Artículo de investigación

Cómo citar: Castro-Manzano, J.-Martín., Amezquita-Paisano, S.-Jazmín., y, Fraile-Galaz, M. (2020). Lógica, Argumentación y educación. *Polisemia*, 16 (30), 04-23. http://doi.org/10.26620/uniminuto.polisemia.16.30.2020. 04-23

ISSN: 1900-4648

eISSN: 2590-8189

Editorial: Corporación Universitaria Minuto de Dios - UNIMINUTO

Recibido: 5 de Julio 2020

Aceptado: 15 de agosto 2020

Publicado: 15 de octubre 2020

Aprender silogística entrenando computadoras

Learning Syllogistic by Training Computers

Aprendendo silogística através do treinamento de computadores

J.-Martín Castro-Manzano

Facultad de Filosofía, UPAEP Universidad

Correo electrónico: josemartin.castro@upaep.mx

S.-Jazmín Amezquita-Paisano

Facultad de Filosofía, UPAEP Universidad

Correo electrónico: silviajazmin.amezquita@upaep.edu.mx

María Fraile-Galaz

Facultad de Filosofía, UPAEP Universidad

Correo electrónico: maria.fraile@upaep.edu.mx

Resumen

Inspirados por los mecanismos y los resultados del método de aprendizaje mediante la enseñanza (LdL, siglas en alemán de *Lernen durch Lehren*), en este trabajo sugerimos un modelo didáctico para aprender lógica enseñando lógica. En particular, proponemos un modelo para aprender silogística entrenando una red neuronal artificial. Al final exponemos un reporte cualitativo de dos experiencias docentes usando el modelo propuesto.

Palabras clave: modelo didáctico LdL, aprender enseñando, aprendizaje mediante la enseñanza, aprendizaje automático, red neuronal artificial, silogística

Abstract

Inspired by the mechanisms and results of the learning by teaching method (LdL, German abreviation for *Lernen durch Lehren*), in this paper we advance a didactic model to learn logic by teaching logic. In particular, we propose a model to learn syllogistic by training an artificial neural network. At the end we show a qualitative report of two teaching experiences using the proposed model.

Keywords: LdL didactic model, learning by teaching, automatic learning, artificial neural network, syllogistic.

Resumo

Inspirado pelos mecanismos e resultados do método "aprender ensinando" (LdL, acrônimo alemão para *Lernen durch Lehren*), neste artigo sugerimos um modelo didático para a lógica de aprendizagem através do ensino da lógica. Em particular, propomos um modelo de aprendizagem da silogística através do treinamento de uma rede neural artificial. No final, apresentamos um relatório qualitativo de duas experiências de ensino utilizando o modelo proposto.

Palavras-chave: modelo didático LdL, aprendizagem pelo ensino, aprendizagem pelo ensino, aprendizagem de máquinas, rede neural artificial, silogística

Introducción

Existe evidencia de que el simple hecho de enseñar favorece el aprendizaje (Bargh y Schul, 1980; Cohen, Kulik y Kulik, 1982; Roscoe y Chi, 2008; Safiye, 2015). Por supuesto, aunque las causas exactas detrás de este efecto aún se estudian y se discuten —especialmente por la influencia de la recuperación de información o *retrieval* (Koh et al., 2018)—, el método de aprendizaje mediante la enseñanza (LdL, siglas en alemán de *Lernen durch Lehren*) (cfr. Martin y Kelchner, 1998) parece ser exitoso.

En este trabajo, inspirado por los mecanismos y los resultados de este método, sugerimos un modelo didáctico para aprender enseñando. Se trata, en particular, de un modelo para aprender silogística mediante el entrenamiento de una red neuronal artificial. Para presentar este modelo y explorar algunas de sus cualidades, hemos estructurado este artículo de la siguiente manera: primero describimos algunos elementos preliminares sobre demostración automática y aprendizaje computacional; posteriormente, exponemos nuestra propuesta, incluyendo algunos ejemplos prácticos, así como un breve recuento cualitativo de la experiencia de dos estudiantes de filosofía con la aplicación del modelo, y, por último, presentamos algunas conclusiones.

1. Preliminares

1.1 Demostración automática y silogística

La demostración automática de teoremas (Huth y Ryan, 2004; Gallier, 2015) es un área de la inteligencia artificial que se asocia con la lógica y los sistemas formales, y, por lo tanto, con el paradigma simbólico *neat,* y la inteligencia artificial simbólica, GOFAI (siglas en inglés de *good old-fashioned artificial intelligence*), en términos de Haugeland (1989).

Así, por sus orígenes, esta disciplina está asociada naturalmente con métodos y estrategias de demostración para sistemas de primer orden; sin embargo, como en este caso trabajamos con la silogística y no estamos desarrollando, propiamente hablando, un demostrador de teoremas sino, como veremos, un clasificador de silogismos, lo que necesitamos es un conjunto de métodos y normas de demostración silogística (para un resumen general de la silogística, véase el apéndice A).

Por ejemplo, de los posibles conjuntos de métodos y normas para la silogística, hemos seleccionado uno, más o menos estándar, que sostiene que un silogismo categórico es correcto *si* y *solo si* cumple con las siguientes condiciones:

1. El término medio está distribuido por lo menos en una premisa (por mor de la brevedad, llamaremos a esta norma *mediación).*

2. Todo término distribuido en la conclusión está distribuido en las premisas (llamaremos a esta norma *distributividad)*.

