Universidad Nacional de Córdoba
Facultad de Matemática, Astronomía, Física y Computación


# Estudio de representaciones mediante co-embeddings para estudiantes y contenidos en minería de datos educativos

Milagro Teruel

Directores:
Dra. Laura Alonso Alemany y Dr. Marcelo Luis Errecalde

TESIS DOCTORAL
Doctorado en Ciencias de la Computación
Septiembre 2019

# Agradecimientos

Este trabajo no habría sido posible sin la ayuda, el apoyo y el amor de muchas personas a mi alrededor. En primer lugar, mis directores, que me guiaron en todos los aspectos técnicos y personales de este proceso. La obtención del título de doctorado ha demostrado ser mucho más que la mera acumulación de conocimientos, y en tan vasto esfuerzo estuvieron a mi lado, incondicionalmente. Mi pareja, que me acompañó durante las largas noches de experimentación y escritura, y que sigue siendo, siempre, la otra mitad vital de mi trabajo. Mis padres y mi familia, que escucharon pacientemente monólogos sobre mi trabajo con interés, si no con evidente perplejidad, pero siempre mezcladas con matices de orgullo. Profesores y estudiantes de FaMAF, y de otras universidades, que me orientaron y aconsejaron, y prestaron oídos cuando nada funcionaba. A todos ustedes, gracias por estos y otros muchos momentos compartidos, sonrisas, palabras de aliento, que encontré muy necesarias durante el largo camino de convertirme en lo que soy hoy.

## Resumen

Este trabajo es un estudio sobre la generación automática de representaciones basadas en métodos neuronales, en aplicaciones dentro del área de Minería de Datos Educacionales (EDM). Se propone utilizar una arquitectura neuronal recurrente para modelar el cambio en el estado de los estudiantes a medida que interactúan con plataformas de aprendizaje en línea. Al mismo tiempo, se generan representaciones automáticas para los elementos de los cursos, como problemas o lecciones, evitando la necesidad de utilizar ejemplos etiquetados con información adicional, y en consecuencia costosos de obtener. Sobre esta base, se modifica la arquitectura para modelar explícitamente la relación entre la representación de los estudiantes y la de los componentes del curso, proyectando ambos tipos de entidades en el mismo espacio latente. De esta manera, se espera mejorar el desempeño del clasificador a través de la inyección directa de conocimiento de dominio en el modelo.

Ambas propuestas son evaluadas para las tareas de estimación de conocimiento (Knowledge Tracing) y predicción del abandono escolar (dropout) en tutores inteligentes y cursos masivos, respectivamente. Se observa que las representaciones conjuntas de estudiantes y lecciones obtienen resultados similares a las representaciones disjuntas, mejorando significativamente en escenarios con pocos datos o con desbalance de clases pronunciado.



## Abstract

This work is a study on the automatic generation of representations based on neuronal methods, for applications in the area of Educational Data Mining (EDM). We proposed to use a recurrent neuronal architecture to model the change in the state of students as they interact with online learning platforms. At the same time, automatic representations are generated for course elements, such as problems or lessons, avoiding the need to use examples labeled with additional information, and consequently costly to obtain. On this basis, the architecture is modified to explicitly model the relationship between the students' representation and that of the course components, projecting both types of entities in the same latent space. In this way, the performance of the classifier is expected to improve through the direct injection of domain knowledge into the model. Both proposals are evaluated for knowledge tracing and dropout prediction in intelligent tutor systems and mass open online courses, respectively. It

is observed that the joint representations of students and lessons obtain results similar to the disjoint representations, improving significantly in scenarios with fewer training data, partial sequences, or with pronounced class imbalance.

# Índice de figuras

# Índice de tablas

# Índice de contenidos

# Capítulo 1

# Introducción

Este trabajo es una exploración sobre la representación del estado de estudiantes mientras aprenden de las plataformas educativas en línea. Las representaciones obtenidas están destinadas a ser aplicadas para personalización automatizada, detección temprana de casos peligrosos o apoyo a especialistas en la toma de decisiones, entre otros. Presentamos una propuesta para inferir automáticamente una representación de los estudiantes usando redes neuronales artificiales. En particular, nuestro objetivo es aprender una proyección o *embedding* de las características de los estudiantes que sea posiblemente más adecuada que la representación original. Más precisamente, proponemos aprender de las características de los estudiantes y del material educativo, tales como ejercicios, lecciones y tareas, conjuntamente (*co-embedding*).

Este problema tiene varias características que lo hacen no trivial de abordar, principalmente porque el proceso de aprendizaje humano es un fenómeno complejo, con muchos factores que influyen en él, algunos de los cuales son sólo indirectamente observables. Las buenas representaciones capturan adecuadamente los cambios que se producen mientras los estudiantes aprenden, resaltando sólo los aspectos y variables importantes que influyen en el proceso. La importancia de tales aspectos está determinada por el objetivo final que se busca. Como esto sugiere, las características de una representación están ligadas a una tarea particular relacionada con el aprendizaje humano que necesita ser resuelta, por ejemplo, la detección y prevención de la deserción escolar, la recomendación de ítems, el rastreo de conocimiento (*knowledge tracing*), entre otros.

En la actualidad, los diseñadores e instructores de cursos interactúan con los estudiantes a través de las plataformas de aprendizaje, posiblemente supervisando a cientos de estudiantes concurrentemente. Estas nuevas formas de interacción con el alumnado afecta la forma en que se puede abordar la comunicación, la enseñanza y la supervisión. Sin embargo, la capacidad humana para el análisis de tantos datos puede no ser suficiente para aprovechar todo el potencial de las herramientas educativas en línea. A menudo, se utilizan procesos automáticos para

ayudar a los diseñadores de cursos a mejorar el contenido y la experiencia del estudiante. Los modelos de aprendizaje automático (*machine learning*), en particular, son herramientas que ayudan a comprender los datos generados y proveen un primer análisis útil para tomar medidas adicionales.

Proponemos aplicar técnicas basadas en datos que permitan descubrir patrones relevantes a partir de los registros de la actividad de los estudiantes grabados automáticamente por el software de las plataformas. Además de la complejidad de los fenómenos a modelar, los datos disponibles presentan sus propios desafíos: los registros suelen ser de bajo nivel y dependientes del curso particular.

A lo largo de este trabajo, nos centramos en cómo las redes neuronales artificiales pueden ser utilizadas para representar a los estudiantes y a los elementos del curso, así como las propiedades de las representaciones obtenidas. Durante este capítulo, presentaremos los conceptos clave necesarios para la definición del problema, el enfoque propuesto y la metodología seguida a lo largo de este trabajo. En la primera mitad describimos el área de Minería de Datos Educativos (EDM por sus siglas en inglés), sus desafíos y la importancia del modelado de estudiantes para los instructores. En la segunda mitad, planteamos el problema como una tarea de aprendizaje de representaciones utilizando enfoques de aprendizaje automático. Mencionamos varios aspectos a tener en cuenta a la hora de seleccionar posibles arquitecturas, y por ello hemos elegido redes neuronales profundas para esta tarea. Terminamos el capítulo con los objetivos, las aportaciones y la estructura de la tesis.

## 1.1 Representaciones en el aprendizaje en línea

El objetivo fundamental de la educación es mejorar el proceso de adquisición de conocimientos, junto con otras habilidades, por parte de los estudiantes, lo que puede lograrse a través de medios muy diversos. Desde el diseño curricular hasta los diferentes enfoques pedagógicos, los educadores de todas las áreas buscan formas de aumentar la efectividad y la eficacia del proceso de aprendizaje. En particular, el campo de la minería de datos educativos contribuye a este objetivo a través del análisis computacional de los datos.

Diferentes tipos de instituciones están incluyendo cada vez más herramientas digitales sobre las interacciones de aprendizaje. Como resultado, se generan nuevos tipos de interacciones entre alumnos, profesores y contenidos educativos. Estas interacciones tienen sus propias características y son diferentes de la experiencia tradicional en el aula, lo que hace que los educadores tengan que luchar para mantenerse al día con los nuevos desafíos.

Para entender lo que está sucediendo durante un curso, los instructores ahora confían en las interfaces de software en lugar de interactuar directamente con los estudiantes.

En particular, el creciente desarrollo de los Cursos Abiertos Masivos en Línea (MOOC) y la nueva disponibilidad de los Sistemas de Tutoría Inteligente (ITS) permiten a los investigadores recopilar volúmenes sustanciales de datos. Las tareas en EDM buscan utilizar los datos generados para diagnosticar y mejorar el contexto educativo. Además, hay casos en los que el análisis automático de los datos es la única forma de lograr estos objetivos. Algunos ejemplos de casos en los que la EDM ayuda a la mejora de los cursos son la devolución de respuestas automática, la prevención de la deserción escolar, la creación de perfiles de estudiantes y la personalización del contenido de los cursos de acuerdo con diferentes aspectos, como el nivel de dominio de las habilidades o los patrones de atención, entre otros.

