Facultad de Matemática,
Astronomía, Física y
Computación

Universidad
Nacional
de Córdoba

Aprendizaje Automático para la Selección de Grupos de Control en Evaluación de Impacto [1]

Autor: Benjamín Bas Peralta.
Directores: Dr. Martín A. Domínguez, Mgter. David Giuliodori.
Facultad de Matemática, Astronomía, Física y Computación
Universidad Nacional de Córdoba

23 de marzo de 2025

Resumen

Hello, here is some text without a meaning...

Palabras Clave: Análisis de Redes Sociales, Aprendizaje Automático, Detección de influenciadores, Detección de comunidades, Modelos de predicción, Twitter

Agradecimientos

– TO DO –

Índice general

Capítulo 1

Introducción

Capítulo 2

Marco Teórico

2.1. Evaluación de Impacto

2.1.1. ¿Qué es una Evaluación de Impacto?

Los programas y políticas de desarrollo suelen estar diseñados para cambiar resultados, como aumentar los ingresos, mejorar el aprendizaje o reducir las enfermedades [5]. Saber si estos cambios se logran o no es una pregunta crucial para las políticas públicas [5], y el método para responderla es lo que se conoce como **evaluación de impacto**.

Una evaluación de impacto mide los cambios en el bienestar de los individuos que se pueden atribuir a un proyecto, un programa o una política específicos [5]. Su sello distintivo radica en que pueden proporcionar **evidencia** robusta y creíble sobre si un programa concreto ha alcanzado o está alcanzando sus objetivos [5].

Estas evaluaciones ponen un fuerte énfasis en los resultados y buscan responder una pregunta específica de causa y efecto: ¿cuál es el impacto (o efecto causal) de un programa? Más precisamente, intentan identificar y cuantificar los cambios directamente atribuibles al tratamiento [5] sobre un conjunto de **variables de resultado** o **de interés** en un conjunto de individuos [1]. Estas variables son aquellas sobre las cuales se espera que el programa tenga un efecto en los beneficiarios [1]. Podrían ser por ejemplo la estatura y peso de los individuos, la cantidad de empleados o de ventas en una empresa, o indicadores de salud de un paciente.

Por lo tanto, el objetivo final de la evaluación de impacto consiste en establecer lo que se conoce como **efecto del tratamiento**, que es la diferencia entre la variable de resultado del individuo participante en el programa en presencia del programa, y la variable de resultado de ese individuo en ausencia del programa [1]. Sin embargo, es evidente que la respuesta a qué habría pasado con los beneficiarios si no hubieran recibido el tratamiento se refiere a una situación que no es observable. Este resultado hipotético se denomina **contrafactual** y es lo que se debe estimar en cualquier evaluación de impacto.

Algunas de las principales razones por las que se debería promover el uso de estas

evaluaciones como herramientas de gestión provienen del hecho que permiten mejorar la rendición de cuentas, la inversión de recursos públicos o la efectividad de una política, obtener financiamiento, como así también probar modalidades de programas alternativos o innovaciones de diseño [5] y revelar la realidad de muchas políticas públicas para de esta forma contribuir a la fiscalización mediática [1].

Un ejemplo claro de por qué son necesarias las evaluaciones de impacto es el que describe Howard White con respecto al Programa Integrado de Nutrición en Bangladesh (PINB) [16]. Este programa identificaba, mediante mediciones en campo, a los niños desnutridos y los asignaba a un tratamiento que incluía alimentación suplementaria a los menores y educación nutricional a las madres [1]. Inicialmente, el programa fue considerado como un éxito ya que los datos de monitoreo mostraban caídas importantes en los niveles de desnutrición. El Banco Mundial decidió, con base en esta evidencia y previo a cualquier tipo de evaluación, aumentar los recursos destinados al programa. Sin embargo, las primeras evaluaciones de impacto, realizadas por el Grupo Independiente de Evaluación del mismo Banco Mundial y por la ONG inglesa *Save the Children*, mostraron que la mejoría de los indicadores de los beneficiarios era similar o inferior a la de otros niños con características comparables que no hacían parte del programa [1]. Estos resultados reflejaron que las percepciones de los administradores del programa y de las entidades financiadoras eran erradas, y sugirieron algunos correctivos al programa [1].

2.1.2. Estimación del Tratamiento

El marco teórico estándar para formalizar el problema de la evaluación de impacto se basa en el **modelo de resultados potenciales** o **modelo causal de Rubin** [13].

Formalmente, se definen dos elementos para cada individuo $i = 1, ..., N$, donde N denota la población total:

- Por un lado, el indicador de tratamiento D_i, tal que $D_i = 1$ implica que el individuo i participó del tratamiento, y $D_i = 0$ en caso contrario.
- Por otro lado, las variables de resultado las definimos como $Y_i(D_i) = Y_i|D_i$ - se lee como "el valor de Y_i *dado* D_i". De esta forma, $Y_i(1)$ es la variable de resultado si el individuo i es tratado, e $Y_i(0)$ es la variable de resultado si el individuo i no es tratado. Estos valores son los resultados potenciales.

Con esto, el **efecto del tratamiento** para un individuo i se puede escribir como:

$$\tau_i = Y_i(1) - Y_i(0) = (Y_i|D_i = 1) - (Y_i|D_i = 0) \tag{2.1}$$

Según esta fórmula, el impacto causal (τ) de un programa (D) en una variable de resultado (Y) para un individuo i es la diferencia entre la variable con el programa ($Y_i(1)$) y la misma variable sin el programa ($Y_i(0)$).

De nuevo, el problema fundamental de la evaluación de impacto es que se intenta medir una variable en un mismo momento del tiempo para la misma unidad de observación pero en dos realidades diferentes. Sin embargo, claramente solo se da uno de los dos resultados potenciales $Y_i(1)$ o $Y_i(0)$, pero no ambos. Es decir, en los datos queda solamente registrado $Y_i(1)$ si $D_i = 1$ e $Y_i(0)$ si $D_i = 0$; no se dispone de $Y_i(1)$ si el individuo no fue tratado ($D_i = 0$), ni tampoco de $Y_i(0)$ si el individuo fue tratado ($D_i = 1$). De esta manera, el **resultado observado de** Y_i se puede expresar como:

$$Y_i = D_i Y_i(1) + (1 - D_i) Y_i(0) = \begin{cases} Y_i(1) \text{ si } D_i = 1 \\ Y_i(0) \text{ si } D_i = 0 \end{cases} \tag{2.2}$$

Si nos concentramos en las unidades tratadas, en la Ecuación 2.1, el término $Y_i(0) = (Y_i|D_i = 0)$ representa la situación contrafactual, es decir *cuál habría sido el resultado si la unidad no hubiera participado en el programa.* Al ser imposible observar directamente el contrafactual, es necesario **estimarlo**. La forma más directa de solucionar este problema sería hallando un "clon perfecto" para cada uno de los individuos participantes del programa pero que no haya participado, lo cual resulta bastante difícil. Por lo tanto, el primer paso para lograr esta estimación consiste en **desplazarse desde el nivel individual al nivel del grupo** [5], concentrando el análisis en el **impacto** o **efecto *promedio*** (y no individual).

En primera instancia, se puede estimar el **impacto promedio del programa *sobre la población*** (o ATE, del inglés *Average Treatment Effect*):

$$ATE = \mathbb{E}\left[Y_i(1) - Y_i(0)\right] \tag{2.3}$$

donde el operador $\mathbb{E}$ representa la media de una variable aleatoria.

El ATE se interpreta como el cambio promedio en la variable de resultado cuando un individuo escogido al azar pasa aleatoriamente de ser participante a ser no participante [1], es decir representa el efecto esperado del tratamiento si toda la población fuera tratada. Esta medida es relevante en el caso de la evaluación de un programa universal. Es importante notar que para calcularlo, habría que estimar un contrafactual tanto para los tratados como para los no tratados.

Sin embargo, en la mayoría de los casos, el tratamiento solo está disponible para un subconjunto de la población, ya sea por restricciones presupuestarias o criterios de elegibilidad, entre otras razones. Además, también ocurre que no todos los que fueron seleccionados para recibirlo efectivamente participan o se inscriben. En estos casos, suele ser más relevante estimar el efecto únicamente en quienes efectivamente recibieron el tratamiento.

Por lo tanto, se define el **impacto promedio del programa *sobre los tratados*** (o ATT, del inglés *Average Treatment Effect on the Treated*), que es en general el parámetro de mayor interés en una evaluación de impacto, y representa el efecto esperado del

tratamiento en el subconjunto de individuos que fueron efectivamente tratados:

$$ATT = \mathbb{E}\left[Y_i(1) - Y_i(0)|D_i = 1\right] = \mathbb{E}\left[Y_i(1)|D_i = 1\right] - \mathbb{E}\left[Y_i(0)|D_i = 1\right] \tag{2.4}$$

Es decir, el ATT es la diferencia entre la media de la variable de resultado en el grupo de los participantes y la media que hubieran obtenido los participantes si el programa no hubiera existido [1].

2.1.3. El Grupo de Control como Estimador del Contrafactual

En la ecuación 2.4, el término $\mathbb{E}\left[Y_i(1)|D_i = 1\right]$ es un resultado observable mientras que $\mathbb{E}\left[Y_i(0)|D_i = 1\right]$ es el promedio contrafactual que debemos aproximar. Para lograrlo, se construye lo que se conoce como **grupo de control** o **de comparación**, formado por individuos que no participan del programa pero que, idealmente, son estadísticamente similares [5] al **grupo de tratamiento**, compuesto por quienes sí recibieron el programa. La Figura 2.1 ilustra esta distinción. A partir de ahora, denotaremos con C_i a un individuo i que pertenezca al grupo de control.

