PONTIFICIA UNIVERSIDAD CATÓLICA DE CHILE
ESCUELA DE INGENIERÍA
DEPARTAMENTO DE CIENCIA DE LA COMPUTACIÓN

IIC2343 - Arquitectura de Computadores
Septiembre 2021

Tarea 1: Bogo-NN

Fecha de entrega: 5 de octubre de 2021, hasta las 23:59.

Objetivos

Para esta tarea, esperamos que se familiaricen con el lenguaje de programación en assembly RISC-V[1] (*risk-five*), aplicando los conocimientos adquiridos hasta el momento, además que aprendan a utilizar el ambiente de programación y emulación de RARS[2], que será el emulador que utilizaremos durante el curso.

Enunciado

Para realizar la esperada Fonda Don Yadran es necesario contar con los mejores terremotos de todo el DCC. Para esto, te será entregada una lista de ingredientes (pipeño, granadina y helado de piña) y dos listas de pesos que representan la ponderación de cada ingrediente. Deberás determinar si la combinación de estos tres ingredientes en la proporción entregada, es la indicada o no para hacer un buen terremoto mediante una Bogo-NN (BNN).

Conceptos

Red Neuronal

Una red neuronal es un modelo simplificado que emula el modo en que el cerebro humano procesa la información, lo que permite extraer información útil y producir inferencias a partir de los datos disponibles gracias a su capacidad de aprendizaje.[1]

Perceptrón

El perceptrón es la forma más simple de una red neuronal usada para la clasificación de un tipo especial de patrones, los linealmente separables (es decir, una clasificación binaria). La estructura básica de un perceptrón es la siguiente:

[1] `https://riscv.org/`
[2] `https://github.com/TheThirdOne/rars`

- Inputs o Señales de Entrada: Datos con los cuales quieres clasificar.
- Pesos sinápticos: Constantes que multiplican a incógnitas (inputs) de la ecuación.
- Umbral: Punto que representa la división de las clasificaciones.
- Función de activación: función para determinar el valor del output.
- Output: Predicción de la clasificación.

[2]

Funcionamiento

Los inputs son los valores con los que esperas obtener una respuesta (output). Cada input es multiplicado por el peso correspondiente, luego los resultados son sumados junto con una constante δ. Este resultado es sometido a una Función de Activación según su valor sea superior o inferior al Umbral, la cual determina un output x.

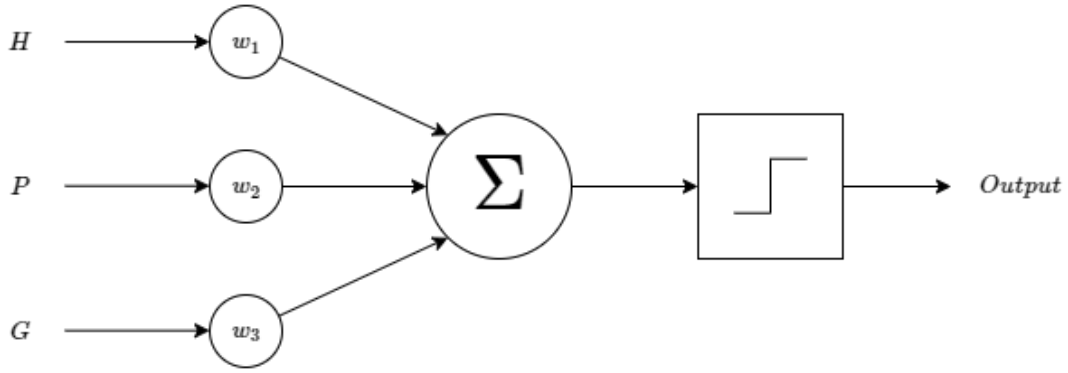

Figura 1: Perceptrón para el terremoto

Un buen Terremoto

Para clasificar un Terremoto como Bueno (1) o Malo (0) debes hacer la suma ponderada de cada entrada x_i de una lista de ingredientes y el peso w_i de una lista de proporciones w_A y la suma ponderada de cada entrada x_i y el peso w_i de una lista de proporciones w_B. El valor de cada una de estas sumas debe ingresar a la Función de activación $\mathcal{A}$ por separado, obteniéndose un output A y un output B. Si ambos output son 1, entonces el terremoto es Bueno. En cualquier otro caso el terremoto es Malo y no debería estar en la Fonda Don Yadran.

Función de activación $\mathcal{A}$

$$f(x) = \begin{cases} \mathcal{H}((3^x \,\%x)\,\%3) & \text{si } (w_0 * H + w_1 * P + w_2 * G) > u \\ \mathcal{H}((x-1)\,\%x) & \text{si } (w_0 * H + w_1 * P + w_2 * G) \leq u \end{cases}$$

34

[3] % es el operador módulo.
[4] $\mathcal{H}(x)$ es la función de Heaviside, por lo que $\mathcal{H}(x) = 0$ si $x \leq 0$, 1 si $x > 0$

Programación

Deberán construir una BNN, con dos perceptrones, A y B, cada uno como el descrito en la figura , y con sus respectivos pesos wa_i y wb_i, que serán entregados como parámetros, junto con el umbral u. Esta BNN recibirá como input el porcentaje de helado de piña, granadina y pipeño. Si el resultado de la función de activación $\mathcal{A}$ de cada perceptrón es 1, se concluye que el terremoto está Bueno, en cualquier otro caso, el terremoto está Malo.