Antes de ahondar en los detalles técnicos, es importante conceptualizar lo que significa analizar los datos automáticamente. Como hemos mencionado, el objetivo es ayudar a diferentes aspectos particulares de la actividad didáctica. Para ello, los investigadores necesitan modelar, con herramientas más o menos sofisticadas, los fenómenos expresados (y observados) en las interacciones registradas. La dificultad surge porque el aprendizaje humano es un proceso profundamente complejo, con causas ocultas y desconocidas que rigen el comportamiento de los estudiantes y el éxito o fracaso de los cursos. Desde este punto de vista, el análisis de los datos educativos puede entenderse como una búsqueda de estas causas ocultas, que explican o predicen los datos observados.

Para dar un ejemplo concreto, una tarea posible es generar recomendaciones de contenido para mejorar el rendimiento del estudiante. Las recomendaciones exitosas deben tener en cuenta i) el conocimiento que tiene el estudiante, ii) qué habilidades o componentes de conocimiento están involucrados en cada ejercicio iii) qué tipo de ejercicios hacen que este estudiante en particular aprenda mejor, por ejemplo, una lección fácil, contenido de video, o algo relacionado con sus intereses personales, y posiblemente iv) qué es lo que hace que el estudiante continúe estudiando, incluyendo variables relacionadas con el involucramiento cognitivo y emocional. Podemos considerar que esas son las causas latentes que determinan la eficacia del proceso de aprendizaje. La evidencia de estas causas latentes está presente en los datos, lo que permite encontrarlas con técnicas automáticas. También es posible modelar estos aspectos basándose en el conocimiento de los fenómenos provisto por expertos en el dominio. En la mayoría de las implementaciones hay una combinación diferente de ambos enfoques: basados en datos y con sesgos teóricos.

En este trabajo, nos centramos en modelar el proceso de aprendizaje desde un punto de vista general basado en datos, sin apuntar a ninguna tarea en particular. Esta es una diferencia importante con el trabajo anterior, que modela el proceso de aprendizaje ya sea en general pero teóricamente, o desde una perspectiva basada en datos pero para un conjunto de tareas similares. Proponemos una arquitectura que crea un modelo o representación para los estudiantes, elementos de los cursos, y modela explícitamente la relación entre ellos. Las representaciones

se obtienen a partir de patrones en los datos, específicamente de las interacciones registradas entre estudiantes y plataformas en línea.

Sin embargo, y como es de esperar, el éxito de muchos modelos automáticos basados en datos depende de la calidad y cantidad de los datos disponibles. Aunque estamos en medio de una "era de los grandes datos", la mayoría de las pequeñas y medianas empresas y organizaciones no tienen la infraestructura para soportar la recolección y posterior análisis de grandes conjuntos de datos. Parte de la propuesta de este trabajo es un modelo que potencialmente podría ser entrenado con menos información, a través de la inyección de conocimiento de dominio. La arquitectura modela la relación entre los estudiantes y los elementos del curso con una función definida. Potencialmente, los clasificadores aprenderán la tarea de predicción mejor y más rápido, considerando que no tienen que inferir esta relación a partir de los datos. Durante las secciones restantes de esta introducción desarrollaremos más a fondo por qué es importante abordar este problema y por qué es apropiado un enfoque basado en los datos que introduzca conocimiento experto.

### 1.1.1 Desafíos de la minería de datos educacionales

El área de la EDM ha ido creciendo de forma constante en la última década, incluyendo poco a poco más aprendizaje automático y enfoques de ciencia de datos. Esto se debe en parte a la reciente explosión de la recopilación de datos y las capacidades informáticas, que también afecta a muchas otras áreas.

Las herramientas didácticas generan miles de registros de la actividad de los alumnos, con un nivel de granularidad muy bajo. Por ejemplo, pueden registrar el tiempo de acceso de cada estudiante al contenido del curso, la duración del acceso o el resultado de las pruebas y evaluaciones. En consecuencia, los educadores tienen acceso a una descripción completa de los eventos que ocurrieron durante el proceso de aprendizaje del estudiante.

No obstante, es un desafío interpretar y utilizar estos registros de bajo nivel de actividad estudiantil (también conocidos como *logs*) en tareas como la predicción del rendimiento de los estudiantes, que generalmente involucran conceptos de alto nivel que se aproximan a causas ocultas, como las habilidades o el nivel de compromiso. Debido al gran tamaño de los conjuntos de datos y a la falta de abstracción en la información, en la mayoría de los casos no es posible obtener conclusiones útiles a partir de un análisis manual de los datos. En cambio, se necesitan herramientas analíticas muy potentes para procesar los datos. Se utilizan procesos automáticos para agregar los datos y desentrañar los patrones en las interacciones de los estudiantes con los contenidos del curso, que de otro modo se oscurecen en los registros sin procesar. Un claro ejemplo de este escenario es el *DataShop* de la Universidad de Carnegie Mellon[1], que actúa como

[1] `https://pslcdatashop.web.cmu.edu/about/`

repositorio de conjuntos de datos y también proporciona herramientas de análisis de estándar.

Hay otro factor a considerar. Una de las principales ventajas de los entornos de aprendizaje en línea es su potencial para llegar a miles de estudiantes, pero a medida que aumenta el número de beneficiarios, también aumenta su heterogeneidad. Las aulas virtuales están llenas de alumnos con antecedentes y expectativas muy diferentes. No seguirán el mismo camino, no obtendrán los mismos resultados ni tendrán los mismos objetivos en relación con el curso. Frente a este escenario, los instructores necesitan dirigirse a diferentes tipos de estudiantes e identificar cómo afectan a las interacciones registradas. Afortunadamente, los conjuntos de datos son tan grandes que se pueden utilizar enfoques basados en datos para inferir patrones y tener en cuenta esta diversidad. Un ejemplo son los modelos de aprendizaje automático, que pueden modelar tareas complejas como la evaluación del nivel de habilidades y conocimientos, la personalización y recomendación, la generación automática de respuestas, el análisis de redes sociales, etc.

Estos enfoques de aprendizaje automático no se basan en reglas estáticas definidas por expertos en la materia. En su lugar, descubren patrones automáticamente a partir de los ejemplos dados, intentando optimizar una tarea determinada. En otras palabras, siguen un paradigma basado en datos. No están limitados por el conocimiento previo de los instructores e investigadores, y pueden ser utilizados para descubrir causas subyacentes previamente desconocidas que describen los datos.

Nuestra propuesta es un modelo de aprendizaje automático que puede ser adaptado y entrenado para predecir diferentes tareas, sin hacer suposiciones sobre el tipo de curso, las interacciones o incluso los estudiantes. El enfoque se centra en modelos capaces de manejar conjuntos de datos con registros de bajo nivel. Esto se debe a dos razones: por un lado, estos datos ya están disponibles en la mayoría de los sistemas. Por otro lado, este tipo de registros pueden ser los únicos disponibles. Para obtener información de más alto nivel, como las lecciones correlativas para cada contenido, un experto humano debe anotar manualmente esta información, lo que implica tiempo y dinero.

El objetivo de utilizar un enfoque basado en datos es obtener representaciones de los estudiantes y de los elementos del curso que no se ven afectados por preconcepciones humanas que pueden no tener en cuenta toda la variabilidad de los datos. Un caso claro que explicaremos en detalle son las lecciones de etiquetado con las habilidades (*skills*) que enseña. La primera limitación es que los elementos del curso se modelan utilizando únicamente información relacionada con las habilidades, dejando fuera conceptos secundarios como la dificultad intrínseca del material. En segundo lugar, el instructor define qué habilidades se utilizan y las adapta a una tarea o aspecto específico del entorno de aprendizaje. Por ejemplo, un problema de matemáticas se centrará en conceptos matemáticos, mientras que puede ser posible necesitar habilidades de lectura para comprender el texto.

Este tipo de sesgo puede evitarse si permitimos que el modelo obtenga la mejor representación de los estudiantes y de los elementos del curso por sí mismo, optimizándolo para una tarea determinada.

Las arquitecturas propuestas, las redes neuronales, son capaces de manejar grandes cantidades de datos, de aprender abstracciones sobre datos de bajo nivel y también son lo suficientemente expresivas como para manejar múltiples tipos de estudiantes en un solo curso, modelando heterogeneidades por separado.

## 1.2 Aprender un modelo de estudiante

En su esencia, muchas tareas de EDM tratan de encontrar un modelo adecuado de los estudiantes basado en su actividad, tanto para explicar su comportamiento, como para los sistemas predictivos.