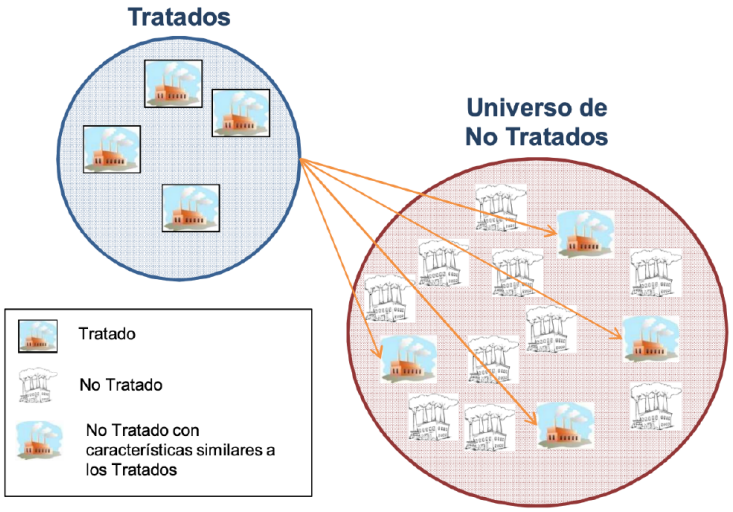

Figura 2.1: El grupo de control debería estar formado idealmente por individuos que no recibieron el tratamiento pero que en promedio poseen características similares a los tratados.

Por lo tanto, en la práctica, el reto está en definir un *buen* grupo de control, es decir uno que sea estadísticamente idéntico al de tratamiento, en promedio, en ausencia del programa [5]. Si se lograra que los dos grupos sean idénticos, con la única excepción que un grupo participa del programa y el otro no, entonces sería posible estar seguros que cualquier diferencia en los resultados tiene que deberse al programa [5].

Para que un grupo de comparación sea válido, debe satisfacer lo siguiente [5]:

- Las características *promedio* del grupo de tratamiento y del grupo de comparación deben ser idénticas en ausencia del programa. Cabe resaltar que no es necesario que las unidades individuales en el grupo de tratamiento tengan clones perfectos en el grupo de control.
- No debe ser afectado por el tratamiento de forma directa ni indirecta.
- El grupo de comparación debería reaccionar de la misma manera que el grupo de tratamiento si fuera objeto del programa.

Cuando el grupo de comparación no produce una estimación precisa del contrafactual, no se puede establecer el verdadero impacto del programa. A continuación, se presentan dos situaciones en las que esto ocurre.

Comparaciones antes-después

Este tipo de comparaciones intenta establecer el impacto de un programa a partir de un seguimiento de los cambios en los resultados en los participantes del programa a lo largo del tiempo. Consideran el contrafactual como el resultado para el grupo de tratamiento antes que comience la intervención, momento también conocido como **línea de base**. Esta comparación supone que si el programa no hubiera existido, el resultado para los participantes del programa habría sido igual a su situación antes del programa, lo cual en la mayoría de los tratamientos este supuesto no puede sostenerse [5].

Comparaciones de inscritos y no inscritos: Sesgo de autoselección

Como explicamos anteriormente, el término $\mathbb{E}\left[Y_i(0)|D_i = 1\right]$ de la Ecuación 2.4 representa el contrafactual, que no es observable. Sin embargo, lo que sí se puede medir es la variable de resultado entre los no inscritos, aquellos individuos que no participaron del programa, es decir $\mathbb{E}\left[Y_i(0)|D_i = 0\right]$.

Por lo tanto, podríamos tomar a todo el conjunto de los no participantes como grupo de control y solucionar el problema del contrafactual utilizando $\mathbb{E}\left[Y_i(0)|D_i = 0\right]$ como estimador, es decir podríamos asumir lo siguiente:

$$\mathbb{E}\left[Y_i(0)|D_i = 1\right] = \mathbb{E}\left[Y_i(0)|D_i = 0\right] \tag{2.5}$$

Esto quiere decir que el valor esperado de la variable de resultado en ausencia del programa ($\mathbb{E}\left[Y_i(0)\right]$) es idéntico para el grupo de tratados ($D = 1$) y para el de no tratados ($D = 0$). Indica que los individuos que participan en el programa no son sistemáticamente distintos de los que no lo hacen [1] o, en otras palabras, que la decisión de participar en el programa no está determinada por factores que también influyen en la variable de resultado.

De esta forma, podríamos calcular el ATT como:

$$ATT = \mathbb{E}\left[Y_i(1)|D_i = 1\right] - \mathbb{E}\left[Y_i(0)|D_i = 0\right] \tag{2.6}$$

Sin embargo, el supuesto de la Ecuación 2.5 se viola toda vez que la participación en el programa es una *elección* del individuo elegible [1]. La razón es que los participantes y los no participantes generalmente son diferentes, aún en ausencia del programa, y son justamente esas diferencias que llevan a que algunos individuos escojan participar y otros no. Por lo tanto, si en estos casos se toma como grupo de control a todos aquellos que no decidieron participar y se mide el impacto con respecto a ellos, puede ocurrir que la diferencia en los resultados se deba a características propias de los individuos que llevaron a unos a anotarse y a otros no.

Este problema se conoce como **sesgo de autoselección**, ya que los integrantes del grupo de control se *autoseleccionaron* para no participar del programa. Más concretamente, este sesgo se produce cuando los motivos por los que un individuo participa en un programa, usualmente no observables y difíciles de medir, están correlacionados con los resultados [5], y por ende, es muy probable que la variable de resultado del grupo de tratamiento y del grupo de control sean diferentes, *aún si el programa no existiera* [1]. Intuitivamente, si hay variables que explican tanto la participación como la variable de resultado, la comparación de medias puede estar atribuyendo al programa un efecto que en realidad se debe a las diferencias preexistentes entre el grupo de tratamiento y el grupo de control [1].

Un buen ejemplo de esta situación lo presenta [1], en el que se plantea un programa de nutrición infantil. Podría ocurrir que las madres de familias participantes del programa sean más proactivas respecto al desarrollo de sus hijos, por lo cual se preocuparon en lograr la participación en el programa. El problema de autoselección en este caso radica en que la motivación de las madres (que no se observa y es difícil de medir) afecta no solo la probabilidad de participar en el programa, sino también el estado nutricional de los niños ya que estas podrían vigilar mejor la dieta de sus hijos. De esta forma, la diferencia observada en el estado nutricional de los niños de los dos grupos podría deberse parcialmente a la diferencia en el nivel de compromiso de las madres, y no exclusivamente a que un grupo participa en el programa y el otro no.

Podemos plantear este problema más formalmente. Notemos que podemos reescribir la fórmula del ATT (2.4) de la siguiente manera:

$$ATT + \mathbb{E}\left[Y_i(0)|D_i = 1\right] = \mathbb{E}\left[Y_i(1)|D_i = 1\right] \tag{2.7}$$

Si ahora restamos $\mathbb{E}\left[Y_i(0)|D_i = 0\right]$ a ambos lados, obtenemos:

$$\mathbb{E}\left[Y_i(1)|D_i = 1\right] - \mathbb{E}\left[Y_i(0)|D_i = 0\right] \; = ATT + \mathbb{E}\left[Y_i(0)|D_i = 1\right] - \mathbb{E}\left[Y_i(0)|D_i = 0\right] \tag{2.8}$$

De esta forma, el sesgo de autoselección aparece en el término $\mathbb{E}\left[Y_i(0)|D_i = 1\right] - \mathbb{E}\left[Y_i(0)|D_i = 0\right]$.

Dicho esto, el gran desafío de la evaluación de impacto es determinar las condiciones o supuestos bajo los cuales $\mathbb{E}[Y_i(0)|D_i = 0]$ se puede utilizar como una estimación válida del contrafactual $\mathbb{E}[Y_i(0)|D_i = 1]$ [1], y por lo tanto utilizarse para poder aproximar el valor del ATT. Asegurarse de que el impacto estimado esté libre de sesgo de autoselección es uno de los principales objetivos de cualquier evaluación de impacto, y plantea importantes dificultades [5].

Las reglas de un programa para seleccionar a los participantes, también llamadas **reglas de asignación**, constituyen el parámetro clave para determinar el método de evaluación de impacto. A continuación, presentamos las distintas reglas y los métodos de evaluación que se utilizan en la práctica al trabajar con cada una de ellas.

2.1.4. Evaluaciones Experimentales o Aleatorias

En este tipo de evaluaciones, los beneficiarios del programa en cuestión son seleccionados al azar, es decir mediante un sorteo. La asignación aleatoria se considera la regla de oro de la evaluación de impacto ya que no solo proporciona a los administradores del programa una regla imparcial y transparente para asignar recursos escasos entre poblaciones igualmente merecedoras de ellos, sino que también representa el método más sólido para evaluar el impacto de un programa [5].

La asignación aletoria trae dos consecuencias importantes: las unidades ya no pueden *elegir* si participar o no, y además todas tienen la misma probabilidad de ser seleccionadas para el programa. De esta forma, el resultado potencial de cada individuo es independiente de sus condiciones previas, ya que la asignación al tratamiento no está determinada por sus características, sino por el azar.