Es necesario que tengan al menos un llamado a una subrutina en su programa (por ejemplo, la función $\mathcal{A}$), pero se recomienda utilizar más.

Input

Cada programa deberá tener la siguiente estructura

```
.globl  start
.data
    # --- TERREMOTO ---
    I: .word H, P, G # porcentajes de cada uno de los ingredientes, siempre suman 100
    # ↓↓ No modificar ↓↓
    Wa: .word 7, 3, 2 # pesos w_a para el primer perceptron
    Wb: .word 4, 2, 8 # pesos w_b para el segundo perceptron
    U:  .word 150 # umbral
    # ↑↑ No modificar ↑↑
    # --- END TERREMOTO ---
    # de aca para abajo van sus variables en memoria
.text
    start:
        # aca va su codigo :3
```

Los valores de los pesos se encuentran en el archivo `template_T1.txt` en el repositorio del curso, junto con este enunciado.

Output

0 o 1 en el registro `a0`, 1 indica que el terremoto está bueno, 0 que está malo.

Evaluación

Tests

Para un set de 4 inputs, tener todos los tests correctos son 1 puntos, los tests parcialmente correctos 0.5 punto y ningún test correcto, 0 puntos.

1 puntos.

Correctitud

Se evaluará en código que este implemente lo pedido en el enunciado, principalmente los siguentes aspectos:

- Uso adecuado de subrutina(s) **2 puntos.**
- Implementación correcta de la función de activación **0.5 puntos.**

- Uso adecuado del heap/stack **1 punto.**
- Lógica correcta, es decir, no hay saltos/comparaciones mal hechas **0.5 puntos.**

4 puntos.

Assembly

El código deberá estar adecuadamente comentado (0.5 pts), de manera de facilitar la corrección/*debugging*, además de respetar las convenciones de llamada para los registros (0.5 pts), especificadas en la segunda página del *green card*[5] de RISC-V. Por ejemplo, para guardar una variable en el registro `x5`, deberán hacerlo a través de `t0`. En su código nunca debiesen llamar a un registro a través de la notación `xN`, además de usar los registros de acuerdo con la descripción provista.

Adicionalmente, podrían haber **descuentos de hasta 1 punto por código muy desordenado**.

```
1        # para sumar dos numeros, escribir
2        addi a2, a2, 3
3        addi a3, a3, zero
4        add a4, a3, a4
5        # NO HACER
6        addi x12, x12, 3
7        addi x13, x13, x0
8        add x14, x13, x14
```

RISC-V Calling Convention			
Register	ABI Name	Saver	Description
x0	zero	---	Hard-wired zero
x1	ra	Caller	Return address
x2	sp	Callee	Stack pointer
x3	gp	---	Global pointer
x4	tp	---	Thread pointer
x5-7	t0-2	Caller	Temporaries
x8	s0/fp	Callee	Saved register/frame pointer
x9	s1	Callee	Saved register
x10-11	a0-1	Caller	Function arguments/return values
x12-17	a2-7	Caller	Function arguments
x18-27	s2-11	Callee	Saved registers
x28-31	t3-t6	Caller	Temporaries
f0-7	ft0-7	Caller	FP temporaries
f8-9	fs0-1	Callee	FP saved registers
f10-11	fa0-1	Caller	FP arguments/return values
f12-17	fa2-7	Caller	FP arguments
f18-27	fs2-11	Callee	FP saved registers
f28-31	ft8-11	Caller	FP temporaries

Figura 2: Convención de llamada para registros. Free & Open RISC-V Reference Card, RISC-V Organization.

1 punto.

[5] https://inst.eecs.berkeley.edu/~cs61c/fa17/img/riscvcard.pdf

Emulador (IMPORTANTE)

Si bien son libres de programar en el editor que prefieran e incluso compilar/emular en la herramienta que les sea más cómoda, la corrección será con el emulador RARS anteriormente mencionado, por lo que su tarea deberá pasar el *assembler* y ejecutar en dicho emulador. **No pasar la etapa del *assembler* en RARS implica 0 puntos en TODA la tarea**.

Se recomienda encarecidamente evitar el uso de tildes (‘, ’, ¨, etc.) u otros caracteres especiales en el código, ya que es probable que si hacen esto, el código se muestre con caracteres inválidos en el computador del ayudante por diferencias en cómo funciona el encoding/decoding de caracteres entre los diferentes sistemas operativos.

Nota tarea

La nota se calculará de la siguiente manera:

$$Puntos + 1 = Nota\ tarea$$

La nota de la tarea se redondea a la decena.

Entrega

La entrega de la tarea será a través de la plataforma SIDING en el formulario habilitado para ello, hasta el día 5 de octubre de 2021, a las 23:59. **No se entregará más plazo.**

El formato a entregar será un (1) archivo `.txt` con el código assembly RISC-V de la tarea. Entregar un archivo en cualquier otro formato implica un descuento de 1 punto.

Referencias

[1] IBM (2021). *El modelo de redes neuronales*. Recuperado de: https://www.ibm.com/docs/es/spss-modeler/SaaS?topic=networks-neural-model

[2] Perceptrón Simple Archivado el 21 de diciembre de 2012 en Wayback Machine., Redes de Neuronas Artificiales, UC3M, RAI 2012.