En términos formales, el modelado de un estudiante consiste en encontrar una representación matemática (en nuestro caso, un vector) que capte los aspectos deseados para la tarea, incluyendo también las relaciones con otros sujetos y objetos del entorno educativo. El área que estudia las técnicas para construir un vector que representa al sujeto se llama Aprendizaje de Representaciones. En este trabajo, usaremos el término representación para referirnos a lo que otros trabajos pueden llamar "modelo del estudiante".

La representación del sujeto en un espacio matemático es útil porque puede ser manipulada con métodos como los clasificadores de aprendizaje automático. Para representar a un estudiante como vector, la información debe ser convertida de la entrada bruta a una serie de números que mantienen ciertas propiedades. En este contexto, las representaciones también se denominan *características*, *vectores de características*, o *embeddings*. Por ejemplo, se podría representar a un estudiante por un vector compuesto por las puntuaciones obtenidas en los exámenes. En este vector, las puntuaciones de cada examen individual serían las características. Esta representación condensa el desempeño de los estudiantes y representa su conocimiento medido por los exámenes. Se puede utilizar para estimar el rendimiento futuro, evaluar la dificultad de un curso, etc.

En muchos casos, la representación que se busca está determinada por la tarea específica que se necesita mejorar, pero no siempre está totalmente moldeada por ella. Por ejemplo, digamos que estamos buscando una representación de los estudiantes que describa su nivel de dominio de ciertas habilidades. No es fácil determinar los factores que afectan el nivel de dominio incluso para los expertos en dominios, y mucho menos para un sistema automático. Desarrollaremos más a fondo las dificultades de este proceso en la Sección 2.1.

En lugar de definir manualmente qué factores se incluye en una representación que captura ciertas propiedades, utilizamos una tarea relacionada para obtener una representación que, con suerte, capture las causas latentes que también son relevantes para la tarea original. Esta tarea relacionada, llamada *tarea de pretexto*, es útil en un sentido práctico para obtener la representación, pero no el objetivo final en sí. Por ejemplo, para la tarea de estimar el conocimiento adquirido por un estudiante, la tarea de pretexto puede ser la predicción del desempeño futuro del estudiante en determinadas pruebas. Es posible obtener datos etiquetados para entrenar un modelo para esta tarea de predicción, utilizando las puntuaciones de los estudiantes en exámenes. Como subproducto, esta tarea de predicción genera una representación donde el conocimiento adquirido está implícitamente representado.

Algunos métodos de aprendizaje de la representación, y las redes neuronales en particular, producen vectores que no pueden ser interpretados directamente. Sin embargo, la interpretación directa no es la única aplicación útil de una representación. Pueden aplicarse a otras tareas de interés para las que no se dispone de datos que permitan elaborar un modelo de estudiante. En el caso de la estimación de los conocimientos del estudiante, la representación obtenida podría utilizarse para recomendar una vía de aprendizaje para el estudiante. Otro ejemplo es el trabajo de Chen et al. [2018], donde los datos de navegación de los estudiantes que ven lecciones en vídeo se utilizan para formar un modelo y para estimar las siguientes acciones del estudiante (tarea de pretexto). A partir de esta información, los autores también construyen un gráfico de los requisitos previos de la lección.

Durante este trabajo, utilizaremos dos tareas de pretexto, el Rastreo del Conocimiento y la predicción de deserción escolar, para evaluar la representación de los estudiantes producida por el modelo propuesto desde diferentes puntos de vista.

## 1.3 El paradigma del aprendizaje automático

Como se mencionó anteriormente, los enfoques de aprendizaje automático aprenden un modelo predictivo sin depender de reglas hechas a mano. En cambio, infieren modelos a partir de una serie de ejemplos dados. En uno de los entornos clásicos, llamado aprendizaje supervisado, estos ejemplos han sido previamente etiquetados por expertos, asociándolos manualmente al resultado deseado. El objetivo del modelo es aprender a predecir la etiqueta asignada.

Por ejemplo, en la tarea de predicción de deserción, el objetivo es predecir si el estudiante dejará de interactuar con la plataforma (deserción) antes de que se logre un final exitoso. Cada ejemplo individual podría ser la secuencia completa de interacciones entre el estudiante y la plataforma de aprendizaje. La secuencia está asociada a una etiqueta binaria que indica si el alumno ha abandonado el curso o no. Un algoritmo de aprendizaje automático puede procesar

todo el conjunto de secuencias del estudiante, con sus etiquetas correspondientes, y aprender a predecir si un estudiante abandonará el curso al ver una secuencia de sus interacciones con la plataforma de aprendizaje. Este proceso se llama *entrenamiento*.

En resumen, el entrenamiento optimiza los parámetros del modelo matemático tratando de minimizar el error de clasificación. Es un proceso iterativo que busca en el espacio de los modelos posibles, hasta que converge a una solución o encuentra un modelo con un error suficientemente pequeño.

Una vez que el modelo ha sido entrenado, puede ser usado para clasificar nuevas secuencias que no han sido vistas previamente. Esta siguiente etapa también se conoce como *predicción*. Para que el modelo funcione como se espera, los nuevos ejemplos a clasificar deben ser lo más similares posibles a los utilizados durante la fase de entrenamiento.

Hay muchos tipos posibles de tareas a resolver mediante el aprendizaje automático, como la regresión, la traducción, la detección de anomalías, etc. Nos enfocaremos en la tarea de clasificación durante este trabajo, y usaremos los términos predicción y clasificación indistintamente.

### 1.3.1 La promesa de un aprendizaje profundo

Por muy prometedores que sean los enfoques de aprendizaje automático, todos los modelos matemáticos tienen limitaciones a la hora de representar fenómenos complejos y, en particular, humanos. Como se explica en Goodfellow et al. [2016], aprender a generalizar a ejemplos no vistos sólo a partir de una muestra parece contradecir las leyes de la lógica. No es lógicamente válido inferir reglas generales a partir de un conjunto limitado de ejemplos. Una forma de hacer frente a este problema es proporcionar únicamente clasificaciones con una probabilidad superior a un determinado umbral. Aún así, los modelos de aprendizaje automático suelen tener algún grado de error en sus predicciones.

Esto es relevante porque necesitamos enfocarnos en modelos que respondan a las preguntas importantes para el área, en lugar de tratar de obtener un modelo óptimo. Muchas veces este problema es subestimado en la búsqueda de mejores resultados de precisión.

La pregunta que sigue es, entonces, ¿qué hace que un modelo sea más adecuado que otros? La capacidad de un modelo para resolver un determinado problema está muy relacionada con:

1. La complejidad del problema.

2. El nivel de abstracción que podemos capturar con el modelo.

Estos factores no son exactamente equivalentes. Es posible que tengamos un problema muy complejo, pero aún así sólo intentemos captar aspectos superficiales del mismo. Tomemos, por ejemplo, la predicción del sentimiento del autor al escribir tweets. La semántica del lenguaje natural aún no ha sido completamente entendida, y menos formalmente modelada. Al expresar acuerdo o sentimiento, encontramos usos comunes de ironía y humor, que añaden una nueva dimensión al problema. Además, uno puede tener un sentimiento positivo hacia un aspecto del tema y un sentimiento negativo hacia otro. Sin embargo, para muchas aplicaciones, basta con detectar sólo la emoción general evocada por el texto. La tarea completa de análisis de sentimientos se simplifica en exceso a "positivo, negativo o neutral", con una comprensión superficial de lo que significa positivo y negativo. En este caso, se pueden utilizar modelos más sencillos y tienen un rendimiento aceptable.

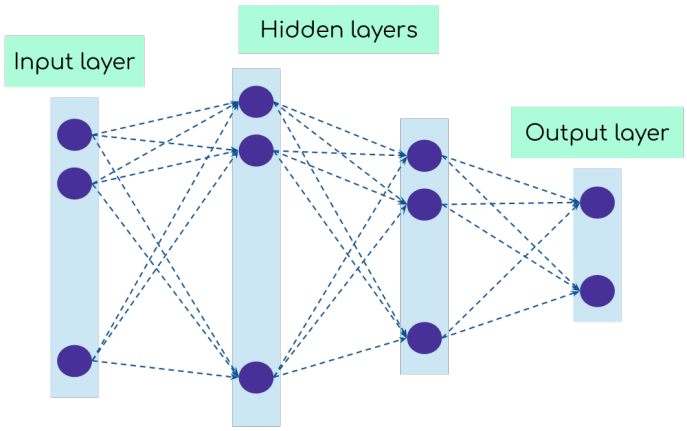
Las tareas de EDM son también fenómenos humanos complejos con causas ocultas. Pero a diferencia del caso anterior, en la mayoría de las aplicaciones no basta con un análisis superficial del proceso de aprendizaje. Esto significa que requieren un modelado de los estudiantes que capte las causas profundas y ocultas.

