boxblur (float[] input) {

    float[] kernel = 1/9 * [1,1,1;1,1,1;1,1,1];
    float[] result = conv(pad(input,1), kernel)
    return result
}
```

One can imagine the input above being an input image that gets padded first and have the 3×3 boxblur kernel applied to it.

References

- http://www.cs.columbia.edu/ sedwards/classes/2018/4115-fall/proposals/MMM.pdf