Como explicamos anteriormente, el grupo de comparación ideal sería lo más similar posible al grupo de tratamiento en todos los sentidos, excepto con respecto a su participación en el programa que se evalúa. Ahora bien, el proceso de asignación aleatoria producirá dos grupos que tienen una alta probabilidad de ser estadísticamente idénticos en todas sus características (observables y no observables), siempre que el número de unidades sea suficientemente grande [5]. Esto es así ya que en general, si la población de unidades elegibles es lo suficientemente grande, este mecanismo asegura que cualquier rasgo de la población se transfiere a ambos grupos. Sin embargo, en la práctica, este supuesto debería comprobarse empíricamente con los datos de línea de base de ambos grupos.

Dicho esto, lo que ocurre en estos casos es que todos aquellos individuos que no resultaron ser beneficiarios del programa conforman un grupo de control que resulta ser naturalmente un excelente estimador del contrafactual $\mathbb{E}[Y_i(0)|D_i = 1]$. Es decir, mediante este tipo de asignación, se asegura que se cumple el supuesto de la Ecuación 2.5.

La asignación aleatoria puede utilizarse como regla de asignación de un programa en dos escenarios específicos: cuando la población elegible es mayor que el número de pla-

zas disponibles del programa, o cuando sea necesario ampliar un programa de manera progresiva hasta que cubra a toda la población elegible [5]. Además, la asignación aleatoria podría hacerse a nivel individual (por ejemplo, a nivel de hogares), o a nivel de conglomerado (por ejemplo, comunidades) [1].

Una vez que se haya asignado el tratamiento de manera aleatoria, gracias a **??**, es bastante sencillo estimar el impacto del programa. Después de que el programa ha funcionado durante un tiempo, tendrán que medirse los resultados de las unidades de tratamiento y de comparación. El impacto del programa es sencillamente la diferencia entre el resultado promedio para el grupo de tratamiento y el resultado promedio para el grupo de comparación, es decir:

$$\mathbb{E}\left[Y_i(1)|D_i = 1\right] - \mathbb{E}\left[Y_i(0)|D_i = 0\right]$$

Resulta claro que las evaluaciones experimentales tienen varias ventajas: por diseño, resuelven el problema del sesgo de selección, y además los resultados obtenidos son transparentes, intuitivos y no es necesario utilizar herramientas econométricas sofisticadas para alcanzarlos [1]. Esto hace que contribuyan a la transparencia de las políticas públicas y a la rendición de cuentas [1].

Sin embargo, este tipo de evaluaciones sufre varias limitaciones que hace que no sea tan común en la práctica. Por un lado, pueden ser costosas y de difícil implementación [1]. Esto se debe a que en general, el experimento aleatorio se hace específicamente para evaluar una intervención, por lo que es necesario destinar recursos para la prueba piloto, la recolección de información, los seguimientos, y a veces incluso la implementación del programa [1]. Otro problema que tienen, y probablemente el más importante, es que por pertenecer al grupo de control, se *excluye* a un segmento de la población, igualmente vulnerable y elegible, de los beneficios de la intervención [1]. Además, dada la imposibilidad de negar los beneficios a un grupo de control por largos períodos, con frecuencia es imposible estimar los impacto de largo lazo [1].

2.1.5. Evaluaciones Cuasi-experimentales

Las evaluaciones cuasi-experimentales son aquellas que buscan estimar el impacto causal, pero a diferencia de las experimentales, no se basan en la asignación aleatoria de la intervención [5], ya sea porque esta no es factible o porque plantea problemas éticos. En estos casos, la regla de asignación suele ser menos clara, por lo que se convierte en una incógnita adicional en la evaluación, acerca de la cual se deben formular supuestos [5]. Para ello, estos métodos intentan *simular* la aleatorización de un diseño experimental controlando las diferencias entre los individuos tratados y no tratados bajo diferentes hipótesis.

Si bien los métodos cuasi-experimentales suelen ser más adecuados en algunos contextos operativos, su desventaja es que requieren más condiciones para garantizar que el

grupo de comparación provea una estimación válida del contrafactual, como veremos a continuación al profundizar en algunos de ellos.

Diferencias en Diferencias

El modelo de diferencias-en-diferencias (DD) es una manera de controlar la estimación del impacto por las diferencias preexistentes entre el grupo de tratamiento y el de control, que estará formado por todos aquellos individuos elegibles que no recibieron el tratamiento. Es decir, el DD no propone una forma de construir un grupo de control adecuado sino una manera de aproximar el impacto que tenga en cuenta las diferencias entre participantes y no participantes. A grandes rasgos, combina las comparaciones antes-después con las comparaciones de inscritos y no inscritos, ambas discutidas anteriormente.

Dada una variable de resultado Y, este modelo establece el impacto simplemente como el cambio esperado en Y entre el período posterior y el período anterior a la implementación del tratamiento en el grupo de tratamiento, menos la diferencia esperada en Y en el grupo de control en el mismo período [1]. Es decir, **compara los cambios en los resultados a lo largo del tiempo** entre unidades participantes y no participantes.

Si denotamos con Y_1 al valor de la variable Y en la línea de base, es decir justo antes de la implementación del programa, y con Y_2 al valor de la misma variable en una período posterior a la implementación (también llamado período de seguimiento), entonces el impacto del programa por el método de DD estaría dado por:

$$\tau_{DED} = (\mathbb{E}[Y_2|D=1] - \mathbb{E}[Y_1|D=1]) - (\mathbb{E}[Y_2|D=0] - \mathbb{E}[Y_1|D=0]) \tag{2.9}$$

Es importante señalar que el contrafactual que se estima en este caso es el cambio en los resultados del grupo de tratamiento, y su estimación se logra midiendo el cambio en los resultados del grupo de comparación [5]. En lugar de contrastar los resultados entre los grupos de tratamiento y comparación después de la intervención, la técnica de DD estudia las tendencias entre los grupos de tratamiento y comparación [5].

Como explicamos con anterioridad, el principal problema de las comparaciones antes-después en el grupo de tratamiento es que se asume que lo único que hizo cambiar la variable de resultado fue el programa, cuando en realidad puede haber sido por factores externos. El DD busca solucionar este problema basándose en la idea que la diferencia en los resultados antes-después para el grupo de tratamiento controla por factores que son constantes a lo largo del tiempo en ese grupo, y la manera de capturar los factores variables en el tiempo es medir el cambio antes-después en los resultados de un grupo que no se inscribió en el programa pero que estuvo expuesto al mismo conjunto de condiciones ambientales. De esta forma, si se "limpia" la primera diferencia de otros factores variables en el tiempo que influyen en el resultado de interés sustrayendo la segunda diferencia, se habrá eliminado una fuente de sesgo [5].

Por otro lado, en las comparaciones de inscritos y no inscritos, el gran inconveniente era el sesgo de autoselección. Para entender mejor la solución que el DD propone a esto,

resulta conveniente reescribir el estimador como:

$$\tau_{DED} = (\mathbb{E}[Y_2|D=1] - \mathbb{E}[Y_2|D=0]) - (\mathbb{E}[Y_1|D=1] - \mathbb{E}[Y_1|D=0]) \tag{2.10}$$

De aquí se puede ver lo que plantea es restarle al cambio visto en la variable entre tratados y no tratados luego de la implementación del programa (Y_2), las diferencias que ya pueda haber habido entre ellos en la línea de base (Y_1). Es decir, desde este punto de vista, se utiliza a $\mathbb{E}[Y_1|D=1] - \mathbb{E}[Y_1|D=0]$ como un estimador de las diferencias preexistentes entre el grupo de tratamiento y el grupo de control.

Ahora bien, volviendo a la Ecuación 2.9, el supuesto que permite utilizar la diferencia en el grupo de no inscritos para controlar los factores externos que afectan a los inscritos es que, en ausencia del programa, dichos factores habrían impactado a los tratados de la misma manera que a los no tratados. En otras palabras, se asume que, sin la intervención, los resultados en el grupo tratado habrían evolucionado en paralelo con los del grupo de control, aumentando o disminuyendo en la misma proporción. Esto se conoce como **supuesto de tendencias paralelas**, y es sobre el se hay que basarse al trabajar con este método ya que claramente no es posible observar qué habría ocurrido con el grupo tratado en ausencia del programa.

Efectos Fijos

Propensity Score Matching

Otro enfoque para lograr una buena aproximación del contrafactual consiste en construir un grupo de comparación artificial, a partir del conjunto de individuos que no recibieron el tratamiento. En esto se basa un método conocido como **pareamiento** o **emparejamiento**, que utiliza técnicas estadísticas y grandes bases de datos para construir el mejor grupo de comparación posible sobre la base de características observables [5]. La principal utilidad de este método es que puede aplicarse en el contexto de casi todas las reglas de asignación de un programa, siempre que se cuente con un grupo que no haya participado en el mismo [5].

El supuesto que utilizan estas técnicas para basarse en características observables al construir el grupo de control es que tanto la participación en el programa como los resultados potenciales están únicamente determinados por estas. En otras palabras, se asume que al condicionar los resultados potenciales de tratados y no tratados en determinadas variables observables (pertinentes en el programa bajo estudio), el sesgo de autoselección es igual a cero. Formalmente, si denotamos con X a los rasgos observables, este supuesto se escribe de la siguiente forma:

$$\mathbb{E}[Y_i(0)|D_i=1, X] = \mathbb{E}[Y_i(0)|D_i=0, X], \tag{2.11}$$

y se lo conoce como **supuesto de Independencia Condicional** (IC).