Si un problema requiere una comprensión completa de los factores que intervienen en un fenómeno dado, los modelos simples pueden no tener la capacidad de captarlos. Una forma de intentar capturar la complejidad es construir una representación inicial del problema que ya codifique toda la información necesaria, con bajos niveles de ruido o datos irrelevantes. Es más fácil inferir un modelo de tal representación, porque las causas latentes han sido (al menos parcialmente) *desenredadas* durante la construcción de la representación. A partir de esta representación, el modelo puede ser simple y seguir funcionando bien. Con este enfoque, la representación se convierte en la clave para el éxito o el fracaso del modelo. En este caso, el esfuerzo manual requerido para desarrollar reglas que describan algún fenómeno se transfiere al desarrollo de características que capturen el mismo fenómeno, el proceso conocido como *ingeniería de características*.

La ingeniería de características tiene un alto costo porque depende del conocimiento del dominio para entender qué información es relevante y qué es ruido. En cierto modo, volvemos a caer en el problema de causas predefinidas, en lugar de causas descubiertas por el modelo, porque en este enfoque las características son hechas a mano por expertos humanos.

Si no queremos depender tanto de la representación inicial de los datos, debemos utilizar un modelo que pueda inferir las causas latentes a partir de una representación más simple. Los modelos de aprendizaje profundo, de los cuales las redes neurales profundas son los ejemplos más frecuentes, tienen exactamente esta propiedad. En capítulos siguientes explicaremos en detalle los desafíos de implementación y entrenamiento de las redes neuronales. Antes de eso, durante esta introducción presentaremos la intuición de los modelos de aprendizaje profundo como una técnica de aprendizaje de representaciones.

Figure 1.1: Arquitectura de una red neuronal *feed forward*




En la figura 1.1 mostramos la arquitectura básica de una red neuronal, que se caracterizan por tener varias capas de computación. En la figura podemos ver una capa de entrada, dos capas ocultas y una capa de salida. El modelo se utiliza para una tarea determinada, por ejemplo, para clasificar la respuesta a una pregunta como correcta o incorrecta. La entrada, en este ejemplo la respuesta a la pregunta, debe traducirse primero en una representación vectorial. La representación se alimenta a la red utilizando la capa de entrada. Luego, se propaga a través de la red hasta la última capa, que proporciona la respuesta a la tarea. En las tareas de clasificación, la capa de salida devuelve un valor para cada clase con la probabilidad de que la entrada pertenezca a esa clase. En este ejemplo, sólo hay dos clases posibles, respuestas correctas o incorrectas.

En su viaje desde la capa de entrada a la de salida, el vector con la representación inicial es procesado por varias capas ocultas. Éstas se conectan a través de matrices de peso (*weights*) y funciones no lineales. Las conexiones se representan en la figura 1.1 utilizando líneas de puntos. Cada capa contiene una transformación del vector en la capa anterior, es decir, una representación transformada del objeto. Las capas pueden no tener el mismo tamaño.

Con estas modificaciones consecutivas a la entrada, la red neuronal está creando internamente nuevas representaciones del objeto. Estas representaciones están optimizadas para desentrañar las causas de las variaciones que afectan a la tarea en cuestión. El proceso continúa hasta que la representación en la capa anterior a la salida es tan cercana a la tarea de clasificación, que una sola capa es suficiente para realizar la clasificación con un error mínimo. Además, la red utiliza operaciones no lineales para obtener cada nueva representación, y es capaz de modelar funciones arbitrariamente complejas.

Cuando entendemos los modelos neurales como *desentrañadores de causas* a través de la generación de representaciones, es más obvio por qué se desempeñan tan bien en tareas con muchos factores latentes, como las del área de EDM. Estos modelos, en lugar de requerir que un experto

en la materia proporcione una representación con todos los factores que podrían contribuir a la tarea, construyen esas representaciones a partir de la inicial por sí mismos. Aunque es cierto que es necesaria una representación inicial con suficiente información, el éxito o fracaso del modelo depende menos de la calidad de la representación, de la cantidad de ruido presente, etc.

En este trabajo, proponemos utilizar redes neuronales para obtener representaciones de estudiantes. Estas representaciones suelen llamarse *embeddings*, porque proyectan vectores de un espacio a otro. Es decir, transforman vectores de una capa de la red a la siguiente.

Utilizaremos un tipo especial de red, una red neuronal recurrente, diseñada para procesar secuencias de eventos, para obtener la representación. Además, presentamos una arquitectura novedosa que también modela elementos del curso en el mismo espacio que los estudiantes. Esto permite una representación conjunta de los estudiantes y los elementos del curso, lo que es potencialmente más adecuado que modelar cada uno de ellos por separado.

## 1.4 Objetivo y contribuciones

Luego de introducir los conceptos presentados anteriormente, podemos describir el objetivo principal de este trabajo y dividirlo en diferentes sub-objetivos.

**Objetivo principal** Estudiar el impacto del aprendizaje de representaciones utilizando redes neuronales en el modelado de interacciones de datos educativos desde plataformas en línea, en dos aplicaciones diferentes: rastreo de conocimiento y predicción de deserción escolar.

Los sub-objetivos identificados son:

1. Evaluar diferentes representaciones obtenidas mediante la modificación de la estructura básica de un clasificador neural en diferentes tareas de pretexto.
2. Medir el impacto de los embeddings conjuntos de estudiantes y elementos del curso en las aplicaciones mencionadas anteriormente.
3. Evaluar hasta qué punto los embeddings neurales obtenidos pueden ser interpretados.

Las contribuciones de este trabajo son:

1. La propuesta de una arquitectura general que modela los elementos del curso y los estudiantes en el mismo espacio (*co-embeddings*), de forma agnóstica el tipo de datos presentes en el conjunto de datos y la aplicación a la que se dirige.

2. La evaluación de las incrustaciones conjuntas en la tarea de rastreo de conocimiento.

3. La evaluación de los co-embeddings en la tarea de predicción de la deserción escolar, incluyendo dos escenarios posibles: usar la secuencia completa de interacciones y usar sólo información parcial (detección temprana).

4. Una exploración de cómo los embeddings altamente dimensionales pueden ser visualizados y posiblemente interpretados.

5. Un esquema teórico de cómo la predicción de la deserción escolar temprana puede ser interpretada como una tarea de Aprendizaje por Refuerzo. En particular, la misma arquitectura de co-embeddings también puede ser beneficiosa en este tipo de escenario.

El resto de esta tesis está organizada en los siguientes capítulos:

En el capítulo 2 presentamos con más detalle el problema del modelado del estudiante, y las dificultades inherentes a la tarea, con los conjuntos de datos disponibles. Proporcionamos una formalización de las tareas de pretexto para obtener representaciones de estudiantes. Finalmente, definimos las preguntas de investigación que enmarcarán el resto del trabajo.

En el capítulo 3 resumimos el trabajo relacionado en el área y presentamos las propuestas más avanzadas para ambas tareas de pretexto. En el caso del rastreo de conocimientos, comparamos diferentes enfoques aplicados durante la última década. Expandimos una discusión común sobre el aumento del rendimiento con modelos de aprendizaje profundo a costa de sacrificar la interpretabilidad de los resultados. En el caso de la predicción de la deserción escolar, presentamos diferentes definiciones de la deserción escolar y posibles escenarios en los que la predicción y la prevención pueden ser útiles.

En el capítulo 4 describimos en detalle la arquitectura utilizada para obtener embeddings para estudiantes y elementos del curso. Se presentan y analizan varias variaciones. Finalmente detallamos una variante del modelo general para cada tarea de pretexto.

En el capítulo 5 describimos los desafíos de entrenar una arquitectura neural y cómo los diferentes hiperparámetros afectan al modelo. En la segunda mitad del capítulo, describimos el entorno experimental utilizado para evaluar el rendimiento de los modelos. Esto incluye los detalles de la implementación y las métricas propuestas para llevar a cabo la evaluación.

Los capítulos 6 y 7 siguen una estructura similar, describiendo los resultados obtenidos y los ajustes experimentales particulares para las tareas de Rastreo de conocimientos y predicción de abandono, respectivamente. Analizamos las tendencias en el desempeño y describimos conclusiones preliminares.

En el capítulo 8 exploramos posibles visualizaciones de las representaciones obtenidas. Nos centramos en qué tipo de información es capturada por el espacio latente utilizando algoritmos de reducción a dos dimensiones.

Terminamos este trabajo con el capítulo 9, que incluye las conclusiones principales, resume las contribuciones del trabajo, y enumera las posibles vías de investigación futura.

