

## Programador neural

Verónica E. Arriola-Rios

Facultad de Ciencias, UNAM

24 de abril de 2023



# Concepto

- 1 Concepto
- 2 Conjunto de entrenamiento
- 3 Módulos
- 4 Entrenamiento
- 5 Experimentos

# Módulos

## Definición (Programador neural)

El *programador neural* propuesto por Neelakantan, Le y Sutskever 2016 es una RNN que se ejecuta durante  $T$  pasos, elegidos con antelación, para inducir programas con un máximo de  $T$  operaciones.

*“El modelo aprende de una señal de supervisión débil [conjunto de salidas esperadas] que es el resultado de la ejecución del programa correcto, por lo que no requiere de costosas anotaciones al programa correcto mismo.” Neelakantan, Le y Sutskever 2016*

# Programador neural

- El campo de la *síntesis de programas* estudia el problema de generar programas correctos que satisfagan diversas condiciones.
- El programador neural intenta acercarse a este campo entrenando redes neuronales, aunque aún es un intento incipiente.
- La elección de sobre qué datos operar y qué operaciones aplicar (*selección suave* o *atención*) se realiza de forma diferenciable, de modo que la red completa se pueda entrenar con métodos de optimización basados en el gradiente.
- Al probar la red **se reemplaza la selección suave por selección dura**.
- Entrenar el modelo es difícil, pero **agregar ruido aleatorio** al gradiente mejora notoriamente el proceso.
- Resuelven un problema que previas redes LSTM con atención no logran resolver.

# Funcionamiento

- Está aumentado con una unidad con un conjunto de operaciones lógico-matemáticas básicas en forma análoga a la ALU en la arquitectura de Von Neumann.

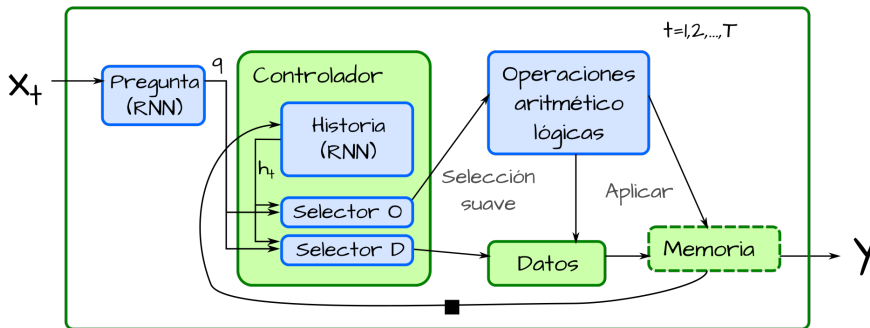


Figura: Diagrama del programador neuronal propuesto por Neelakantan, Le y Sutskever 2016.

## Conjunto de entrenamiento

- 1 Concepto
- 2 Conjunto de entrenamiento
- 3 Módulos
- 4 Entrenamiento
- 5 Experimentos

# Conjunto de entrenamiento

- Cada entrada es una tripla:

(pregunta, datos, respuesta)

- Los **datos** tiene la forma de una tabla,  $\text{tabla} \in \mathbb{R}^{M \times C}$  con  $M$  renglones y  $C$  columnas, que varían entre experimentos.
- En los experimentos de Neelakantan, Le y Sutskever 2016 los **segmentos de datos** son las columnas y cada columna tiene un nombre.

# Ejemplo

(pregunta, datos, respuesta)

**pregunta:** “What is the sum of elements in column B whose field in column C is word:1 and field in column A is word:7?”