La versión más directa de esta metodología consiste en encontrar un "clon" *para cada individuo* tratado en el grupo de no tratados y contrastar las variables de resultados de ambos [1]. Un clon en este caso quiere decir un individuo (o grupo de individuos) con características observables X lo más parecidas posible, como pueden ser el ingreso de una persona, su nivel de educación, edad, sexo, o cualquier variable que sea pertinente para el programa bajo estudio [1].

Notemos entonces que la búsqueda de una buena pareja para cada participante del programa requiere aproximarse todo lo posible a las características que explican la decisión del individuo de inscribirse en el programa [5]. Sin embargo, en la práctica esto es más difícil. Si la lista de características observables relevantes es muy grande, o si cada característica adopta muchos valores, puede que sea complicado identificar una pareja para cada una de las unidades del grupo de tratamiento [5]. Esta situación es conocida como *el problema de la dimensionalidad*.

Este problema puede solucionarse utilizando un método denominado pareamiento por puntajes de propensión (*Propensity Score Matching*, PSM). Consiste en emparejar individuos ya no con base en X, sino en su probabilidad estimada de participación en el programa, dadas sus características observables, es decir en:

$$P(X) = P(D_i = 1|X)$$

donde P denota el operador de probabilidad de un evento. Este valor es conocido como **probabilidad de participación** o **puntaje de propensión**. De esta forma, el clon adecuado para cada individuo del grupo de tratamiento será aquel del grupo de no tratados con una probabilidad de participación en el programa suficientemente cercana [1]. La ventaja de emparejar a partir de $P(X)$, a diferencia de X, es que $P(X)$ es un número, mientras que X puede tener una dimensión muy grande [1]. Cabe aclarar que para calcular este puntaje, solo deberían utilizarse las características en la línea de base, ya que post-tratamiento, estas pueden haberse visto afectadas por el propio programa [5].

Ahora bien, puede surgir la siguiente pregunta: ¿es posible encontrar al menos una pareja para todos los individuos tratados? Y la respuesta es no, y para aquellos individuos no será posible estimar el impacto del programa. Lo que puede ocurrir en la práctica es que para algunas unidades inscritas, no haya unidades en el conjunto de no inscritos que tengan puntajes de propensión similares. En términos técnicos, puede que se produzca una falta de **rango o soporte común**.

La condición de soporte común (SC) establece que individuos con el mismo vector de variables X tienen probabilidad positiva de ser tanto participantes como no participantes del programa [1]. Esto es, fijado un vector de variables $\hat{X}$:

$$0 < P(D_i = 1|\hat{X}) < 1$$

Si esta condición no se cumple, sería posible encontrar una combinación particular de características que predice perfectamente la participación en el programa, y por tanto, no existiría un individuo que fuera un buen control (o viceversa) [1].

El SC implica que solo se utilizan en la estimación aquellos individuos tratados para los cuales se pueda encontrar uno no tratado con un puntaje de propensión similar. Por ejemplo, si existen individuos del grupo de tratamiento con una probabilidad de participación muy alta, pero ningún individuo no participante exhibe un puntaje tan alto, entonces estos individuos tratados se descartarán a la hora de hacer el emparejamiento.

Asumiendo que se cumplen las condiciones de CI y SC, el estimador del ATT por PSM estaría dado por:

$$ATT_{PSM} = \mathbb{E}_{P(X)|D=1} \{\mathbb{E}[Y_i(1)|D_i = 1, P(X)] - \mathbb{E}[Y_i(0)|D_i = 0, P(X)]\}$$

donde $\mathbb{E}_{P(X)|D_i=1}$ es el valor esperado con respecto a la probabilidad de participación $P(X)$, condicional en ser participante del programa [1]. Es decir, un promedio ponderado de las diferencias entre corchetes, donde los ponderadores son funciones de la probabilidad de participación en el programa [1].

2.2. Inteligencia Artificial

La definición más general acerca de la Inteligencia Artificial (IA) establece que es el área de las Ciencias de la Computación que se enfoca tanto en entender como en crear sistemas que simulen comportamientos *inteligentes*. Si bien existen distintas perspectivas sobre qué significa que una computadora actúe de manera inteligente, la que más ha prevalecido a lo largo de los años es aquella que refiere con la capacidad de computar cómo actuar de la mejor manera posible en una determinada situación [14].

En sus comienzos, los métodos desarrollados en el área de la IA estaban principalmente **basados en conocimiento**, es decir reglas matemáticas formales que permitían a las computadoras llevar a cabo inferencias lógicas y de esta forma resolver problemas que eran intelectualmente difíciles para los humanos [6].

Sin embargo, determinar reglas que describan la complejidad y diversidad de la realidad no era una tarea fácil. De esta manera, con el objetivo de hacer a estos sistemas más flexibles y capaces de adaptarse y entender diferentes situaciones, se transicionó hacia un enfoque en el que estos pudieran obtener su propio conocimiento aprendiendo patrones directamente a partir de los datos en lugar de depender exclusivamente de reglas predefinidas.

Este cambio de paradigma dio lugar a lo que hoy se conoce como **Aprendizaje Automático**, la subdisciplina de la IA que permite a los algoritmos mejorar su desempeño en una tarea específica automáticamente a partir de la experiencia. Diversos factores como la creciente disponibilidad de datos, el aumento en la capacidad computacional, y los avances en los algoritmos de optimización [6] han hecho que esta área sea la que mayor desarrollo e impacto ha tenido durante las últimas décadas.

Actualmente, la IA abarca una diversidad de tareas, que van desde lo general, como son las habilidades de aprendizaje, razonamiento, y percepción, entre otras; hasta lo específico,

como probar teoremas matemáticos, manejar vehículos, mejorar procesos industriales o incluso diagnosticar enfermedades.

2.2.1. Aprendizaje Automático

El Aprendizaje Automático, más conocido por su nombre en inglés *Machine Learning* (ML), es un campo dentro de la IA cuyo objetivo es desarrollar técnicas que permitan que las computadoras *aprendan* automáticamente a partir de la *experiencia* - los datos -, sin la necesidad de ser explícitamente programadas para hacerlo.

Un ejemplo de estos algoritmos puede ser un clasificador de correo spam, que aprende a distinguir correos spam de regulares viendo ejemplos de cada tipo de correo. Otro ejemplo puede ser un sistema que aprenda a predecir la edad de una persona a partir de una imagen habiendo experimentado previamente algunos ejemplos de imagen-edad.

En 1997, Tom Mitchell definió en su libro Machine Learning [10] el concepto de "aprender" de la siguiente manera: "Se dice que un programa de computadora aprende de la experiencia E con respecto a una clase de tareas T y una medida de desempeño P, si su desempeño en las tareas de T, medido por P, mejora con la experiencia E". Para tener un mejor entendimiento, nos enfocamos a continuación en cada uno de estos tres componentes.

Las tareas de ML se describen usualmente en términos de cómo el sistema debería procesar un *ejemplo* o *entrada*, entendiendo a esta como un conjunto de características (o en inglés, *features*) medidas cuantitativamente a partir de un cierto objeto o evento [6]. Por ejemplo, una entrada puede estar compuesta por los datos de un hogar, como la cantidad de habitaciones y de baños, si tiene patio o no, el tamaño de la cocina, etcétera. Dentro de las tareas que pueden ser resueltas por un sistema de ML se encuentran la clasificación, la regresión, la traducción, la detección de objetos en imágenes, la generación de nuevos datos, la detección de valores atípicos, entre muchas otras.

La experiencia hace referencia al tipo de información que el algoritmo "puede ver" durante su proceso de aprendizaje o *entrenamiento* [4]. A esta información se la conoce como "conjunto (de datos) de entrenamiento", y es un subconjunto del *dataset*, que es simplemente el conjunto de todos los ejemplos o datos con los que se cuenta. En base a la experiencia, los algoritmos de ML se clasifican en dos grandes categorías:

- **Algoritmos de Aprendizaje Supervisado**: experimentan un dataset que contiene pares de entrada-salida, esto es cada ejemplo contiene sus features pero también su "etiqueta", que vendría a ser la "respuesta correcta" para dicha entrada. El algoritmo aprende una función que asigna (*mapea*) entradas a salidas. Las tareas más comunes llevadas a cabo con este tipo de aprendizaje son las de regresión, en donde la etiqueta corresponde a un valor continuo; y clasificación, en donde la solución viene dada por la categoría (dentro de un conjunto predefinido de categorías) a la que pertenece un ejemplo.

- **Algoritmos de Aprendizaje No Supervisado**: ven un dataset que cuenta solamente con características de cada entrada pero sin etiquetas, e intentan aprender automáticamente patrones y propiedades útiles de la estructura de los datos. Algunas tareas que se llevan a cabo con este tipo de aprendizaje son *clustering*, reducción de dimensionalidad, y detección de anomalías.

También existen otros tipos de algoritmos, como los de **Aprendizaje Semi-supervisado**, en donde el dataset contiene algunos ejemplos etiquetados y otros sin etiquetas; y los de **Aprendizaje por Refuerzo**, en donde el algoritmo aprende la mejor estrategia para una situación a partir de la interacción con su entorno en forma de recompensas y castigos.

Por último, para medir el desempeño de estos algoritmos, se definen métricas cuantitativas que dependen de la tarea que se esté realizando y del objetivo que se intente lograr. Por ejemplo, en clasificación, una de las más comunes es la de exactitud (*accuracy*), que es la proporción de ejemplos para los cuales se predijo la salida correcta (dada por el dataset); aunque también existen otras como la precisión, sensibilidad, especificidad, y el puntaje F1, que pueden resultar más adecuadas según el dominio del problema. Por otro lado, en tareas de regresión, algunas métricas que se suelen utilizar son el Error Cuadrático Medio y el Error Absoluto Medio.