Finalmente, en el Apéndice A se describe un diseño para la aplicación de los algoritmos de Aprendizaje de Refuerzo Profundo a la tarea de predicción de la deserción escolar temprana, y se explica cómo los embeddings utilizados para el enfoque anterior de aprendizaje supervisado pueden ser fácilmente adaptados a este nuevo entorno.

# Capítulo 2

# Descripción del problema

En este capítulo, profundizaremos en los desafíos a los que nos enfrentamos al trabajar con tareas de EDM. En primer lugar, comenzaremos por analizar el proceso de aprendizaje humano y los diferentes factores latentes que pueden determinar su eficacia. Comprender los fenómenos que estamos tratando de predecir es importante para explicar el comportamiento de cualquier modelo estadístico o automático.

En una segunda sección, describiremos los diferentes entornos educativos en línea donde los estudiantes pueden aprender. Los modelos automáticos deben diseñarse teniendo en cuenta la diversidad de plataformas y los datos que generan, así como la heterogeneidad de los estudiantes. Analizaremos qué aspectos del alumno es necesario modelar para cada tarea y las diferentes alternativas de representación. Específicamente, analizaremos cómo el aprendizaje de una representación para los estudiantes puede integrarse en diferentes tareas y el impacto que puede tener en estos modelos automáticos.

En las últimas secciones del capítulo, presentamos nuestra propuesta de optimizar representaciones conjuntas para los estudiantes y los elementos del curso. Explicaremos por qué es importante y los posibles beneficios de este enfoque. Finalmente, proporcionaremos una definición formal del problema y plantearemos las preguntas de investigación que guían el resto del trabajo.

## 2.1 Proceso de aprendizaje

Cuando se abordan tareas de EDM, es común empezar por describir la información disponible registrada por una plataforma y construir a partir de ahí un modelo que pueda explicar o predecir algún comportamiento en particular. Sin embargo, describir a un estudiante sólo a través de sus interacciones con un sistema de aprendizaje es una simplificación excesiva del

proceso de aprendizaje. Otros factores, incluyendo pero no limitados a las características del estudiante y el contenido del curso, determinarán el éxito de este proceso.

Muchas teorías de aprendizaje han sido propuestas desde diferentes escuelas psicológicas y sociológicas, tratando de explicar cómo los humanos incorporan y retienen el conocimiento. Cuando se trata de construir un sistema automático, es importante entender el tipo de fenómenos que estamos tratando de predecir. Esto nos permitirá determinar cuáles de los factores latentes se pueden aprender del conjunto de datos y cuáles permanecerán ocultos.

Las descripciones del proceso de aprendizaje son tan antiguas como las de Sócrates, que entendía al ser humano como una entidad llena de conocimiento que debía ser extraída a través del diálogo, en un proceso llamado mayeútica.

La mayoría de las teorías, desde la transferencia del aprendizaje a los modelos cognitivos, diferencian algún concepto de aprendizaje *profundo* versus *superficial*, donde los conceptos no son plenamente comprendidos o instrumentalizados por el alumno. Esto apoya nuestra intuición de que el aprendizaje es un fenómeno muy complejo, difícil de definir y aún más difícil de modelar formalmente.

El conocimiento, y el aprendizaje en general, fue visto inicialmente como un producto medido por los cambios en el comportamiento. El conductismo y el cognitivismo sostienen que el conocimiento es fijo y finito. "El conocimiento es algo que el maestro ha dominado, y que los estudiantes deben ahora dominar de manera similar replicando el conocimiento del maestro" [Harasim, 2011, p. 13]. Las teorías más recientes, como el aprendizaje experimental de David A. Kolb, entienden el aprendizaje como un proceso. En particular, Kolb propone un ciclo de cuatro etapas de aprendizaje y cuatro estilos de aprendizaje separados [Kolb, 1984].

Podemos nombrar algunos de los factores que se cree que influyen en la adquisición de conocimientos. El psicólogo educativo David Ausubel propuso el concepto de "aprendizaje significativo", en contraste con otros tipos de aprendizaje como la memorización. Bajo esta teoría, "el factor más importante que influye en el aprendizaje es lo que el estudiante ya sabe". El aprendizaje significativo, que implica una mayor retención, ocurre cuando los estudiantes construyen nuevos conceptos a partir de conceptos preexistentes, y pueden integrarlos en un sistema de conocimiento consistente [Ballester Vallori, 2014, Ausubel, 1963].

El estudiante como agente es también un factor determinante del aprendizaje. La teoría de Howard Gardner de la inteligencia múltiple (1993) propone que no hay una sola capacidad de aprendizaje, sino muchas de ellas. Una inteligencia se define como "un potencial bio-psicológico para procesar información que puede ser activada en un entorno cultural para resolver problemas o crear productos de valor en una cultura". En consecuencia, un estudiante no puede ser modelado de acuerdo a un solo factor y se espera que se desempeñe de manera diferente para diversas tareas.

Como se dijo anteriormente, Kolb también distingue diferentes tipos de estudiantes. Esta categorización determina cómo la persona construye nuevos conocimientos y cuál es su formato preferido para el contenido educativo. Un estudiante puede beneficiarse de explicaciones detalladas de los conceptos en video, mientras que otros pueden requerir ejercicios para desarrollar un razonamiento hipotético-deductivo.

Los enfoques modernos de la neurociencia explican el proceso de aprendizaje desde un punto de vista aún más centrado en el sujeto, midiendo las respuestas neurológicas a diferentes situaciones. Las principales teorías hablan de la motivación intrínseca, entendida como la tendencia espontánea de las personas a ser curiosas e interesadas, como el factor principal detrás del aprendizaje efectivo. Aunque la motivación intrínseca es una función psicológica, también depende de los respaldos ambientales para las necesidades psicológicas básicas, especialmente las de competencia y autonomía. Los instructores, en las aulas o a través de contenidos online, deben despertar estas motivaciones internas para lograr un verdadero y profundo aprendizaje de los conceptos.

Harasim [2011] se adentra en el cambio de las teorías del aprendizaje en el siglo XXI con la aparición del Aprendizaje Colaborativo en Línea (OCL por sus siglas en inglés) como una forma de adaptarse a las nuevas tecnologías. Estas nuevas corrientes, similares al constructivismo, identifican al alumno como un agente central y activo. Sin embargo, sitúa el aprendizaje activo "dentro de un proceso de desarrollo social y conceptual basado en el discurso de conocimiento".

Adicionalemente a las causas latentes que afectan el aprendizaje del estudiante, el desarrollo del contenido del curso también está determinado por múltiples factores. La transposición didáctica se refiere al proceso de transformar el conocimiento científico en conocimiento realmente enseñado y recursos pedagógicos. Teorías posteriores, específicamente la obra de Pierre Clément, afirman que la transposición didáctica implica otros elementos además del conocimiento, como los valores y las prácticas sociales [Amador et al., 2016]. Como resultado, existe una información contextual involucrada en el desarrollo del material del curso, que no puede ser interpretada de la misma manera por personas ajenas a dicho contexto.

Asimismo, la teoría del texto de Bernstein se centra en el aspecto práctico de las relaciones pedagógicas, y cómo se ve afectada por el contexto social. Los diferentes grupos generan diferentes modalidades de comunicación, con diferente efectividad dependiendo de los valores de la escuela, los modos de práctica y las relaciones con sus comunidades Bernstein [2000]. Las comunicaciones observadas en los principales contextos educativos son radicalmente diferentes de las observadas en la educación primaria y secundaria. De ello se deduce que el proceso de aprendizaje se verá afectado por el grado de alineación entre el contexto del alumno, el contexto del instructor y los objetivos del curso.

Desde todas estas perspectivas, podemos concluir que el proceso de aprendizaje no es nada

sencillo. Los factores individuales, contextuales y sociales intervienen en el alumno, el instructor y las instituciones involucradas. Cuando se abordan los fenómenos desde una perspectiva estadística y computacional, muchos de estos factores no serán capturados, tanto porque no están registrados en el conjunto de datos, como porque el sistema carece de las capacidades de representación para modelar tal complejidad.

## 2.2 Aplicaciones de la minería de datos educacionales

Dependiendo de las intenciones de los diseñadores de cursos, los sistemas educativos pueden beneficiarse de diversos tipos de adaptaciones. Como hemos mencionado anteriormente, existen muchos aspectos de los entornos de aprendizaje en línea que pueden mejorar utilizando intervenciones automáticas: predicción de deserción escolar y su posterior prevención, recomendación de rutas o contenidos, y retroalimentación automática, entre otros.