**datos:** Tabla:

A	B	C
word:7	-10	word:1
word:6	22	word:2
word:7	45	word:1
word:6	-2	word:1

**respuesta:** 35



# Módulos

- 1 Concepto
- 2 Conjunto de entrenamiento
- 3 Módulos
- 4 Entrenamiento
- 5 Experimentos

# Módulos

El programador neural está compuesto por cuatro módulos:

**Pregunta** Una RNN que procesa la pregunta de entrada.

**Selector** Asigna una distribución de probabilidades sobre el conjunto de operaciones y otra sobre el de datos.

**Operaciones** Contiene la lista de operaciones que el modelo puede aplicar.

**Historia** Otra RNN para recordar las operaciones y segmentos de datos seleccionados por el modelo desde el inicio hasta el tiempo actual.

# Temas

## 3 Módulos

- Módulo pregunta
- Módulo selector
- Módulo historia
- Operaciones
- Texto

## Módulo pregunta

- Es una RNN codificadora sobre preguntas tipo  $Q = \{w_1, w_2, \dots, w_Q\}$ .

$$z_i = \tanh(W^{\text{pregunta}}[z_{i-1}; V(w_i)]), \forall i = 1, 2, \dots, Q$$

donde  $V(w_i) \in \mathbb{R}^d$  es la representación incrustada (*embedded*) de la palabra  $w_i$  y  $W^{\text{pregunta}} \in \mathbb{R}^{d \times 2d}$  es la matriz para la conexión recurrente.

- Para las preguntas más complejas la RNN es bidireccional.
- El estado inicial para la capa oculta se inicializa en ceros  $z_0 = [0]^d$ .
- Denotamos por  $q = z_q$  a la representación compacta de la pregunta que genera este módulo.
- Las **constantes numéricas** en la pregunta son separadas en preprocesamiento y son almacenadas en una lista aparte, al lado de la palabra que les precede en la oración.

# Temas

## 3 Módulos

- Módulo pregunta
- **Módulo selector**
- Módulo historia
- Operaciones
- Texto

# Módulo selector

## Selección de operaciones.

- Cada operación se representa con un vector  $d$ -dimensional.
- Todas las operaciones quedan dentro de una matriz  $U \in \mathbb{R}^{O \times D}$  donde  $O$  es el número de operaciones.
- La distribución de probabilidad para elegir una operación está dada por:

$$\alpha_t^{op} = \text{softmax}(U \tanh(W^{op}[q; h_t]))$$

donde  $W^{op}$  es una matriz de parámetros entrenables.

## Selección de datos.

- Sea  $P \in \mathbb{R}^{C \times D}$  la matriz que almacena los nombres de las columnas.
- La distribución de probabilidad para elegir una columna está dada por:

$$\alpha_t^{col} = \text{softmax}(P \tanh(W^{col}[q; h_t]))$$

donde  $W^{col}$  es una matriz de parámetros entrenables.

# Temas

## 3 Módulos

- Módulo pregunta
- Módulo selector
- **Módulo historia**
- Operaciones
- Texto

# RNN Historia

- La RNN para la historia es la memoria cuyo objetivo es almacenar información concentrada sobre las operaciones seleccionadas previamente y los datos sobre los que operaron.
- Recibe como entrada:

$$c_t = [(\alpha_{t-1}^{op})^T U; (\alpha_{t-1}^{col})^T P]$$

- Su estado oculto al tiempo  $t$  se calcula con:

$$h_t = \tanh(W^{\text{historia}}[c_t; h_{t-1}]), \quad \forall i = 1, 2, \dots, Q$$

donde  $W^{\text{historia}} \in \mathbb{R}^{d \times 3d}$  y  $h_t \in \mathbb{R}^d$ .

- Al tiempo  $t = 1$  el estado oculto se inicializa  $h_1 = [0]^d$



# Temas

## 3 Módulos

- Módulo pregunta
- Módulo selector
- Módulo historia
- Operaciones
- Texto

# Operaciones

## Estructuras

- Hay dos tipos de salidas:

**Escalar**  $\text{respuesta\_escalar}_t \in \mathbb{R}$

**Lista de elementos de la tabla**  $\text{respuesta\_búsqueda}_t \in [0, 1]^{M \times C}$  probabilidad de que cada celda esté en la consulta.