3. El número de premisas negativas es igual al número de conclusiones negativas (*cualidad*).

4. El número de premisas particulares es igual al número de conclusiones particulares (*cantidad*).

5. El número de términos es igual tres (*términos*).

Así pues, siguiendo con este ejemplo, el silogismo del cuadro 1-a es incorrecto, pero el del cuadro 1-b es correcto:

Cuadro 1. Ejemplos de silogismos

1. Todo P es M	1. Todo M es P
2. Todo S es M	2. Todo S es M
⊬ Algún S es P	⊢ Todo S es P
(a)	(b)

En efecto, el silogismo del cuadro 1-a es incorrecto porque tiene problemas de mediación (el término medio, *M*, no aparece distribuido en las premisas) y de cantidad (el número de conclusiones particulares no es igual al número de premisas particulares), lo cual, por cierto, muestra que un silogismo puede ser incorrecto por más de una causa. En contraste, el silogismo del cuadro 1-b es correcto porque cumple con todas las normas de corrección, lo cual indica que hay solamente un camino para producir un silogismo correcto.

Por supuesto, también podríamos seleccionar otro conjunto de normas, como el que sostiene que un silogismo categórico es correcto *si y solo si* cumple con las siguientes reglas (Sommers y Englebretsen, 2000):

1. La suma algebraica de las premisas es igual a la conclusión (*sumatoria).*

2. El número de premisas particulares es igual al número de conclusiones particulares *(cantidad).*

Bajo este otro conjunto de normas, el silogismo del cuadro 1-a es incorrecto porque tiene problemas de sumatoria (la suma de las premisas no es igual a la conclusión) y de cantidad (el número de conclusiones particulares no es igual al número de premisas particulares); pero el silogismo del cuadro 1-b es correcto porque cumple con las dos normas.

Lo que esta breve presentación sugiere es que una misma base deductiva, la silogística en este caso, puede tener diferentes conjuntos de normas y, ciertamente, aunque podríamos haber elegido otros conjuntos, hemos

decidido comenzar con estos dos por razones de claridad expositiva. Más adelante comentaremos cómo es que el hecho de tener a la mano varios conjuntos de normas puede ser benéfico para el modelo didáctico que estamos sugiriendo.

1.2 Aprendizaje automático y redes neuronales artificiales

A diferencia de la demostración automática, el aprendizaje automático se asocia más bien con proyectos subsimbólicos, *scruffies*, y con la *nouvelle AI* (*cfr*. Brooks, 1999). En otras palabras, mientras que la demostración de teoremas se estudia, por lo general, desde un enfoque, digamos, *top-down* (porque supone el establecimiento de normas preestablecidas para resolver casos particulares); el aprendizaje, típicamente, se estudia más bien desde una óptica *bottom-up* (en la medida en que comienza con casos particulares para producir normas generales).

Entre la variedad de herramientas para implementar aprendizaje automático hemos elegido, por razones que mencionaremos más adelante, el modelo computacional bioinspirado de las redes neuronales artificiales mediante un algoritmo de retropropagación del error (EBP, siglas en inglés de *error back-propagation*). Para explicar la naturaleza de este algoritmo y su relevancia en este contexto, es preciso empezar con algunos comentarios históricos.

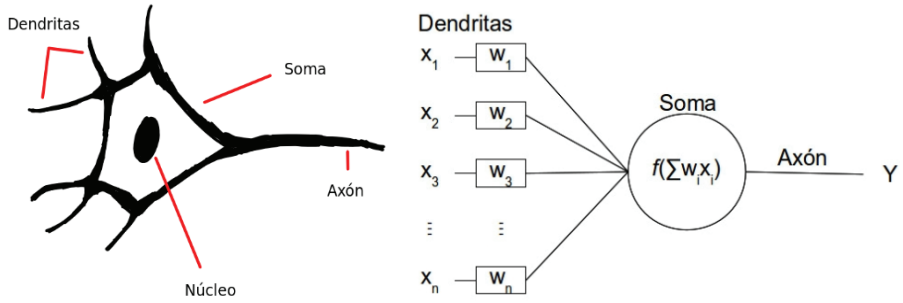
El primer modelo computacional de una neurona fue el modelo lógico propuesto por McCulloch y Pitts (1943), quienes lograron dos cosas de suma importancia: introducir la vía de la computación conexionista y representar el mecanismo de una neurona usando funciones lógicas en el contexto de un álgebra booleana.

Siguiendo este paradigma, Rosenblatt (1957) desarrolló lo que ahora se conoce como perceptrón simple. Un perceptrón simple es una función que toma como entrada un vector $W_i\ X_i$ y produce como resultado una salida dado el siguiente mapeo:

$$f(W_i X_i) = \left\{ \begin{array}{l} 1, \text{ si } W_i X_i + b > u \\ 0, \textit{ de otro modo} \end{array} \right\}$$

donde $X_i = x_1, \ldots, x_n$ representa las n entradas que puede tener una neurona y $W_i = w_1, \ldots, w_n$ representa los pesos asociados a cada neurona de entrada. La función f es, entonces, una función de activación tal que si obtiene un resultado mayor que cierto umbral definido u (como un umbral de voltaje), "dispara", es decir, produce un 1 como salida; y de otro modo, se "inhibe", produciendo un cero (0). De esta forma, un perceptrón simple resulta ser un modelo matemático de una neurona (figura 1).