El objetivo fundamental del ML es que un algoritmo actúe correctamente ante nuevas entradas desconocidas, es decir que **generalice** más allá de los ejemplos del conjunto de entrenamiento. Por ello, aunque las métricas anteriores pueden calcularse durante el entrenamiento para verificar que el algoritmo esté mejorando, su verdadero desempeño se evalúa en un subconjunto del dataset distinto al de entrenamiento, denominado "conjunto de test". Como buena práctica, este conjunto debe permanecer completamente separado del proceso de aprendizaje para obtener una estimación realista de la capacidad de generalización del algoritmo.

Habiendo presentado una idea general sobre cuál es el objetivo de los algoritmos de Aprendizaje Automático, ahora nos enfocaremos en los de Aprendizaje Supervisado, que son los que usaremos en este trabajo.

Aprendizaje Supervisado

Como mencionamos anteriormente, en el Aprendizaje Supervisado el algoritmo aprende una función que mapea entradas a salidas a partir de un conjunto de entrenamiento compuesto por datos etiquetados. El objetivo es que al presentarle a esta función una nueva entrada no vista previamente, esta sea capaz de computar la salida que mejor se ajusta a los **patrones** aprendidos durante el entrenamiento.

Resulta conveniente introducir en este punto un término muy utilizado en el ML: **modelo**. Un modelo es simplemente una ecuación matemática, que se presenta como una forma simplificada de describir hechos de la realidad. En este contexto, será una manera

de intentar capturar las relaciones entre los datos, con el objetivo de hacer predicciones basadas en los patrones aprendidos de ejemplos previos. Habiendo presentado este concepto, la función de la que hablamos en el párrafo anterior se suele llamar modelo.

Formalmente, una tarea de Aprendizaje Supervisado es la siguiente:

> Dado un conjunto de entrenamiento de N ejemplos de entrada-salida:
>
> $$(x_1, y_1), (x_2, y_2), ..., (x_N, y_N)$$
>
> donde cada par fue generado por una **función desconocida** $y = f(x)$ (x denota una entrada cualquiera del mismo tamaño que los x_i), descubrir una función h que aproxime a la función real f [14].
> Por comodidad, denotaremos con $\mathbf{x}$ al vector de entradas del conjunto de entrenamiento, y con $\mathbf{y}$ al vector de salidas, ambos de tamaño N y bien ordenados (notar la *negrita*). Es decir:
>
> $$\mathbf{x} = (x_1, x_2, ..., x_N)$$
> $$\mathbf{y} = (y_1, y_2, ..., y_N)$$
>
> Cabe notar que cada x_i puede ser a su vez un vector, que por convención asumiremos que es de números reales. Es decir $x_i \in \mathbb{R}^n$, con $n \in \mathbb{N}$.

De esta forma, el modelo es la función h, que toma una entrada x y devuelve una salida y, y se presenta como una **hipótesis** de f. La suposición sobre la que se trabaja es que si el modelo funciona bien para los pares de entrenamiento, entonces se espera que va a realizar buenas predicciones para nuevas entradas cuya etiqueta se desconoce.

Ahora bien, esta función h se obtiene a partir de un **espacio de funciones**, que es elegido por quien diseña el modelo. Este espacio podría ser el conjunto de funciones lineales, de polinomios de grado 2, de polinomios de grado 3, etcétera. Por lo tanto, el espacio determina la forma o "**arquitectura**" que va a tener la función h, y consecuentemente cuáles son sus parámetros, cuyo vector denotaremos con la letra ϕ. De esta forma, este espacio determina la familia de posibles relaciones entre entradas y salidas, y los parámetros especifican la relación particular (el modelo).

Por ejemplo, si suponemos que cada entrada x es un número real y establecemos como espacio el conjunto de funciones lineales, entonces la forma de h será:

$$h(x) = \phi_1 x + \phi_0$$

y sus parámetros $\phi = (\phi_0, \phi_1)$. Este caso es comúnmente conocido como regresión lineal unidimensional (ya que la entrada es simplemente un número real). Modelos particulares dentro de este espacio son $h(x) = 5x + 13$, $h(x) = \pi x + 1{,}25$, entre muchos otros.

Otro caso un poco más complejo podría ser el que $x \in \mathbb{R}^3$, es decir $x = (x_1, x_2, x_3)$ y seguimos tomando una relación lineal. Aquí, h sería:

$$h(x) = \phi_1 x_1 + \phi_2 x_2 + \phi_3 x_3 + \phi_0$$

y sus parámetros $\phi = (\phi_0, \phi_1, \phi_2, \phi_3)$[1].

En cambio, si tomamos el conjunto de polinomios de grado 2, entonces la forma de h será:

$$h(x) = \phi_2 x^2 + \phi_1 x + \phi_0$$

y sus parámetros $\phi = (\phi_0, \phi_1, \phi_2)$. A este caso se lo suele llamar regresión polinómica de segundo grado unidimensional.

Notemos entonces que quienes van a determinar qué modelo es "mejor" que otro, es decir qué modelo aproxima mejor a f, son los valores de los parámetros ϕ_i. Por lo tanto, h en realidad es una función de la entrada x pero también de los parámetros, establecidos por el espacio de funciones elegido:

$$h(x, \phi)$$

Y lo que querremos idealmente es:

$$f(x) \approx h(x, \phi)$$

Entonces, cuando un modelo se entrena o aprende, lo que realmente hace es encontrar los valores de los parámetros que describan la verdadera relación entre entradas y salidas [11]. Un algoritmo de aprendizaje toma el conjunto de entrenamiento y manipula los parámetros hasta que las predicciones dadas por él sean lo más cercanas posible a las etiquetas verdaderas.

Para que el algoritmo sepa cómo modificar estos parámetros para mejorar sus predicciones, necesita una forma de saber cómo está siendo su desempeño. Para esto, se define lo que se conoce como **función de pérdida** o **función de error**, que denotaremos con la letra L y que justamente hace eso: retorna un número que resume qué tan bien o mal está funcionando el modelo con sus parámetros ϕ actuales, en términos de qué tan lejos están sus predicciones de las respuestas reales del conjunto de entrenamiento.

Una vez establecida su forma, la *performance* del modelo va a estar dada por los valores actuales de sus parámetros. Por lo tanto, tiene sentido tratar a la función de pérdida como una que depende de los parámetros del modelo, es decir $L(\phi)$[2]. A continuación, se mencionan dos ejemplos de funciones de error:

- Una función de pérdida que se es común usar en problemas de regresión es la de Error Cuadrático Medio (ECM), dada por:

$$ECM(\phi) = \frac{1}{N} \sum_{i=1}^{N} \left(h(x_i, \phi) - y_i\right)^2$$

[1]Para simplificar la notación, lo que se suele hacer es asumir que la entrada x tiene un componente adicional constante con el valor 1, es decir $x = (x_1, x_2, x_3, 1)$. De esta forma, podríamos escribir: $h(x) = x \cdot \phi$, donde el operador $\cdot$ representa el producto escalar

[2]En realidad la función de pérdida también depende de los datos de entrenamiento, por lo que si somos estrictos deberíamos escribir $L(\{\mathbf{x}, \mathbf{y}\}, \phi)$. Sin embargo, como estos datos están fijos durante todo el proceso de aprendizaje, podemos omitirlos.

- Para problemas de clasificación binaria como el que tratamos en este trabajo, se suele emplear la Entropía Cruzada Binaria (ECB), dada por:

$$ECB(\phi) = -\frac{1}{N}\sum_{i=1}^{N}[y_i\ log(h(x_i,\phi)) + (1-y_i)\ log(1-h(x_i,\phi))]$$

  Sin entrar en detalles, lo que mide esta función es la diferencia entre dos distribuciones de probabilidad, que en este caso son la distribución real de los datos dada por el conjunto de entrenamiento y la distribución predicha por el modelo en su estado actual.

En general, se asume que un valor más bajo de $L(\phi)$ indica un mejor desempeño del modelo, y por lo tanto el objetivo del proceso de entrenamiento se convierte **encontrar los parámetros ϕ^* que minimicen la función de pérdida**:

$$\phi^* = \arg\min_{\phi} L(\phi)$$

Si luego de la minimización la pérdida es baja, quiere decir que hemos encontrado parámetros que predicen precisamente las salidas de entrenamiento y_i a partir de los entradas x_i (con $i = 1, ..., N$). Sin embargo, como explicamos previamente, la evaluación final del modelo se debe hacer con el conjunto de test, sobre el cual se puede también calcular la pérdida.

La pregunta que surge entonces es cómo hacer para lograr esta minimización. Dependiendo del espacio de funciones y de la función de costo seleccionada, puede que exista una fórmula cerrada para ϕ^*, que se calcule analíticamente. Sin embargo, si la función de costo tiene muchas variables (parámetros) o los modelos son complejos, resulta más útil recurrir a métodos numéricos para aproximar este mínimo.

El algoritmo por defecto que se utiliza para resolver este problema de optimización es el llamado **Descenso por el Gradiente** (DG). Es muy potente ya que se puede aplicar a cualquier función de pérdida [14], sin importar el espacio de funciones elegido.

La idea del DG es ir modificando los parámetros iterativamente hasta eventualmente llegar a un "valle" de la función de pérdida. El algoritmo se basa en que la dirección dada por el gradiente[3] de una función en un punto es la de máximo crecimiento, y por lo tanto su opuesta es la de máximo decrecimiento. En este caso, la función de la cual se calcula el gradiente es la de error, con respecto a los parámetros en ϕ.