- La repuesta final del modelo es el contenido de la variable modificada al tiempo T.
- `selección_de_renglones` es una variable temporal.
- Los resultados se inicializan en zero y la selección de renglones en  $[1]^M$ .



# Operaciones

## Tiempo

- Las operaciones pueden acceder a todas las salidas anteriores a su tiempo  $t$ :

$$(\text{respuesta\_escalar}_i, \text{respuesta\_búsqueda}_i) \quad \forall_i = 1, 2, \dots, t-1$$

- Hay una operación de **reinicio** cuyo efecto es producir programas con menos de  $T$  operaciones.

# Operaciones

Tipo	Operación
<b>Escalares</b>	
Agregación	Suma $\text{sum}_t[j] = \sum_{i=1}^M \text{selección\_de\_renglones}_{t-1}[i] * \text{tabla}[i][j],$ $\forall j = 1, 2, \dots, C$
	Cuenta $\text{count}_t = \sum_{i=1}^M \text{selección\_de\_renglones}_{t-1}[i]$
Aritmética	Diferencia $\text{diff}_t = \text{respuesta\_escalar}_{t-3} - \text{respuesta\_escalar}_{t-1}$
<b>Búsqueda</b>	
Asignación de búsqueda	Asigna $\text{assign}_t[i][j] = \text{selección\_de\_renglones}_{t-1}[i],$ $\forall_{(i,j)} i = 1, 2, \dots, M, j = 1, 2, \dots, C$

Tipo	Operación
Selección de renglón	
Lógica	Y $\text{and}_t[i] = \min(\text{selección\_de\_renglones}_{t-1}[i],$ $\text{selección\_de\_renglones}_{t-2}[i]),$ $\forall_i = 1, 2, \dots, M$
	O $\text{or}_t[i] = \max(\text{selección\_de\_renglones}_{t-1}[i],$ $\text{selección\_de\_renglones}_{t-2}[i]),$ $\forall_i = 1, 2, \dots, M$
Reinicio	Reinicia $\text{reset}_t[i] = 1,$ $\forall_i = 1, 2, \dots, M$

Tipo	Operación
Selección de renglón (cont.)	
Comparación	Mayor qué $gt_t[i][j] = tabla[i][j] > pivote_g,$ $\forall_{(i,j)}, i = 1, 2, \dots, M, j = 1, 2, \dots, C$
	Menor qué $lt_t[i][j] = tabla[i][j] < pivote_l,$ $\forall_{(i,j)}, i = 1, 2, \dots, M, j = 1, 2, \dots, C$

- Se utiliza un mecanismo estilo atención para seleccionar el valor numérico en la pregunta que debe participar en la comparación.
- Para calcular los pivotes se usa la suma pesada de los valores numéricos  $qn_1, qn_2, \dots, qn_N$  que aparecen en la pregunta.
- Los pesos son asignados según:

$$\beta_{op} = \text{softmax}(ZU(op))$$
$$\text{pivote}_{op} = \sum_{i=1}^N \beta_{op}(i) qn_i$$

donde  $U(op) \in \mathbb{R}^d$  es la representación vectorial incrustada de la operación y  $Z \in \mathbb{R}^{N \times d}$  es la matriz con los vectores ocultos de la RNN pregunta en las posiciones a la izquierda de donde aparecen los números<sup>[1]</sup>.

---

<sup>[1]</sup>Para el idioma inglés.