Figura 1. Representación de un perceptrón simple.



(a) Neurona (b) Representación

Fuente: Adaptado de Russell y Norvig (1995, p. 565 y ss*)*.

Como es fácil notar, la función de activación de un perceptrón es lineal, por lo que es muy útil para discriminar ítems en un plano linealmente separable. Sin embargo, Minsky y Papert (1969) demostraron que este modelo es incapaz de distinguir ítems dadas ciertas funciones lógicas, como la función XOR. Ante este problema, Rumelhart et al. (1986), al mismo tiempo que otros investigadores, crearon un algoritmo que reutilizaba la idea básica del perceptrón junto con una modificación ingeniosa que hoy se conoce como EBP (para un resumen de este algoritmo véase el apéndice B).

Este algoritmo, que hoy por hoy es uno de los más populares y efectivos, se define por una red neuronal de al menos tres capas de neuronas artificiales: una de entrada, una intermedia y una de salida. La capa de neuronas de entrada se mapea en, por lo menos, una capa intermedia oculta, que a su vez se dirige hacia una capa de neuronas de salida, formando así una red de mayor complejidad.

Una vez que la información de entrada pasa por todas las neuronas y llega a la capa de salida, se evalúa si el resultado obtenido es el deseado, pues este algoritmo está diseñado para optimizar (en este caso, minimizar) la diferencia entre las entradas y las salidas esperadas; y mientras este error no sea el óptimo (*i. e.* el mínimo), una red neuronal de este tipo lleva a cabo una actualización de su estructura interna usando una función que se conoce como regla delta (véase el apéndice B).

Al proceso —como al resultado— de esta actualización lo llamamos *aprendizaje*, ya que nos permite optimizar una función de pérdida a partir de un conjunto de ejemplos singulares, como cuando afirmamos que una persona ha aprendido algo nuevo con base en la medición de la diferencia entre su estado epistémico final y su estado epistémico inicial después de exponerla a experiencias concretas. Y esto, como decíamos renglones arriba, es relevante para nuestros fines porque un modelo didáctico parece tener sentido solo en un contexto en el que el aprendizaje sea posible.

* * *

A modo de resumen, lo que nos gustaría aclarar con estos elementos preliminares es que no estamos proponiendo otro demostrador de teoremas ni otro modelo formal de aprendizaje. Lo que buscamos es más simple: aprovechar un modelo de aprendizaje automático para clasificar silogismos, asumiendo los mecanismos de demostración silogística. Lo que pretendemos a continuación es explicitar cómo este propósito puede resultar útil para la didáctica de la lógica.

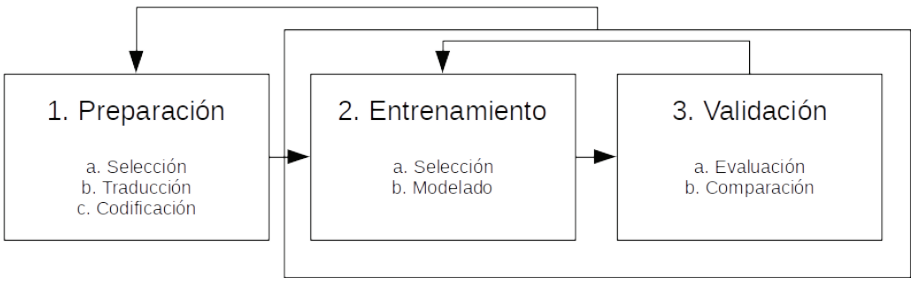
2. Modelo didáctico

Pues bien, dado que es posible diseñar una red neuronal artificial para que aprenda a clasificar silogismos, por ejemplo, por medio de un algoritmo EBP, ¿cómo podríamos aprovechar esto para aprender lógica y, en particular, silogística?

Pensemos en una computadora personal equipada con un algoritmo EBP como un agente completamente ignorante e irracional —como una *tabula rasa*— que cuenta con el talento suficiente para explorar información y superar su ignorancia, y la habilidad necesaria para explotar sus capacidades y salir de su irracionalidad —como un esquematismo *a priori*—. El modelo didáctico que proponemos, asumiendo el método de aprendizaje a través de la enseñanza, consiste en asumir un papel docente para ayudar a un agente de esta naturaleza a salir de sus estados iniciales de ignorancia e irracionalidad.

Para explicitar cómo podemos asumir dicho papel y, en el ínterin, explicar cómo funciona esta propuesta, ofrecemos a continuación algunos detalles de la estructura general del modelo didáctico (figura 2).

Figura 2. Estructura del modelo didáctico.



Este modelo presenta tres procesos seriados (preparación, entrenamiento y validación) y dos fases de retroalimentación (la primera va de la validación al entrenamiento; y la segunda, del resultado de estos dos últimos procesos a la preparación); y cada proceso, a su vez, contiene subprocesos ordenados (p. ej., la preparación conlleva subprocesos de selección, traducción y codificación, en ese orden). A continuación, mostramos algunas especificaciones y ejemplos de estos procesos.