Para cada escenario, hay diferentes aspectos de los estudiantes que pueden ser modelados para ayudar a mejorar la experiencia de aprendizaje. En este sentido, los autores Chrysafiadi and Virvou [2013] hacen una extensa revisión del tema, nombrando "nivel de conocimiento, habilidades, preferencias y estilos de aprendizaje, errores y conceptos erróneos, motivación, características afectivas como emociones y sentimientos, aspectos cognitivos como la memoria, la atención, la resolución, la toma de decisiones y las capacidad de análisis, el pensamiento crítico y la habilidad de comunicación, y aspectos meta-cognitivos como la autorregulación, la auto-explicación, la auto-evaluación y la autogestión"'.

Una de las características más populares para modelar a través del aprendizaje automático es el conocimiento adquirido por un estudiante. Esta tarea se llama *Knoweledge Tracing*. El nivel de competencia en un conjunto fijo de habilidades se estima a partir de sus interacciones con los elementos del curso y los resultados de dichas interacciones. Sin embargo, existe un creciente interés en predecir otros aspectos del comportamiento de los estudiantes, como el compromiso y la probabilidad de deserción escolar, como se vio en la competencia KDDCup 2015 [1].

También hay otros aspectos importantes a tener en cuenta a la hora de modelar a los estudiantes: el método utilizado y los datos disponibles, así como la compatibilidad entre ambos. Trabajaremos con modelos de aprendizaje profundo, que requieren grandes cantidades de datos para obtener predicciones fiables. A pesar de esta deficiencia, las técnicas de aprendizaje profundo son especialmente adecuadas para modelar datos de bajo nivel y no requieren la adición de información etiquetada por el ser humano.

En la siguiente sección analizaremos los retos de varias tareas de modelado bajo estas circunstancias, teniendo en cuenta el tipo de datos para cada una de ellas. Finalmente, presentaremos

[1] `https://biendata.com/competition/kddcup2015/`

una formalización general del problema de modelado.

## 2.3 La diversidad en los datos: MOOC vs ITS

Podemos distinguir dos tipos principales de sistemas en EDM en cuanto a la organización del material didáctico y la plataforma donde los estudiantes pueden acceder a él: Cursos Abiertos Masivos en Línea (MOOCs) y Sistemas de Tutoría Inteligente (ITSs). Los ITSs son entornos de aprendizaje digitales destinados a proporcionar retroalimentación automática en el momento de la interacción. La información está fuertemente estructurada por expertos, generalmente en lecciones cortas complementadas con muchos ejercicios. La trayectoria de los estudiantes a través del material está determinada por un algoritmo que estima el conocimiento del estudiante y sugiere la actividad más conveniente a realizar a continuación. Para calcular estas trayectorias, cada elemento del curso debe estar completamente etiquetado con las habilidades que se supone que debe enseñar u otros aspectos que ayuden al sistema a elegir el siguiente.

Los registros generados por los ITSs usualmente contienen los intentos de cada estudiante para resolver ejercicios, conteniendo información como tiempo, resultado, número de intentos previos y descripción de los ejercicios.

Los MOOCs, por otro lado, están inspirados en los cursos tradicionales en el aula. El material se presenta en su totalidad al estudiante o siguiendo un horario predefinido, junto con un orden sugerido a seguir. El contenido suele estar más centrado en lecciones, a veces sin ejercicios. Una manera común de calificar es utilizar calificaciones asignadas por los compañeros, sin la intervención de un instructor. La falta de personalización en los MOOCs se compensa con un menor coste y un mayor alcance, teniendo a veces millones de estudiantes.

Otra característica importante de los MOOC nace precisamente de su precio generalmente más bajo y una mayor disponibilidad: los estudiantes proceden de orígenes muy diferentes y también tienen expectativas distintas. Kizilcec et al. [2013] analiza los antecedentes, la demografía y los caminos comunes de aprendizaje de los estudiantes en los MOOCs. Entre otras cosas, los autores descubren que muchos estudiantes no están interesados en completar las actividades del curso, pero siguen todas las lecciones.

En general, el contenido es tan diverso que resulta problemático descubrir patrones y sistematizarlos de manera uniforme. Incluso los cursos creados en una sola plataforma pueden ser muy diferentes. Por ejemplo, tendrán un conjunto totalmente desarticulado de lecciones y ejercicios, pueden ser diseñados para diferentes niveles de compromiso y carga de trabajo, las unidades pueden ser independientes o altamente correlacionadas, entre otras. Estas diferencias son aún mayores entre las diferentes plataformas, que a su vez pueden recoger señales diferentes o proporcionar más herramientas para los diseñadores de cursos.

La heterogeneidad es aún más aguda cuando se comparan los datos de los MOOCs con los de los ITSs, donde el entorno de aprendizaje y la organización de los materiales es significativamente diferente. Por ejemplo, el contenido utilizado por los ITS es seleccionado y etiquetado con la meta-información necesaria para las recomendaciones producidas por el sistema, lo que implica una anotación humana experta. El formato estático de los MOOCs hace que su creación sea menos costosa y, por lo general, no incluyen una meta-información tan detallada.

Esperamos que una propuesta general basada en arquitecturas profundas sea lo suficientemente flexible como para tratar diferentes tipos de datos y conservar las propiedades básicas. Proponemos utilizar estos embeddings para evitar la necesidad de costosas etiquetas de ejercicios, lo que hace que el método sea aplicable a los contenidos de los cursos que se desarrollan con menos recursos.

## 2.4 Modelado conjunto de estudiantes y elementos del curso

Como hemos explicado en el capítulo anterior, nuestro objetivo es obtener representaciones que pongan en manifiesto los principales factores que intervienen en la experiencia de aprendizaje humano. Para abordar el reto de modelar a los estudiantes con datos de bajo nivel no etiquetados, proponemos no sólo modelar las acciones de los mismos, sino también los elementos del curso. Ambos tipos de elementos pueden ser representados a través de embeddings neurales, ya que proporcionan una capa de abstracción y agregación sobre datos de bajo nivel.

En este contexto, la palabra *embedding* puede utilizarse para denotar la función que transforma un objeto en un vector, así como el vector obtenido en sí mismo. Se usará con ambos significados a lo largo de este trabajo.

Nos inspiramos en el trabajo de Reddy et al. [2016], donde se presenta un método para generar representaciones de estudiantes en un "espacio de habilidades latente". Los autores interpretan el espacio de habilidades latente como el nivel de conocimiento del estudiante sobre cada habilidad, donde el conjunto de habilidades se descubre durante el entrenamiento del modelo. Por otra parte, este trabajo propone encontrar representaciones de los elementos del curso utilizando el mismo espacio de habilidades latente. Como este enfoque se basa en estructuras de cursos similares a las de MOOC, tiene en cuenta tanto los tipos de cursos calificados (evaluaciones) como los no calificados (lecciones).

El uso de un espacio de embedding conjunto en la propuesta de Reddy et al. permite modelar explícitamente las relaciones entre los estudiantes y el contenido, en lugar de dejar que el modelo lo descubra automáticamente. El modelo está construido con operaciones explícitas

que combinan ambos embeddings, reflejando cómo se entiende el proceso de aprendizaje. Por ejemplo, si un estudiante ya ha dominado la habilidad enseñada en una lección, esperamos que la ganancia de aprendizaje de esa lección sea pequeña. Esto se representa en el espacio latente como el embedding del estudiante estando "lejos" del embedding de la lección. El caso contrario, en el que el estudiante no ha visto nada relacionado con el contenido de la lección, puede conducir a la misma pobre ganancia de aprendizaje. El estudiante no tiene conocimientos previos para entender la nueva lección. Podemos modelar esta situación también como la integración del estudiante estando "lejos" del embedding de la lección, pero en la dirección opuesta. Entonces, si la distancia entre el estudiante y el elemento del curso es grande, la contribución de la interacción al conocimiento del estudiante es menor.

Interpretar explícitamente la distancia entre los embeddings inyecta conocimiento sobre los fenómenos que se pretende descubrir en el modelo, lo que puede tener un impacto significativo en tareas con menos ejemplos disponibles.

Esperamos que este espacio latente pueda generalizarse no sólo a las habilidades y conocimientos, sino también a otros aspectos relevantes del proceso de aprendizaje. Por ejemplo, en la detección de eventos de abandono escolar, otras causas más relacionadas con la participación sentimental y cognitiva son importantes. Si el estudiante corre el riesgo de abandonar la escuela, una lección mal escogida puede terminar en el abandono del curso. Por otro lado, si el estudiante no muestra signos de desconexión, la lección puede tener poco que aportar al estado del estudiante, que permanecerá constante. Dependiendo de la tarea de pretexto utilizada para obtener la incrustación en el espacio latente, se capturarán diferentes aspectos del alumno y de los elementos del curso. Sin embargo, estas hipótesis sobre las características del espacio latente son sólo para fines ilustrativos, ya que el espacio en sí no es directamente interpretable.