# Operaciones

## Aplicación

**En cada tiempo se aplican todas las operaciones sobre todos los datos y su contribución va pesada por las probabilidades asignadas por el modelo.**

**Salidas:**

$$\text{respuesta\_escalar}_y = \alpha_t^{\text{op}}(\text{cuenta})\text{count}_t + \alpha_t^{\text{op}}(\text{diferencia})\text{diff}_t +$$

$$\sum_{j=1}^C \alpha_t^{\text{col}}(j) \alpha_t^{\text{op}}(\text{suma})\text{sum}_t[j]$$

$$\text{respuesta\_búsqueda}_t[i][j] = \alpha_t^{\text{col}}(j) \alpha_t^{\text{op}}(\text{asigna})\text{assign}_t[i][j],$$

$$\forall_{(i,j)} i = 1, 2, \dots, M, j = 1, 2, \dots, C$$



## Selector de renglones:

$$\begin{aligned} \text{selección\_de\_renglones}_t[i] = & \alpha_t^{\text{op}}(\text{y})\text{and}_t[i] + \alpha_t^{\text{op}}(\text{o})\text{or}_t[i] + \\ & \alpha_t^{\text{op}}(\text{reinicia})\text{reset}_t[i] + \\ & \sum_{j=1}^C \alpha_t^{\text{col}}(j) (\alpha_t^{\text{op}}(\text{mayor que})g_t[i][j] + \alpha_t^{\text{op}}(\text{menor que})l_t[i][j]), \\ & \forall_i, i = 1, \dots, M \end{aligned}$$

El mismo sistema se puede extender fácilmente para incluir equal, max, min, not, etc.

# Temas

## 3 Módulos

- Módulo pregunta
- Módulo selector
- Módulo historia
- Operaciones
- Texto

# Columnas con texto

- Sea  $K$  el número de columnas  $TC_1, TC_2, \dots, TC_K$  cuyos datos son texto.
- Los parámetros para representar vectorialmente las entradas en la columna de texto se comparten con el módulo *pregunta*.
- $A \in M \times K \times d$  almacena las representaciones vectoriales de las entradas de texto en tabla.



**Primera etapa:** Para cada renglón y columna de texto. Producto punto sobre las representaciones vectoriales (primer vector de atención):

$$B[m][k] = \text{sigmoid} \left( \sum_{p=1}^d A[m][k][p] \cdot q[p] \right) \quad \forall_{(m,k)} m = 1, \dots, M, k = 1, \dots, K$$

Representación de las columnas de interés para la pregunta en específico, se aplica el vector de atención:

$$D[k][p] = \frac{1}{M} \sum_{m=1}^M (B[m][k] \cdot A[m][k][p]) \quad \forall_{(k,p)} k = 1, \dots, K, p = 1, \dots, d$$

Agregando las representaciones de los nombres de las columnas P calculadas anteriormente se obtiene la representación para las columnas:

$$E = [D; P]$$

**Segunda etapa:** Vector de atención sobre las **columnas** de la tabla:

$$G = \text{softmax}(E \cdot H)$$

donde H son los estados ocultos de la RNN Pregunta. G es la combinación pesada de los estados ocultos de la RNN Pregunta.

**Coincidencia de texto:**

$$\text{text\_match}[m][k] = \text{sigmoid} \left( \sum_{p=1}^d A[m][k][p] \cdot G[k][p] \right),$$
$$\forall_{(m,k)} m = 1, \dots, M, k = 1, \dots, K$$

**Selector de renglones** cuando hay columnas con texto:

Colóquense las columnas de texto en las primeras C posiciones:

$$\begin{aligned} \text{selección\_de\_renglones}_t[i] = & \alpha_t^{\text{op}}(\text{y})\text{and}_t[i] + \alpha_t^{\text{op}}(\text{o})\text{or}_t[i] + \\ & \alpha_t^{\text{op}}(\text{reinicia})\text{reset}_t[i] + \\ & \sum_{j=K+1}^C \alpha_t^{\text{col}}(j) (\alpha_t^{\text{op}}(\text{mayor que})g_t[i][j] + \alpha_t^{\text{op}}(\text{menor que})l_t[i][j]) \\ & + \sum_{j=1}^K \alpha_t^{\text{col}}(j) (\alpha_t^{\text{op}}(\text{correspondencia de texto})\text{text\_match}_t[i][j]), \\ & \forall i, i = 1, \dots, M \end{aligned}$$

Se observó experimentalmente que ambas etapas son requeridas para obtener buenos resultados.