1. Preparación

Este proceso consiste en organizar y acondicionar la información que nuestro agente ha de explorar, e incluye tres subprocesos: *a)* selección, *b)* traducción *y c)* codificación.

a. Selección

Comenzamos el proceso de preparación organizando la información en una base de datos. Para ello basta con obtener o generar un conjunto de n inferencias (in)correctas arbitrarias y bien etiquetadas, esto es, un conjunto de n silogismos arbitrarios debidamente identificados como correctos o incorrectos (para hacer esto es útil conocer las normas de demostración silogística). Posteriormente, dada una selección aleatoria de k elementos de este conjunto, la base de datos luciría, por ejemplo, como se muestra en el cuadro 2.

Cuadro 2. Base de datos

Ítem	Silogismo	Etiqueta
1.	Como calcular implica pensar y las computadoras calculan, las computadoras piensan.	Correcto
2.	Los animales necesitan agua para vivir, dado que los mamíferos necesitan agua para vivir y los mamíferos son animales	Incorrecto (distributividad)
⋮	⋮	⋮
k.	Puesto que hay profesores listos y profesores introvertidos, se sigue que algunas personas listas son también introvertidas.	Incorrecto (cantidad)

Vale la pena notar dos ventajas de este procedimiento: la primera, que por la naturaleza de los datos y su generación, no hay datos faltantes ni *outliers*; y la segunda, que por la naturaleza aleatoria de la selección, se reduce la presencia de sesgos o preferencias personales en la base de datos.

b. Traducción

En un segundo momento necesitamos explicitar la estructura de cada silogismo, con el propósito de normalizar la base de datos. Para ello hemos de traducir los ítems de la base de datos a un sistema apropiado, por ejemplo, haciendo una traducción (en lenguaje) natural. En este caso, en consecuencia, nuestra base de datos luciría como se muestra en el cuadro 3.

Este subproceso es interesante porque sugiere que, así como podemos variar los conjuntos de normas de corrección y los ejemplos de silogismos, también podemos variar las traducciones, y, como comentaremos, la

existencia de estas variaciones puede ser relevante para el modelo. Por ejemplo, en el cuadro 4 exponemos otras opciones notacionales para llevar a cabo una traducción.

Cuadro 3. Base de datos traducida

Ítem	Silogismo	Etiqueta
1.	Todo lo que calcula es algo que piensa. Toda computadora es algo que calcula. Toda computadora es algo que piensa.	Correcto
2.	Todo mamífero es algo que necesita agua para vivir. Todo mamífero es animal. Todo animal es algo que necesita agua para vivir.	Incorrecto (distributividad)
⋮	⋮	⋮
k.	Algún profesor es introvertido. Algún profesor es listo. Algún listo es introvertido.	Incorrecto (cantidad)

Cuadro 4. Otras traducciones de la base de datos

Ítem	Notación				
	Latina	Booleana	Peano-Frege	Polaca	ATL
1.	MaP	M$\not$P=0	∀x(Mx→Px)	CKΠmpΠsmΠsp	-M+P
	SaM	S$\not$M=0	∀x(Sx→Mx)		-S+M
	SaP	S$\not$P=0	∀x(Sx→Px)		-S+P
2.	MaP	M$\not$P=0	∀x(Mx→Px)	CKΠmpΠmsΠsp	-M+P
	MaS	M$\not$S=0	∀x(Mx→Sx)		-M+S
	SaP	S$\not$P=0	∀x(Sx→Px)		-S+P
⋮	⋮	⋮	⋮	⋮	⋮
k.	MiP	MP≠0	∃x(Mx^Px)	CKΣmpΣmsΣsp	+M+P
	MiS	MS≠0	∃x(Mx^Sx)		+M+S
	SiP	SP≠0	∃x(Sx^Px)		+S+P

Esta consideración notacional no es baladí: contar con varias opciones de traducción es computacional y epistemológicamente relevante porque una red neuronal es sensible a la representación de los datos, lo cual podría ser causalmente importante, como veremos más adelante.

c. Codificación

Por supuesto, aunque este subproceso de codificación puede ser innecesario y podríamos pasar del subproceso anterior (traducción) al siguiente proceso (entrenamiento), en esta presentación nos parece conveniente codificar los ítems de la base de datos en secuencias binarias. Esto puede ser engorroso, pero algo así como prestar atención a nuestro agente aprendiz: digamos que para comunicarnos con tal agente necesitamos usar su idioma, no el nuestro. En consecuencia, nuestra base de datos codificada luciría, por ejemplo, como se muestra en el cuadro 5.

Cuadro 5. Codificación de la base de datos

Ítem	Silogismo	Etiqueta
1.	001011010100...01010000	11
2.	001011010100...01010000	00
⋮	⋮	⋮
k.	001010110100...01010000	00

2. Entrenamiento

En este proceso se explotan las facultades y las habilidades de nuestro agente, a través de dos subprocesos: selección y modelado.

a. Selección

Con la información ya preparada podemos comenzar con el entrenamiento de nuestro agente. Para ello, definimos su estado inicial, asignando pesos aleatorios a las capas de sus neuronas, es decir, a su estructura interna:

```
PESOS1←matriz de |capa2|*|capa1| de números aleatorios
PESOS2←matriz de |capa3|*|capa2| de números aleatorios
```

El tamaño de la `capa1` es la longitud del código binario de un silogismo codificado, el de la `capa2` es la mitad de la longitud de la `capa1`, y el tamaño de la `capa3` es dos, ya que tenemos dos neuronas de salida (para distinguir los silogismos válidos de los inválidos).