Concretamente, se empieza con un vector de parámetros ϕ con valores arbitrarios que

[3]Dada una función g de varias variables, es decir, $g(v_1, v_2, ..., v_M)$, el **gradiente** de g es el vector formado por las derivadas parciales de g con respecto a cada uno de sus parámetros:

$$\nabla g = \left(\frac{\partial g}{\partial v_1}, \frac{\partial g}{\partial v_2}, \ldots, \frac{\partial g}{\partial v_M}\right)$$

va mejorando gradualmente, tomando un paso a la vez en dirección opuesta al gradiente, con el objetivo de reducir el valor de la función de pérdida, hasta que el algoritmo *converja* a un mínimo (local) de la función de pérdida.

Un hiperparámetro[4] determinante en el algoritmo del DG es el tamaño de los pasos llamado **tasa de aprendizaje**, que denotaremos con la letra α y supondremos por el momento que se mantiene constante en todo el proceso. Si el valor α es muy pequeño, entonces el algoritmo va a tener que realizar muchas iteraciones para converger [4]. Si en cambio el valor es muy alto, entonces puede ser que vayamos "saltando" de un lado a otro de un valle de la función, haciendo que el algoritmo diverja [4].

La regla del DG está dada por:

$$\phi_{p+1} = \phi_p - \alpha \nabla L(\phi)$$

donde p indica el número de iteración actual.

La parte más "pesada" computacionalmente de este algoritmo es que en la función de pérdida, de la forma en la que la definimos hasta ahora, están incluidas todas las muestras del conjunto de entrenamiento. Es decir, en cada iteración, el cálculo del gradiente requiere evaluar la contribución de cada una de las muestras, lo que puede ser caro cuando N es muy grande. Para mitigar este problema, existen variantes del algoritmo que buscan aproximar el gradiente utilizando subconjuntos del conjunto de entrenamiento, como el Descenso por el Gradiente Estocástico y el Descenso por el Gradiente por Mini-Lotes.

Hasta aquí hemos hablado de cómo está compuesto prácticamente cualquier algoritmo de aprendizaje supervisado. Sin embargo, no hemos discutido la correspondencia del espacio de funciones con el conjunto de datos con el que estamos tratando.

El éxito del aprendizaje supervisado depende en gran medida de elegir un espacio de funciones adecuado que sea capaz de capturar la distribución de los datos. Por ejemplo, si elegimos un modelo lineal cuando los datos tienen una relación cuadrática, el modelo no podrá representar correctamente la estructura del problema. Por otro lado, si seleccionamos un espacio de funciones altamente flexibles - o *expresivas* -, como polinomios de grado alto, puede ocurrir que el modelo termine describiendo peculiaridades de los datos de entrenamiento que son atípicas y llevan a predicciones inusuales [11].

Si bien existen otros modelos que permiten identificar relaciones no lineales, varios de ellos requieren tener un buen conocimiento sobre la estructura de los datos, y muchas veces su desempeño está condicionado a ciertas transformaciones, proceso que se conoce por su nombre en inglés *feature engineering*.

---

donde la notación $\frac{\partial g}{\partial v_i}$ representa la **derivada parcial** de g con respecto a v_i, que mide cómo cambia g cuando se varía v_i mientras se mantienen constantes las demás variables. Este vector apunta en la dirección de mayor crecimiento de g.

[4]Hablaremos más adelante sobre qué es un hiperparámetro, pero lo importante es que no es un parámetro que se aprende, como lo son los contenidos en el vector ϕ.

En este punto es donde aparecen los modelos conocidos como **Redes Neuronales**, que permiten describir un espacio de funciones complejas, expresivas, y no lineales sin la necesidad de tener un conocimiento profundo sobre los datos. En cambio, son capaces de aprender las representaciones subyacentes automáticamente. En la sección siguiente, explicaremos el funcionamiento de las Redes Neuronales.

2.3. Redes Neuronales Artificiales

Las Redes Neuronales Artificiales (RNAs) son un modelo específico dentro del ML, cuya estructura y funcionamiento están inspirados en el intento de simular los del cerebro. Durante los últimos años, ha sido el área que más desarrollo e impacto ha tenido, principalmente gracias a su versatilidad, potencia y escalabilidad, cualidades que hacen que sean capaces de enfrentar problemas grandes y complejos [4].

Breve Contexto Histórico

Aunque su gran éxito ha sido reciente, lo cierto es que la idea de las RNAs data desde 1943, cuando Warren McCulloch y Walter Pitts presentaron en su artículo *A Logical Caclulus of Ideas Immanent of Nervous Activity* [9] un modelo computacional simplificado utilizando lógica proposicional acerca de cómo funcionan las neuronas de cerebros animales en conjunto para llevar a cabo cómputos complejos [4]. Presentaron una versión muy simplificada de la neurona biológica, que solamente tenía una o más entradas binarias y una salida binaria.

Posteriormente, en 1958, Frank Rosenblatt presentó una de las arquitecturas más simples de RNAs: el "Perceptrón" [12], el cual servía principalmente para resolver problemas de clasificación en donde los datos son linealmente separables. Su aporte más notorio es que definió un algoritmo para el entrenamiento del Perceptrón que le permitía mejorar automáticamente sus parámetros internos para poder llegar una solución.

Más tarde, se descubrió que los problemas que no podían ser resueltos por el Perceptrón, sí podían ser resueltos "apilando" múltiples perceptrones, lo cual llevó a la invención del "Perceptrón Multicapa" (PMC), también conocido como "Red Neuronal de Propagación Directa" (del inglés *Feedforward Neural Networks*, FFNN) [6].

Durante los últimos años se han desarrollado diversas *arquitecturas* de redes neuronales, complejizando los algoritmos y las técnicas para lograr resolver determinados tipos de problemas. Sin embargo, todas ellas tienen elementos en común y se entrenan prácticamente de la misma manera. Para explicar estos aspectos, tomaremos como referencia una Red Neuronal Feedforward.

Red Neuronal Feedforward

El elemento básico de cualquier red neuronal es la **neurona artificial**, que es la que lleva a cabo el "cómputo". Una neurona está conectada con otras, de las cuales recibe información y a quienes también les envía. Estas conexiones representan las **sinapsis** de las neurona biológicas y cada una tiene asociada un determinado número llamado **peso**, que simboliza la intensidad de la conexión.

Matemáticamente, lo que hace una neurona es primero computar una suma pesada de sus entradas:

$$z = w_1 x_1 + w_2 x_2 + \cdots + w_n x_n,$$

donde x_i representa la entrada proveniente de la neurona i y w_i es el peso asociado a esta conexión. Esta suma también se puede ver como el producto escalar entre el vector de entradas y el vector de pesos.

Sobre este valor resultante, aplica una **función de activación** h, siendo como resultado final de la neurona:

$$h(z) = h(w_1 x_1 + w_2 x_2 + \cdots + w_n x_n)$$

Esto se puede ver gráficamente en la Figura 2.2.

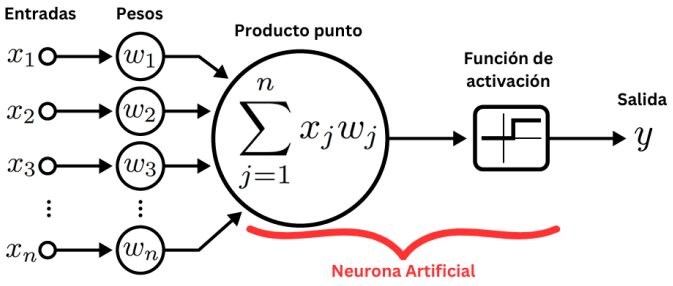

Figura 2.2: La neurona computa una suma pesada de sus entradas y aplica una función de activación sobre el resultado.

La idea detrás de la función de activación es el hecho de propagar la información proporcionalmente al estímulo provocado por las entradas en la neurona, basándose en la idea de "disparo" de una neurona biológica en el momento en que se supera el umbral de activación. La función de activación utilizada en el Perceptrón, mencionado anteriormente, fue la llamada *heavyside*, dada por:

$$heavyside\,(z) = \begin{cases} 0 \text{ si } z < 0 \\ 1 \text{ si } z \geq 0 \end{cases}$$

Durante los años, se han empleado diferentes funciones de activación, como la sigmoide, la tangente hiperbólica, y la ReLU, dada por $\text{ReLU}(z) = \max(0, z)$, siendo esta última la más común actualmente.

Dicho esto, en una red neuronal está compuesta por diferentes **capas de neuronas**: la capa de entrada, que es por donde ingresan los datos, una o más capas ocultas o intermedias, y finalmente la capa de salida.

En una red Feedforward particularmente, las neuronas de una capa están conectadas solamente con las neuronas de la capa siguiente, implicando de esta forma que la información fluye solamente en una dirección (de aquí la nomenclatura Feedforward). Algo a notar también sobre estas redes es que cada neurona de una capa está conectada con todas las neuronas de la capa siguiente, razón por la cual este tipo de conexión también se suele llamar "densa". Esta situación puede verse en la Figura 2.3.