# Entrenamiento

- 1 Concepto
- 2 Conjunto de entrenamiento
- 3 Módulos
- 4 Entrenamiento**
- 5 Experimentos

# Parámetros entrenables

RNN Pregunta  $W^{\text{pregunta}}$

RNN Historia  $W^{\text{historia}}$

Incrustamientos de palabras  $V(.)$  y operaciones  $U$ .

Selector de operaciones  $W^{\text{op}}$  y de columnas  $W^{\text{col}}$



# Función de error

- Para salida **escalar** se usa la pérdida de Huber:

$$a = |\text{respuesta\_escalar}_T - y|$$
$$L_{\text{escalar}}(\text{respuesta\_escalar}_T, y) = \begin{cases} \frac{1}{2}a^2 & \text{si } a \leq \delta \\ \delta a - \frac{1}{2}\delta^2 & \text{de otro modo} \end{cases}$$

donde  $\delta$  es la constante de Huber y se trata como un hiperparámetro.

**Nota:** Los autores encontraron que diferencias al cuadrado hacen inestable el entrenamiento, mientras que el valor absoluto dificultan la optimización cerca del punto de discontinuidad, por esto eligieron esta fórmula.

- Para salida **lista de elementos** se utiliza entropía cruzada:
  - La salida se convierte a  $y \in \{0, 1\}^{M \times C}$

$$L_{\text{búsqueda}}(\text{respuesta\_búsqueda}_T) = -\frac{1}{MC} \sum_{i=1}^M \sum_{j=1}^C \left( y[i, j] \log(\text{respuesta\_búsqueda}_T[i, j]) + (1 - y[i, j]) \log(1 - \text{respuesta\_búsqueda}_T[i, j]) \right)$$

- La función de error queda:

$$L = \frac{1}{N} \sum_{k=1}^N \left[ n_k == \text{True} \right] L_{\text{escalar}}^{(k)} + [n_k == \text{False}] \lambda L_{\text{búsqueda}}^{(k)}$$

donde  $N$  es el número de ejemplares de entrenamiento y  $\lambda$  es un hiperparámetro para balancear ambos tipos de pérdidas.

# Experimentos

- 1 Concepto
- 2 Conjunto de entrenamiento
- 3 Módulos
- 4 Entrenamiento
- 5 Experimentos


# Experimentos I

- Los experimentos se realizan sobre datos generados sintéticamente.
- Los valores de las columnas en tiempo de entrenamiento se muestrean del rango  $[-100, 100]$ ; para prueba, de  $[-200, 200]$ .
- El número de renglones en la tabla durante el entrenamiento es de 120; en prueba es muestreado de  $[30, 100]$ .
- El conjunto de entrenamiento incluyó 50,000 triplas.
- El texto de las preguntas fue generado a partir de plantillas.
- Hubo diferentes tipos de pruebas, variando entre una columna numérica hasta 10, uso de frases equivalentes para referirse a la misma operación; más pequeños ejemplos donde un máximo de dos columnas contenían texto, acompañadas de hasta tres numéricas.
- El número de pasos temporales = número de operaciones llegó hasta  $T = 4$ .

# Experimentos II

- Se truncó el gradiente si éste rebasaba un valor umbral y se entrenó con Adam.
- La exactitud varió entre 80 % y 100 % dependiendo del experimento.

# Referencias I

 Neelakantan, Arvind, Quoc V. Le e Ilya Sutskever (2016). «Neural Programmer: Inducing Latent Programs with Gradient Descent». En: *Proceedings of the ICLR 2016*.

# Licencia

Creative Commons  
Atribución-No Comercial-Compartir Igual