Posteriormente, tomamos dos tercios de los *k* ejemplos de nuestra base de datos codificada:

```
silogismos←dos tercios de los k silogismos de la base de datos
                                codificada
etiquetas←dos tercios de las k etiquetas de la base de datos
codificada Con esta información podemos comenzar el proceso de
```

```
entrenamiento aplicando una función que implemente un EBP,
como la siguiente (véase el apéndice C):
[PESOS1,PESOS2]=EBP_entrenador(silogismos,etiquetas,PESOS1,
PESOS2)
```

Por supuesto, la implementación de la función que aquí sugerimos es solo una de tantas implementaciones posibles, por lo que, además de contar con cierta variabilidad en el proceso de preparación, también contamos con cierta variabilidad en la estructura interna de cada agente.

b. Modelado

Después de un tiempo, la función anterior debería regresar una primera reorganización de la estructura interna de la red mediante la definición de sus nuevos pesos. Este es un primer modelo de la red que dependerá de los ejemplos, la traducción y la estructura interna propia del agente.

3. Validación

En este proceso valoramos si los modelos que el agente ha producido son apropiados, mediante dos subprocesos: evaluación y comparación.

a. Evaluación

Una vez que contamos con un modelo de la red podemos evaluar si nuestro agente está listo. Para ello, definimos un nuevo silogismo a clasificar, usando algún ejemplo extraído de la tercera parte de los k ejemplos de nuestra base de datos codificada que no usamos para entrenar al agente. Llamemos a este ejemplo `silogismo_a_clasificar`. Así, podemos aplicar una función como la siguiente (véase el apéndice C):

```
[Respuesta]=EBP_clasificador(PESOS1,PESOS2,silogismo_a_clasificar)
```

Esta función recibe el modelo generado (i. e. `PESOS1` y `PESOS2`), el nuevo `silogismo_a_clasificar` y produce una respuesta: o bien el silogismo es correcto, o bien es incorrecto, o bien el resultado es indefinido (i. e., el agente *no sabe* cuál es la respuesta). Ahora, como este modelo de aprendizaje estará bajo nuestra supervisión, tenemos que emplear una función como la siguiente (véase el apéndice C):

```
[silogismos,etiquetas]=EBP_supervisión(silogismo_a_clasificar)
```

Esta última función nos pregunta si la respuesta del agente es correcta, es decir, si el modelo que ha aprendido le permite clasificar cualquier silogismo. Si la respuesta del agente es correcta, le extendemos una felicitación; de otro modo, lo corregimos indicándole cuál es la respuesta correcta para que se tome un tiempo y vuelva a producir un nuevo modelo: este subproceso, como puede notarse, da cuenta de la primera fase de retroalimentación.

b. Comparación

Finalmente, este subproceso es el que nos sirve para iniciar la segunda fase de retroalimentación, que ya no es para el agente, sino para nosotras y nosotros como docentes. Para ello, podemos hacer eco de la variedad de reglas, ejemplos, traducciones y redes, ya que la existencia de estas variaciones propicia un ambiente generoso para hacer cambios, ajustes y adaptaciones para facilitar, a su vez, comparaciones en términos, por ejemplo, de eficiencia, de complejidad, de efectividad, etc. Con el fin de ilustrar este punto, en la siguiente sección hacemos un breve recuento cualitativo de un par de experiencias —con las redes Boecio y Diotima— usando este modelo.

3. Reporte de experiencias

En este apartado usaremos el esquema de comprensión ordenada del lenguaje (COL) descrito por Campirán (2005), el cual nos permite reportar información a través de la respuesta a las preguntas *¿Qué pasó?, ¿Qué aprendí?* y *¿Qué sentí?;* pero antes, expondremos algunos detalles específicos de cada entrenamiento, comparando sus resultados con Occam, una red testigo que usamos para comparar los entrenamientos de Boecio y Diotima, mediante la discriminación de una base de 10 silogismos inválidos (cuadro 6).

Cuadro 6. Comparación

Características	Boecio	Diotima	Occam
Notación	Latina	ex professo	ATL
Función de activación	Sigmoide	Sigmoide	Sigmoide
Capas	3	3	3
Número de ejemplos	281	247	489
Número de ejemplos válidos	57	174	185
Número de ejemplos inválidos	224	73	304
Tiempo de entrenamiento (horas)	2	4	7
Porcentaje de efectividad	0 %	10 %	60 %

3.1 Boecio

¿Qué pasó? Para preparar a Boecio creamos 150 silogismos. Para hacerlo visitamos diferentes sitios de internet que ofrecían ejemplos válidos e inválidos. Cuando los ejemplos empezaron a ser insuficientes, optamos por realizar los propios. Durante este proceso generamos más silogismos tipo *Barbara, Celarent y Darii que*, por ejemplo, *Barbari, Celaront* o *Cesaro*, es decir, preparamos más formas válidas que inválidas. Y para la representación decidimos usar la notación latina.

Ahora bien, aunque Boecio comenzó con 150 silogismos, aprendió un total de 380 ejemplos y fue entrenado principalmente con formas inválidas tipo *iaa*-2. Con todo, el sistema no logró aprender que esta estructura era inválida y terminó por convertirse en una red relativista, fanática de la validez.