## 2.5 Formalización del problema

En esta sección describiremos una formalización del problema de modelar los estados de los estudiantes a través del tiempo y los elementos del curso.

Consideramos un curso en línea como un conjunto de elementos de aprendizaje $E$, por ejemplo, lecciones o ejercicios. Cada elemento está asociado a un conjunto de posibles resultados $O_e$. Los resultados varían según el tipo de elemento: completado o no completado para las lecciones, una calificación de evaluación para el ejercicio, una respuesta de opción múltiple, un evento de deserción escolar, etc. Con estos elementos podemos definir el conjunto de interacciones $I_s$ para cada alumno $s \in S$, en cualquier momento $t \geq 0$. Es necesario tener en cuenta que, en este contexto, el tiempo se utiliza sólo para indicar el orden de la interacción dentro de la secuencia (índice) y no se refiere a las horas o días transcurridos entre las interacciones.

$$I_{st} = \{(e, o)|e \in E, o \in O_e\}$$

Ahora que hemos establecido nuestra notación, podemos abordar la solución del problema. Como se ha descripto anteriormente, modelar algo significa encontrar una representación del objeto en algún espacio matemático, normalmente un vector, llamado *embedding*. Nuestro objetivo es, entonces, encontrar las funciones de embedding $\varphi_E$ y $\varphi_S$ que proyectan los elementos del curso $E$ y los estudiantes $S$, respectivamente, en un espacio de habilidades latentes $LS$.

En este espacio de habilidades latente, representamos los elementos del curso simplemente como vectores. Esta representación vectorial puede interpretarse como la forma en que la interacción con el elemento del curso transforma el estado del estudiante. Entonces:

$$\varphi_E : E \to LS$$

Alternativamente, la función $\varphi_E$ también puede tomar la salida de la interacción como un parámetro adicional para generar el vector de embedding.

Los estudiantes serán representados como una secuencia de vectores, cada uno de los cuales corresponderá al estado del estudiante después de cada interacción con el contenido del curso. Es necesario tener en cuenta que, para calcular el estado del estudiante después de una interacción, necesitamos usar el estado anterior. Como resultado, la función de embedding $\varphi_S$ se define recursivamente y tiene la siguiente forma:

$$\varphi_S : (LS \times LS) \to LS$$

donde el primer argumento es la incorporación de una lección y el segundo es la incorporación de un estudiante.

Podemos entonces definir las siguientes series:

$$e_{st} = \varphi_E(I_{st})$$

$$s_t = \varphi_S(e_{st}, s_{t-1})$$

Usando estas ecuaciones, tenemos i) una definición del tipo de proyecciones de embedding que estamos buscando, y ii) un marco para comparar otros trabajos y determinar si están encontrando enfoques similares para representar a los estudiantes y los recursos educativos.

### 2.5.1 Optimización

Las funciones de embedding $\varphi_S$ y $\varphi_E$ se aprenden mediante la optimización de un modelo $C$ con respecto a una *tarea de pretexto*. En esta configuración, es importante definir una función de coste o una función de pérdida $L$ que calcule el error de clasificación. El modelo se optimizará con respecto a esta función de pérdida.

Nos centraremos en el caso del aprendizaje supervisado, donde los ejemplos tiene etiquetas asignadas. La secuencia $I_s$ asociada con cada estudiante puede ser etiquetada con una sola etiqueta, como por ejemplo, "dropout", o con una etiqueta para cada paso del tiempo, como la salida esperada de la interacción. El conjunto de ejemplos puede formalizarse como:

$$X = \{(I_s, \hat{y}) : s \in S\}$$

donde $\hat{y}_s$ es la etiqueta del estudiante $s$, opcionalmente ser un vector del tamaño $|I_s|$ o una sola etiqueta. En ambos casos, la función de pérdida estima el error entre la salida del clasificador y la etiqueta real del ejemplo. El entrenamiento del modelo $C$ con los parámetros $\theta$ consiste en minimizar la función de pérdida o *loss* para un conjunto dado de ejemplos con respecto a $\theta$.

$$loss = \sum_{s \in S} error(\hat{y}_s, C(I_s; \theta))$$

## 2.6 Preguntas de investigación

Hemos definido previamente el objetivo de este trabajo, que se centra en la evaluación de los métodos de aprendizaje de la representación. En este capítulo hemos descripto el problema a abordar, incluyendo una propuesta formal de un marco adecuado para las diferentes tareas de EDM. Para guiar el resto de esta investigación, diseñamos las siguientes preguntas de investigación que se responderán en los siguientes capítulos:

1. ¿Cómo se relacionan las representaciones propuestas con los enfoques anteriores del modelado de estudiantes?

2. ¿Cómo implementar y optimizar los embeddings propuestos utilizando el mismo espacio latente?

3. ¿Son útiles los embeddings conjuntos para diferentes tareas de aplicación? Esta pregunta tiene tres sub-preguntas:

(a) ¿Se observan mejoras significativas en el rendimiento?

(b) ¿Se observan mejoras significativas en la interpretabilidad de las representaciones?

(c) ¿Qué escenarios se benefician más del uso de los embeddings conjuntos? ¿Cuáles son los menos beneficiados?

# Capítulo 3

# Tareas de aplicación y trabajo previo

Para entender el impacto de las diferentes técnicas de aprendizaje de representaciones, es necesario evaluar el rendimiento y el comportamiento de un modelo en un entorno determinado. Para ello, nos centramos en dos tareas principales: *Knowledge Tracing* o seguimiento de conocimiento y la predicción de la deserción escolar, que se definirán en detalle en las siguientes secciones.

En este capítulo, estudiaremos el trabajo previo relacionado con cada tarea, centrándonos en cómo se modelaron los estudiantes, y posiblemente los elementos del curso. Muchas veces, las representaciones creadas no son analizadas explícitamente por los autores originales, ya que tienen como objetivo principal resolver la tarea en cuestión. Presentaremos cada trabajo de manera relevante a su contribución al campo, y también los compararemos en base a cómo entienden y representan el proceso de aprendizaje.

La primera tarea, Knowledge Tracing, está claramente definida y existen varios conjuntos de datos para comparar diferentes enfoques. Presentaremos cuatro de los métodos más exitosos utilizados, los diferentes aspectos que captan y sus beneficios o desventajas. Se prestará especial atención a las dos obras principales que inspiraron esta tesis: Piech et al. [2015] y Reddy et al. [2016].

Por otra parte, la predicción de la deserción escolar puede referirse a diferentes tareas que dependen del conjunto de datos particular considerado en cada trabajo. No existe un consenso claro sobre qué es la deserción escolar y cómo medirla. Comenzaremos el análisis de la tarea comparando diferentes definiciones, factores comunes que se tienen en cuenta para predecir la deserción escolar, y finalmente compararemos diferentes modelos.

Cerramos este capítulo con una comparación de los diferentes escenarios de formación y su posible impacto en escenarios reales. Proponemos un entorno de detección temprana de deserción

escolar que podría ayudar a utilizar el modelo en tiempo real para detectar a los estudiantes en riesgo y tomar medidas preventivas.

## 3.1 Knowledge Tracing

Knowledge Tracing (KT), o rastreo de conocimiento, es un método de referencia para modelar el conocimiento de los estudiantes con el fin de predecir su rendimiento futuro. La tarea es estimar hasta qué punto un estudiante ha dominado cada una de las habilidades que se enseñan son lecciones o ejercicios. Para ello, es necesario disponer de un método que permita estimar los conocimientos del alumno, lo que requiere que los elementos del curso sean graduados (al menos algunos). En muchos casos, también es necesario conocer las habilidades o componentes de conocimiento asociados a cada elemento del curso. Este tipo de información es más común en los entornos de aprendizaje de los ITSs que en los MOOC. Además, el algoritmo interno de los ITSs para recomendar itinerarios de aprendizaje utiliza una estimación de este tipo. Como resultado, la mayoría de los enfoques de KT se centran en los datos de los sistemas de tutoría.

El KT es usualmente entendido como una tarea de predicción de series de tiempo, donde la entrada es la secuencia ordenada de interacciones entre un estudiante y los ejercicios graduados. El resultado de cada interacción es el resultado de una prueba o ejercicio, y se utiliza para actualizar el nivel de conocimiento estimado del estudiante. Hay varias variaciones: en algunas, las habilidades posibles son preestablecidas por el curso, mientras que en otras, son descubiertas por el modelo durante el aprendizaje. También hay una distinción entre los modelos que estiman un nivel de dominio binario, es decir que la habilidad es aprendida o no, y otros modelos que utilizan escalas más granulares.