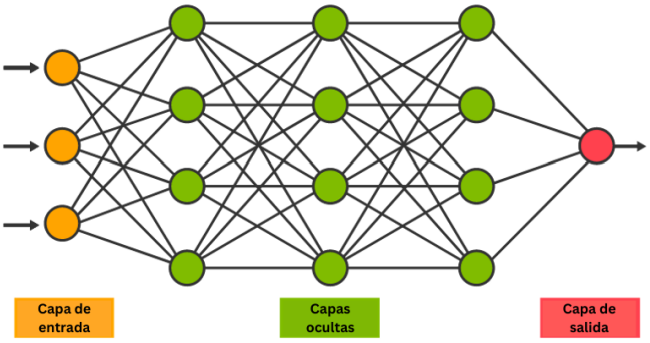

Figura 2.3: Esquema de una Red Neuronal Profunda, con tres capas ocultas. Las neuronas naranjas forman la capa de entrada, las neuronas verdes forman las tres capas ocultas, y la roja la de salida. Cabe notar que la capa de salida podría tener más de una neurona.

Ahora bien, el objetivo de una red neuronal es **aproximar una función desconocida** g^*. Para un clasificador por ejemplo, $y = g^*(x)$ determina una categoría y para una entrada x. De esta forma, una red define una asignación $y = g(x; \theta)$ y aprende el valor de los **parámetros** θ que resultan en la mejor aproximación de la función "oculta" [6]. En este caso, los parámetros a aprender por la red serán los pesos sinápticos de todas las conexiones.

Matemáticamente, los pesos entre una capa de neuronas y la siguiente se pueden pensar como una matriz w, en donde la entrada w_{ij} indica el peso sináptico de la conexión entre la neurona i de la capa "actual" y la neurona j de la capa "anterior". La pregunta que surge entonces es **cómo** hacer para que la red pueda aproximar esta función.

Entrenamiento de una RNA Feedforward

A grandes rasgos, el entrenamiento de una red consiste en una suma de **evaluación** y **optimización** [3]. Con evaluación, nos referimos al establecimiento de una medida que determine qué tan lejos está el modelo de hallar una buena solución, para lo cual se suele definir una función a minimizar o maximizar llamada **función objetivo** o **criterio** [6];

y la optimización es el método usado para aproximarse a dicha solución, maximizando o minimizando la función objetivo según corresponda.

Cuando lo deseado es minimizar la función objetivo, esta pasa a ser llamada **función de costo**, **función de error** o **función de pérdida**. En problemas de aprendizaje supervisado, esta función compara la salida de la red para determinadas entradas - la **predicción** - en su estado actual, dado por el valor de los pesos sinápticos en ese momento, con las salidas reales para dichas entradas, dadas por el conjunto de entrenamiento. En resumen, una vez que están fijos los datos del conjunto de entrenamiento, la función de costo es una función de los pesos de la red. Por lo tanto, la denotaremos con $f(w)$.

Dos de las funciones de costo más utilizadas son el **Error Cuadrático Medio** y la **Entropía Cruzada**. De estas dos funciones, nos concentraremos en la última ya que es la más apropiada para problemas de clasificación como el que tratamos. Esto se debe a que mide la diferencia entre la distribución de probabilidad predicha por el modelo y la distribución real de los datos. En este caso, la capa de salida tiene tantas neuronas como categorías haya, y la salida de cada una de ellas representa el puntaje que le da la red a la categoría correspondiente a cada neurona, donde un mayor puntaje indica una mayor probabilidad de pertenencia de la entrada particular a dicha clase. Para obtener una estimación de la distribución de probabilidad dada por estas salidas, se aplica la función *softmax* [4].

Una vez que se tiene esta función establecida y se calculan las pérdidas para los datos, el algoritmo por defecto que se utiliza para la optimización es el de **Descenso por el Gradiente** (*Gradient Descent*). Este se basa en modificar los parámetros (i.e. los pesos sinápticos) iterativamente con el objetivo de encontrar mínimos locales de una función, que en este caso será la de costo. Se basa en la idea que la dirección dada por el gradiente es la de mayor crecimiento, y por lo tanto su opuesta es la de menor. De esta forma, lo que se hace en cada paso es calcular el gradiente de la función de costo con respecto a los pesos, y luego actualizarlos siguiendo la siguiente regla:

$$w = w - \eta \, \nabla f(w),$$

donde w representa una matriz de pesos de una determinada capa, y la cantidad η es llamada la **tasa de aprendizaje**, que determina el tamaño de cada paso. Si η es muy pequeño, entonces el algoritmo tendrá que hacer muchos pasos para converger; pero si es demasiado grande, puede llegar a divergir.

Actualmente, se utilizan otros optimizadores más sofisticados y eficientes, pero que todos parten de la idea del Descenso por el Gradiente. Los que usaremos en los experimentos son el Descenso por el Gradiente Estocástico (*Stochastic Gradient Descent*, SGD) y Adam (*Adaptive Moment Estimation*), de los cuales se puede leer más en el Capítulo 8 de [6].

Ahora bien, recordando que la función f está fija en todas las entradas y salidas del conjunto de entrenamiento, hay algoritmos de optimización que utilizan todos estos

datos para calcular el gradiente, otros que utilizan de a un ejemplo a la vez para calcular el gradiente y otros que están entre medio de los dos anteriores, es decir utilizan un subconjunto del conjunto de datos para calcular el gradiente (esto es lo que hace SGD).

Es necesario en este punto introducir el concepto de "**época**" y de "**lote**". Un lote es simplemente una subdivisión de tamaño fijo del conjunto de entrenamiento (o de test). Dicho esto, se llama época al proceso completo en el cual el modelo entrena utilizando **todos** los datos disponibles en el conjunto de entrenamiento una vez, ya sea recorriéndolos de a lotes o todos de una vez. Si el conjunto está efectivamente dividido en lotes, una época va a consistir de: **para cada lote**, calcular las predicciones en base a los pesos actuales, obtener los errores de cada muestra del lote, computar los gradientes en base a los ejemplos del lote y actualizar los pesos. Si no, se hacen los mismos pasos pero para todos los datos de una vez.

Al entrenar un modelo, existen ciertos ajustes que son utilizados para controlar su comportamiento llamados **hiperparámetros**. Los valores de los hiperparámetros no son aprendidos por el modelo [6], sino que se fijan de antemano y no se modifican a lo largo del entrenamiento. Algunos de estos son: la cantidad de neuronas en cada capa oculta, la función de activación utilizada en cada capa, el optimizador, la tasa de aprendizaje, el número de épocas, el tamaño de lote, entre otros. Todos estos pueden contribuir a que el modelo mejore su desempeño. Para probar distintos valores de hiperparámetros, que es lo que haremos a continuación, se utiliza un conjunto de datos extra además del de entrenamiento y el de test, llamado de **validación** que sirve justamente para encontrar los hiperparámetros óptimos.

Otras mejoras que se han implementado para mejorar la eficacia de los modelos, y que emplearemos en los experimentos, son las nombradas a continuación. Por un lado, tenemos la **detención temprana**, que consiste en frenar el entrenamiento en el último mejor "estado" del modelo cuando se observa que a partir de dicho punto el error sobre el conjunto de validación no mejora (notar que cuántas épocas se espera y qué se considera una mejora son nuevos hiperparámetros). Y por otro lado, el **dropout**, que es un algoritmo que implica suprimir un cierto número de neuronas de una capa de manera aleatoria [**apuntes-redes-neuronales**].

2.4. Redes Neuronales Convolucionales

2.5. Redes Neuronales Recurrentes

2.5.1. Long Short-Term Memory

2.6. Clasificación de Series de Tiempo

Una **serie de tiempo** es una secuencia **ordenada** de valores medidos sucesivamente en intervalos de tiempo igualmente espaciados [8]. Este tipo de datos está presente en distintos fenómenos, como lecturas meteorológicas, registros financieros, señales fisiológicas, y observaciones industriales [15].

Formalmente, una serie de tiempo **univariada** U es un conjunto ordenado de valores reales $U = (x_1, x_2, ..., x_N$, y su dimensión está dada por la cantidad N de valores [8].

Dicho esto, la **Clasificación de Series de Tiempo** (CST) consiste en: dado un conjunto de clases o categorías Y, y un conjunto de datos D formado por series de tiempo univariadas U_i donde a cada una le corresponde una etiqueta $y(U_i) \in Y$, el objetivo es encontrar una función f, a la que llamaremos "clasificador" o "modelo" de forma tal que $f(U) = y(U)$ [8]. Un buen modelo será uno que pueda capturar y generalizar el el patrón de las series de datos de forma tal que sea capaz de clasificar correctamente nuevos datos. El repositorio que se suele tomar como referencia para evaluar los diferentes modelos de CST es el de la Universidad de California Riverside, llamado *UCR Time Series Classification Archive* [2].

Durante las últimas décadas, se han propuestos diferentes métodos para afrontar este problema. Los primeros trabajaban directamente sobre las series de tiempo utilizando alguna medida de similitud predefinida que pueda capturar la similitud entre ellas, como por ejemplo la distancia euclideana o la deformación dinámica del tiempo (*Dynamic Time Warping*) [15].

Se han desarrollado también algoritmos basados en características o *features*, en donde cada serie temporal es convertida en un conjunto de features globales, que es posteriormente usando para definir la semejanza entre pares de series de tiempo [8]. Algunos de los más conocidos son ...

El gran problema de todas las estrategias mencionadas es que requieren un gran preprocesamiento de los datos y una ingeniería de atributos (*feature engineering*) [15], para lo cual generalmente se necesita tener un buen conocimiento de campo sobre las series sobre las que se está trabajando. Es por esto que recientemente, las redes neuronales profundas han comenzado a ser utilizadas para clasificar series temporales [15] ya que permiten trabajar de forma directa con los datos, y se encargan de detectar automáticamente cuáles son las características que distinguen a las series. Algunas arquitecturas utilizadas hasta el momento son el perceptrón multicapa [15], redes neuronales convolucionales [15], y

estas últimas combinadas con unidades LSTM unidireccionales [7] y bidireccionales [8].