¿Qué aprendí? Creemos que lo anterior pudo ocurrir por la sobrecarga de silogismos válidos entre los 150 originalmente planteados: el sobreentrenamieno podría explicar la baja eficacia de Boecio. Adicionalmente, fue extraordinario observar cómolas redes neuronales pueden adaptarse para estudiar diferentes temas de interés; sin embargo, lo más interesante de esta actividad es que nos permitió comprender cuán pacientes tenemos que ser al enseñar lógica a un agente artificial, ya no digamos a una persona.

¿Qué sentí? Esta actividad nos ha permitido vivir, en carne propia, algunas de las dificultades y retos que las y los docentes enfrentan todos los días al enseñar lógica.

3.2 Diotima

¿Qué pasó? Para preparar a Diotima creamos 150 silogismos. Algunos se extrajeron de sitios de internet, pero la mayoría fueron inventados. Siguiendo la estructura de la notación latina, inventamos una propuesta *ex professo* en la que se marca el término medio con una X en una posición especial. Por ejemplo, un silogismo tipo *Barbara* luce así: XaaXa. Esta notación es breve y, aunque muestra la posición del término medio y el modo de un silogismo, ignora la posición del sujeto y el predicado.

¿Qué aprendí? Aunque existen notaciones ya establecidas, aprendimos que es posible crear y sugerir notaciones nuevas y alternativas. En este caso, nuestra nueva notación podría explicar la baja efectividad de Diotima..

¿Qué sentí? Al principio sentimos frustración, especialmente cuando notábamos que Diotima era incapaz de distinguir silogismos, pero después empezamos a sentir cierta empatía con su lento proceso de aprendizaje, pues si una red específicamente diseñada para aprender silogismos tiene sus problemas, no deberíamos ser tan duras con nosotras mismas cuando tenemos dificultades.

4. Conclusiones

Hemos expuesto en este trabajo una propuesta didáctica inspirada por los mecanismos y los resultados del método de aprendizaje a través de la enseñanza. Se trata de un modelo didáctico para aprender lógica enseñando lógica; en particular, para aprender silogística entrenando a una red neuronal artificial.

El modelo que sugerimos es un proceso de retroalimentación que se desarrolla en tres momentos seriados que incluyen la preparación de un curso, la implementación de dicho curso y su evaluación. Sin embargo, lo que vale la pena notar aquí es que no estamos proponiendo otro demostrador de teoremas ni otro modelo formal de aprendizaje; lo que buscamos es rastrear un modelo didáctico para aprender silogística usando herramientas computacionales. Lo que importa, pues, no es tanto la herramienta, sino lo que quienes utilizan este tipo de herramientas pueden aprender implementando los procesos y los pasos de este modelo.

En efecto, el reporte cualitativo de experiencias (en este caso un par de estudiantes de filosofía) sugiere que las y los usuarios comprenden, por ejemplo, *a*) que planificar un curso no es una tarea trivial, puesto que hay que preparar, con cierta dedicación, los materiales adecuados; *b*) que el aprendizaje de un agente es un proceso complejo, puesto que la variedad de reglas, ejemplos y traducciones propicia un ambiente generoso para hacer cambios, ajustes y adaptaciones; *c*) que el proceso de enseñanza no solo debería considerar las variaciones en nuestros procesos de preparación, sino las variaciones individuales de los agentes, puesto que las variaciones en nuestras redes promueven diferentes tasas de aprendizaje; *d*) que, en el ínterin, se aprende lógica (silogística en este caso) al preparar ejemplos, discriminar argumentos, inventar notaciones, comparar evaluaciones y cuestionar los propios procesos. Y sobre todo, se aprende lógica mediante procesos que permiten reconocer algunas de las dificultades a las que se enfrentan las y los docentes de lógica cuando se preparan para ayudar a que otras personas aprendan a razonar.

Referencias

Bargh, J. A. y Schul, Y. (1980). On the cognitive benefits of teaching. *Journal of Educational Psychology*, *72*(5), 593–604. https://doi.org/10.1037/0022-0663.72.5.593

Brooks, R. A. (1999). *Cambrian intelligence: the early history of the new AI*. MIT Press.

Campirán, A. (2005). Autobservación y metacognición. *Ergo. Colección Temas Selectos,* (1), 91-106.

Cohen, P. A., Kulik, J. A. y Kulik, C. C. (1982). Educational outcomes of tutoring: A meta-analysis of findings. *American Educational Research Journal, 19*(2), 237-248. https://doi.org/10.3102/00028312019002237

Gallier, J. H. (2015). *Logic for computer science: foundations of automatic theorem proving* (2.a ed.). Courier Dover.

Haugeland, J. (1989). *Artificial intelligence: the very idea*. MIT Press.

Huth, M. y Ryan, M. (2004). *Logic in computer science: modelling and reasoning about systems*. Cambridge University Press.

Koh, A. W. L, Lee, S.C. y Lim, S. W. H (2018). The learning benefits of teaching: A retrieval practice hypothesis. *Applied Cognitive Psychology* 32, 401– 410. https://doi.org/10.1002/acp.3410

Martin, J. y Kelchner, R. (1998). *Lernen durch lehren*. http://www.lernen-durch-lehren.de/Material/Publikationen/timm.pdf

McCulloch, W. y Pitts, W. (1943). A logical calculus of the ideas immanent in nervous activity. *Bulletin of Mathematical Biophysics*, *5*, 115-133. https://doi.org/10.1007/BF02478259

Minsky M. L. y Papert S. A. (1969). *Perceptrons*. MIT Press.

Moss, L. (2015). Natural Logic. En S. Lappin & C. Fox (Eds.), *The handbook of contemporary semantic theory*. John Wiley & Sons.

Roscoe, R. D. y Chi, M. T. H. (2008). Tutor learning: the role of explaining and responding to questions. *Instructional Science*, *36*(4), 321-350.

Rosenblatt, F. (1957). *The Perceptron —a perceiving and recognizing automaton* (Report 85-460-1). Cornell Aeronautical Laboratory.

Rumelhart, D. E., Hinton, G. E. y Williams, R. J. (1986). Learning representations by back-propagating errors. *Nature, 323*(6088), 533-536.

Russell, S. y Norvig, P. (1995). Artificial Intelligence. A modern approach. Nueva Jersey: Prentice Hall.

Safiye, A. (2015). Is learning by teaching effective in gaining 21st century skills? The views of pre-service science teachers. *Educational Sciences: Theory & Practice*, *15*(6), 1441-1457. https://doi.org/10.12738/estp.2016.1.0019

Sommers, F. (1984). *The logic of natural language*. Oxford University Press.

Apéndice A. Silogística

La silogística es una lógica de términos que estudia la relación de inferencia entre proposiciones categóricas. Una proposición categórica es una proposición compuesta por dos términos, una cantidad y una cualidad. El sujeto y el predicado de la proposición se llaman términos: el términoesquema S denota el término sujeto de la proposición y el términoesquema *P* denota el predicado. La cantidad puede ser universal (*Todo*) o particular (*Algún*) y la cualidad puede ser afirmativa (*es*) o negativa (*no es*).

Las proposiciones categóricas se denotan mediante una etiqueta —a, para la universal afirmativa (*SaP*); *e*, para la universal negativa (*SeP*); *i*, para la particular afirmativa (*SiP*); *o*, para la particular negativa (*SoP*)— que nos permite determinar una secuencia de tres proposiciones que se conoce como modo. Un silogismo categórico es, entonces, un modo ordenado de tal manera que dos proposiciones fungen como premisas y la última como conclusión.

En el interior de las premisas existe un término que ocurre en ambas premisas, pero no en la conclusión. Este término especial, usualmente denotado con el término-esquema M, funciona como un enlace entre los términos restantes, y se conoce como término medio. De acuerdo con la posición del término medio se pueden definir cuatro arreglos o figuras (cuadro A1)que codifican los modos o patrones silogísticos válidos (por mor de la brevedad, pero sin perder generalidad, omitimos los silogismos que requieren carga existencial).

Cuadro A1. Inferencias mediatas válidas

Primera figura	Segunda figura	Tercera figura	Cuarta figura
aaa	*eae*	*iai*	*aee*
eae	*aee*	*aii*	*iai*
aii	*eio*	*oao*	*eio*
eio	*aoo*	*eio*	

Apéndice B. Retropropagación del error

El algoritmo EBP busca minimizar una función de error en un espacio de pesos usando el método de gradiente descendente. El arreglo de pesos que minimiza la función de error es considerado como la solución óptima para resolver un problema del aprendizaje.

Dado que este método requiere calcular el gradiente del error, es necesario garantizar que la función de error sea continua y diferenciable en todo el dominio de la función a considerar. En consecuencia, la función de activación del perceptrón resulta inútil, puesto que la composición de funciones de perceptrones resulta ser discontinua. La función de activación más popular para resolver este problema es la función sigmoide:

$$s(x)=\frac{1}{1+e^{-cx}}$$

donde c es una constante de gradación. La forma de la sigmoide varía de acuerdo con c y, en particular, cuando $c \to \infty$, converge en la función del perceptrón simple, además de contar con la ventaja de tener una (hermosa) derivada con respecto a x con $c = 1$:

$$\frac{d}{dx}s(x)=\frac{e^{-x}}{(1+e^{-x})^2}=s(x)(1-s(x))$$

Con esto resulta más claro el funcionamiento del EBP. Comenzamos con una capa de entradas que se mapean en, por lo menos, una capa oculta de neuronas, que a su vez se dirigen a una capa de neuronas de salida, formando así una red de mayor complejidad.

Una vez que la información llega a la capa de salida, se evalúa si la salida es la deseada, pues el aprendizaje se entiende como el mínimo error entre las entradas y las salidas; mientras este error no sea el mínimo, el aprendizaje se convierte en una actualización constante de los pesos de la red mediante la regla delta, la cual es, precisamente, el cálculo del gradiente.

La regla delta se calcula para cada neurona de cada capa de la red. Para la capa de salida se calcula así:

$$\delta_k = s_k(1-s_k)(t_k-s_k)$$

donde k es el total de miembros de la capa de salida, s es el resultado de la función de activación sigmoide de dicho miembro y t el resultado esperado.