Capítulo 3

Presentación del problema

Como explicamos anteriormente, el principal problema en la Evaluación de Impacto de tratamientos sin asignación aleatoria es el de la identificación de un grupo de control o comparación adecuado, que cumpla las características descritas previamente.

Cuando el ingreso al tratamiento por parte de los individuos se realiza en un único instante de tiempo, el grupo de control obtenido mediante la técnica de PSM conforma el mejor estimador del contrafactual. Sin embargo, cuando la entrada al programa ocurre de manera secuencial, los modelos logísticos no son capaces de gestionar esa probabilidad variable en el tiempo. Lo que se hace entonces es estimar la probabilidad por camada de ingreso al programa, es decir se agrupan individuos que hayan ingresado al tratamiento en temporalidades similares, y se buscan controles separadamente para cada una de estas camadas.

En este trabajo, exploramos una alternativa distinta para este último caso. Proponemos la utilización de redes neuronales, específicamente del tipo LSTM **y convolucionales????**, capaces de incorporar la dimensión temporal de los datos, para la detección automática de grupos de control, en el caso particular en que el programa en cuestión se implementa **secuencialmente** y existe una **alta dependencia temporal** entre una variable observada de los individuos y el momento de ingreso al tratamiento. El principal objetivo es evaluar el potencial y la efectividad de estas redes para capturar dependencias dinámicas y características temporales inherentes a los datos observados, para de esta forma mejorar el proceso de inferencia causal en la evaluación de impacto.

Capítulo 4

Marco Experimental

4.1. Conjuntos de Datos

A continuación, detallamos la metodología utilizada para llevar a cabo las simulaciones y medir el desempeño de las redes, y explicamos cómo este problema se termina enmarcando dentro de una clasificación de series de tiempo.

El primer paso para poder llevar a cabo los experimentos es contar con datos. Nuestro entorno - *framework* - de simulación está diseñado para generar **datos sintéticos** que imiten escenarios reales en donde la asignación al tratamiento es no aleatoria, escalonada y potencialmente dependiente de resultados pasados.

El proceso de generación de datos involucra la creación de una serie de tiempo univariada para cada individuo, que incorpora efectos fijos, componentes autoregresivos e impactos del tratamiento. Trabajamos bajo el supuesto que el comportamiento dinámico de la variable modelada para cada unidad es el que determina si esta ingresa al tratamiento o no, y es la misma sobre la que se espera que el tratamiento tenga un efecto.

Los parámetros para la generación de los datos son los siguientes:

- `n_sample`: cantidad de individuos en la simulación.
- `treated_pct`: porcentaje de individuos tratados.
- `control_pct`: porcentaje de individuos de control.
- `T`: cantidad de períodos observados de cada individuo.
- `first_tr_period`: primer período de tratamiento (o único, dependiendo de la cantidad de cohortes).
- `n_cohorts`: cantidad de cohortes.
- `phiT`: persistencia auto-regresiva para los tratados.
- `phiC`: persistencia auto-regresiva para los controles.

- `n_dep_periods`: cantidad de períodos de dependencia para la participación en el tratamiento.

El resultado de la simulación es un conjunto de datos de panel compuesto por series de tiempo univariadas de longitud `T` para los distintos tipos de individuos:

- **Individuos de tipo 1**: son aquellos que han recibido el tratamiento en algún período. Notar que en una situación real, son datos con los que contamos. Los llamaremos también "tratados".
- **Individuos de tipo 2**: son aquellos que en los datos sintéticos, sabemos que forman parte del grupo de control. Es importante notar que en un escenario real, no sabemos quiénes son estos individuos sino que son los que tratamos de identificar. Nos referiremos a ellos también como "controles".
- **Individuos de tipo 3**: son aquellas unidades que no han sido tratadas y que en los datos sintéticos, sabemos que no forman parte del grupo de control. A estos también los mencionaremos como "NiNi" (ni tratado ni control).

En el dataset, cada individuo tiene un identificador y su tipo; y aquellos de tipo 1 y 2 tienen como feature extra el período (desde `first_tr_period` hasta `first_tr_period + n_cohorts`) en el que ingresaron al programa. A continuación, se muestran algunos ejemplos, tomando `T = 6`, `first_tr_period = 3` y `n_cohorts = 2`:

En este punto cabe recordar cuál es nuestra meta: a partir de información sobre los individuos que fueron tratados, queremos identificar de entre los no tratados, quiénes son los que podrían formar parte del grupo de control.

También resulta muy importante tener en cuenta que en los datos generados sintéticamente, sabemos quiénes son los controles, pero en la realidad esto es justamente lo que queremos identificar.

Ahora bien, para entrenar y evaluar a nuestros modelos, tomamos en cuenta lo que ocurriría en un escenario real, en el que sabríamos solamente quiénes son los tratados y quiénes los no tratados. Por lo tanto, construimos el conjunto de entrenamiento con la totalidad de los individuos de tipo 1 con etiqueta 1 y algunos de tipo 3 con etiqueta 0, y en el conjunto de test colocamos a todos los de tipo 2, queriendo predecir en ellos un 1, y al resto de tipo 3, queriendo predecir en ellos un 0.

Más aún, como queremos identificar a los grupos de control de cada cohorte,

4.2. Arquitecturas de Redes Neuronales

4.3. Herramientas

4.3.1. Python

4.3.2. PyTorch

4.3.3. Pandas

4.3.4. Numpy

4.3.5. Jupyter

4.3.6. Optuna

4.3.7. MLflow

Capítulo 5

Resultados

Capítulo 6

Conclusiones y Trabajos Futuros

Bibliografía

[1] Raquel Bernal y Ximena Peña. *Guía práctica para la evaluación de impacto*. 1.ª ed. Universidad de los Andes, Colombia, 2011. ISBN: bernal2011. URL: http://www.jstor.org/stable/10.7440/j.ctt1b3t82z (visitado 10-02-2025).

[2] Hoang Anh Dau et al. *The UCR Time Series Classification Archive*. https://www.cs.ucr.edu/~eamonn/time_series_data_2018/. Oct. de 2018.

[3] Pedro Domingos. "A few useful things to know about machine learning". En: *Commun. ACM* 55.10 (2012), págs. 78-87. ISSN: 0001-0782. DOI: 10.1145/2347736.2347755.

[4] Aurélien Géron. *Hands-On Machine Learning with Scikit-Learn and TensorFlow*. 1st. O'Reilly Media, 2017. ISBN: 9781491962299.

[5] Paul Gertler et al. *La evaluación de impacto en la práctica: Segunda edición*. Banco Internacional para la Reconstrucción y el Desarrollo/Banco Mundial, 2016. DOI: https://doi.org/10.18235/0006529. URL: https://hdl.handle.net/10986/25030.

[6] Ian Goodfellow, Yoshua Bengio y Aaron Courville. *Deep Learning*. http://www.deeplearningbook.org. MIT Press, 2016.

[7] Fazle Karim et al. "LSTM Fully Convolutional Networks for Time Series Classification". En: *IEEE Access* 6 (2018), págs. 1662-1669. ISSN: 2169-3536. DOI: 10.1109/access.2017.2779939. URL: http://dx.doi.org/10.1109/ACCESS.2017.2779939.

[8] Riaz A. et al. Khan M. Wang H. "Bidirectional LSTM-RNN-based hybrid deep learning frameworks for univariate time series classification". En: *The Journal of Supercomputing* (2021), págs. 7021-7045. URL: https://doi.org/10.1007/s11227-020-03560-z.

[9] Warren McCulloch y Walter Pitts. "A logical calculus of the ideas immanent in nervous activity". En: *The bulletin of mathematical biophysics* (19430). URL: https://doi.org/10.1007/BF02478259.

[10] Tom M. Mitchell. *Machine Learning*. McGraw-hill, 1997. ISBN: 0070428077.

[11] Simon J.D. Prince. *Understanding Deep Learning*. The MIT Press, 2024. URL: http://udlbook.com.

[12] Frank Rosenblatt. "The Perceptron: A Probabilistic Model for Information Storage and Organization in the Brain". En: *Psychological Review* 65.6 (1958), págs. 386-408. DOI: 10.1037/h0042519.

[13] Donald Rubin. "ESTIMATING CAUSAL EFFECTS OF TREATMENTS IN EXPERIMENTAL AND OBSERVATIONAL STUDIES". En: *Journal of Education Psychology* 66.5 (1974), págs. 688-701. DOI: https://doi.org/10.1002/j.2333-8504.1972.tb00631.x. URL: https://onlinelibrary.wiley.com/doi/abs/10.1002/j.2333-8504.1972.tb00631.x.

[14] Stuart J. Russell y Peter Norvig. *Artificial Intelligence: A Modern Approach*. 4, Global Edition. Pearson Education, 2020. ISBN: 9781292401133.

[15] Zhiguang Wang, Weizhong Yan y Tim Oates. *Time Series Classification from Scratch with Deep Neural Networks: A Strong Baseline*. 2016. arXiv: 1611.06455 [cs.LG]. URL: https://arxiv.org/abs/1611.06455.

[16] Howard White. "Theory-based impact evaluation: principles and practice". En: *Journal of development effectiveness* 1.3 (2009), págs. 271-284.