Para las capas intermedias se calcula así:

$$\delta_j = s_j(1 - s_j)\sum w_{kj}\delta_k$$

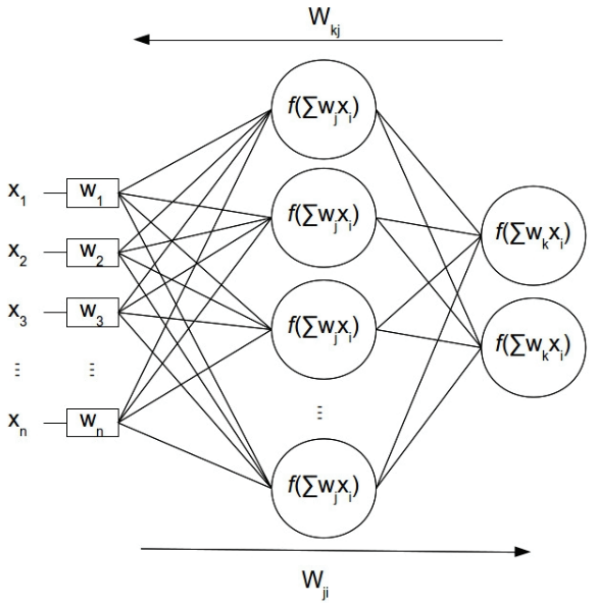
Donde j es el conjunto de miembros de la capa intermedia, k es el número de miembros de la capa posterior, s es lo obtenido de la función de activación, W_{kj} es el peso que va del miembro de la capa actual al de la posterior y δ_j es el resultado de la regla delta de ese miembro de la capa posterior.

Por último, los pesos de cada neurona se actualizan con

$$w_{ji} = w_{ji} + \alpha\delta x_{ji}$$

donde α es un factor de aprendizaje. Este proceso continúa hasta que el error es minimizado o hasta que se cumpla una condición de paro. De esta forma, a partir de una salida deseada, es posible entrenar una red de neuronas artificiales para que sea capaz de aprender a identificar patrones de entrada (figura B1).

Figura B1. Una interpretación visual del EBP



Como podemos apreciar, la idea general del algoritmo EBP es que la conducta de aprendizaje se puede modelar/interpretar/implementar como un problema de optimización de funciones que se resuelve mediante una combinación de neuronas artificiales (perceptrones) conectadas entre sí en una red que permite la retroalimentación entre ellas.

Apéndice C. Pseudocódigo

```
[PESOS1,PESOS2]←EBP_entrenador(silogismos,etiquetas,PESOS1,PESOS2)
\\Especificación de la estructura: las capas
capa1←tamaño del código de cada silogismo
capa2←mitad de capa1
capa3←2
\\Módulo de entrenamiento
alpha←0.1
Mientras error_total es grande
  error_total←0
  Para i=1 hasta número_de_silogismos hacer
    Para j=1 hasta capa2 hacer
      suma←0
      Para k=1 hasta capa1 hacer
        suma←suma+(silogismos(i,k)*PESOS1(j,k))
      FinPara
      entrada(j)←1/1+e-suma
    FinPara
    Para m=1 hasta capa3 hacer
      suma←0
      Para j=1 hasta capa2 hacer
        suma←suma+(entrada(j)*PESOS2(m,j))
      FinPara
      salida(m)←1/1+e-suma
    FinPara
      Para m=1 hasta capa3 hacer
      delta_1(m)←(etiqueta(i,m)-salida(m))*(salida(m)*(1-salida(m)))
      error_total←error_total+delta_1(m)
    FinPara
    Para j=1 hasta capa2 hacer
      error_de_entrada←0
      Para m=1 hasta capa3 hacer
        error_de_entrada←error_de_entrada+(delta_1(m)*PESOS2(m,j))
      FinPara
      delta_2(j)←(entrada(j)*(1-entrada(j)))*error_de_entrada
    FinPara
    Para m=1 hasta capa3 hacer
      Para j=1 hasta capa2 hacer
        PESOS2(m,j)←PESOS2(m,j)+(alpha*delta_1(m)*entrada(j))
      FinPara
    FinPara
    Para j=1 hasta capa2 hacer
      Para k=1 hasta capa1 hacer
        PESOS1(j,k)←PESOS1(j,k)+(alpha*delta_2(j)*silogismos(i,k))
      FinPara
```

```
    FinPara
  FinPara
error_total←|error_total|
alpha←alpha++
FinMientras

[Respuesta]←EBP_clasificador(PESOS1,PESOS2,silogismo_a_clasificar)
Para j=1 hasta capa2 hacer
  suma←0
  Para k=1 hasta capa1 hacer
    suma←suma+(silogismo_a_clasificar(1,k)*PESOS1(j,k))
  FinPara
  entrada(j)=1/1+e-suma
FinPara
Para k=1 hasta capa3 hacer
  suma←0
  Para j=1 hasta capa2 hacer
    suma←suma+(entrada(j)*PESOS2(k,j))
  FinPara
  salida(k)=1/1+e-suma
FinPara
Si salida=[1,1] entonces
  Respuesta←"Correcto"
Si salida=[0,0] entonces
  Respuesta←"Incorrecto"
De otro modo
  Respuesta←"Indefinido"
FinSi

[silogismos,etiquetas]←EBP_supervisión(silogismo_a_clasificar)
Si la respuesta es correcta entonces
  silogismos←silogismos
  etiquetas←etiquetas
De otro modo
  silogismos←silogismos+silogismo_a_clasificar
  etiquetas←etiquetas+nueva_etiqueta
  [PESOS1,PESOS2]←EBP_entrenador(silogismos,etiquetas,PESOS1,PESOS2)
FinSi
```