Al igual que en otras tareas de EDM, todavía no existe un vocabulario común entre los enfoques posibles. Lo que en BKT se llama comúnmente componentes de conocimiento, en el trabajo de SPARFA se llama conceptos abstractos, y en los trabajos de Reddy et al. [2016] y Piech et al. [2015] se llama habilidades. En la mayoría de los casos estos diferentes conceptos representan, al menos parcialmente, la misma intuición de causa latente que proponemos en este trabajo, y los modelos también intentan descubrir representaciones de estudiantes y elementos del curso utilizando diferentes enfoques. En las siguientes secciones explicaremos y compararemos las diferentes soluciones propuestas para la tarea de KT, y las relacionaremos con la definición formal del problema que hemos introducido en la Sección 2.5.

### 3.1.1 Bayesian Knowledge Tracing

El Bayesian Knowledge Tracing (BKT) es uno de los métodos más utilizados para estimar el conocimiento de los estudiantes, y ha sido ampliamente validado a lo largo de los años [Corbett and Anderson, 1994].

El conocimiento de los estudiantes se representa como un conjunto de variables binarias latentes, una para cada habilidad, siguiendo la teoría de todo o nada de aprendizaje humano. Cada variable indica si el estudiante ha aprendido esa habilidad en particular o no. Como sugieren estudios posteriores, esto no es un modelo completo del proceso de aprendizaje, ya que sólo puede capturar un conjunto pre-estipulado de variables latentes.

En el método original, el estado de la variable latente está condicionado por cuatro probabilidades: $P(K_{s,0} = 1)$, la probabilidad de que el estudiante haya dominado la habilidad antes de resolver el primer ejercicio; $P(K_{s,i+1} = 1|K_{s,i} = 0)$, la probabilidad de transición del estado no dominado al dominado; $P(X_{s,i} = 1|K_{s,i} = 0)$, la probabilidad de adivinar correctamente la respuesta antes de dominar la habilidad; y $P(X_{s,i} = 0|K_{s,i} = 1)$, la probabilidad de responder incorrectamente debido a un deslizamiento después de dominar la habilidad. Todas las habilidades deben ser determinadas antes de la optimización del modelo, y todos los ejercicios deben ser etiquetados con las habilidades correspondientes. El modelo no contempla la posibilidad de olvidar una habilidad dominada.

Los parámetros del modelo BKT se aprenden generalmente utilizando el método de *expectation maximization*, la búsqueda de gradiente conjugado o la búsqueda discreta por fuerza bruta.

El modelo básico BKT es un modelo muy adecuado e interpretable, a costa de sacrificar generalidad. La necesidad de contenido etiquetado, los pocos estados y transiciones posibles, y la binarización de las variables crean una representación restringida del proceso de aprendizaje.

Khajah et al. [2016] describen posibles extensiones de BKT que abordan algunas de sus deficiencias. En primer lugar, se incluye en el modelo una probabilidad de olvidar una habilidad aprendida. En segundo lugar, las habilidades latentes del estudiante se incluyen en el modelo como un parámetro inferido a partir de los datos, lo que permite descubrir habilidades distintas a las predefinidas. Por último, los autores proponen un enfoque para descubrir asociaciones latentes entre ejercicios para derivar sus habilidades asociadas. Con estas modificaciones, los ejercicios se modelan como abstracciones que se infieren exclusivamente a partir de datos. Sin embargo, el modelado de los estudiantes permanece como una combinación de ciertas probabilidades y no puede ser adaptado a otros escenarios aparte de KT. No hay modelado de otros aspectos del estudiante que no estén relacionados con las habilidades en sí, como la facilidad intrínseca o el involucramiento con el tema.

### 3.1.2 Métodos de Factor Analysis

Performance Factor Analysis (PFA) [Pavlik et al., 2009], y SPARse Factor Analysis (SPARFA) [Lan et al., 2014], son otros métodos bien conocidos para el Rastreo de Conocimiento. En términos generales, representan la probabilidad de que un estudiante proporcione la respuesta correcta a una pregunta dada. Esta probabilidad se basa en tres factores: su conocimiento de los conceptos subyacentes, los conceptos implicados en cada pregunta y la dificultad intrínseca de cada pregunta. Esto significa que, para estimar el nivel de dominio de un componente de conocimiento, el modelo utiliza conjuntamente *análisis del aprendizaje* y *análisis del contenido*.

SPARFA se basa en varios supuestos, que se dan en entornos de aprendizaje en línea: i) "dominios educativos de interés típicos implican sólo un pequeño número de conceptos clave", ii) cada elemento del curso implica sólo un pequeño subconjunto de conceptos abstractos, y iii) los elementos del curso tienen una fuerte relación positiva con estos conceptos abstractos. El primer supuesto es una limitación de los escenarios en los que puede aplicarse el método. Algunos conjuntos de datos, como las últimas versiones del conjunto de datos ASSISTments, involucran miles de componentes de conocimiento, lo que contradice el supuesto i). Los autores evalúan este modelo en una versión antigua del conjunto de datos ASSISTments que comprende sólo 403 estudiantes y 129 preguntas diferentes, aunque Xiong et al. [2016] incluye resultados para la versión completa del conjunto de datos.

El tercer supuesto, por otro lado, determina la relación entre los conceptos abstractos y los estados del estudiante. Establece que si los estudiantes tienen un fuerte dominio sobre un concepto, interpretado como un número positivo, no puede perjudicar su desempeño. Este modelado de la relación entre el estudiante y los elementos del curso sólo tiene en cuenta las preguntas o ejercicios calificados. Es posible pensar que otros tipos de interacciones, como las lecciones no calificadas y el acceso a los foros, pueden tener un tipo de relación diferente. Por ejemplo, si hay un fuerte dominio de una habilidad en particular, la ganancia de una lección que explica que la habilidad puede ser casi nula. Además, existe un modelo explícito del decaimiento del aprendizaje de una habilidad cuando no se ve en mucho tiempo.

Se proponen dos algoritmos para optimizar el modelo a partir de las observaciones incompletas proporcionadas por los conjuntos de datos. El primero utiliza una optimización biconvexa de expectation maximization para producir una estimación puntual de los factores. El segundo utiliza la estimación bayesiana para obtener la distribución posterior de los factores.

Una ventaja de este tipo de modelo es que puede ser interpretado. Según los autores, es posible "identificar el nivel de conocimiento de los alumnos sobre conceptos particulares y diagnosticar por qué un alumno determinado ha respondido incorrectamente a una pregunta o tipo de pregunta en particular". Además, existe un modelado explícito de las relaciones ocultas entre los elementos del curso utilizando los factores latentes.

Muchos aspectos de nuestra propuesta son similares al modelo presentado en SPARFA. Primero, también apoyamos la idea de modelar al estudiante con respecto a los elementos del curso usando las mismas causas latentes, o conceptos abstractos en SPARFA. A diferencia de los métodos BKT, tanto SPARFA como nuestra propuesta descubren la representación de los elementos del curso con respecto a las causas latentes, en lugar de basarse en un conjunto de factores predefinidos manualmente. En otras palabras, SPARFA también define y aprende las funciones de incrustación $\varphi_E$ y $\varphi_S$. El número de posibles causas latentes implicadas en cada elemento es un hiperparámetro del modelo.

Sin embargo, nuestro enfoque también tiene algunas diferencias con SPARFA. El más importante es el uso de incrustaciones neurales en lugar de un modelo de análisis factorial para encontrar las funciones de incrustación, introduciendo representaciones no lineales. Además, exploramos diferentes tipos de funciones de relación entre las representaciones de los estudiantes (interpretadas como el nivel de dominio de las habilidades) y los elementos del curso, enumerados en la Sección 4.2. Esto nos permite contemplar más categorías de elementos del curso, aparte de las preguntas calificadas.

### 3.1.3 Item Response Theory

Item Response Theory (IRT) es una teoría general de evaluación, aplicada en diferentes aspectos de la educación y no limitada a entornos online [Harvey and Hammer, 1999]. Se centra principalmente en el modelado y desarrollo de ítems, donde el término ítem es genérico, cubriendo todo tipo de ítems informativos. A diferencia de otras teorías, la IRT no asume que todos los elementos de prueba tengan la misma dificultad intrínseca. En el campo de la EDM, se ha utilizado como base para varios métodos estándar que modelan las respuestas de los estudiantes.

El paradigma de la IRT también se basa en la idea de que la probabilidad de una respuesta correcta a un elemento es una función matemática de los parámetros de la persona y del elemento. Además, los estudiantes evaluados son considerados en su ambiente y no aislados o en un estado constante. En el trabajo de Wilson et al. [2016], la IRT se describe como un método para "estimar las cantidades latentes correspondientes a la capacidad de los estudiantes y las propiedades de evaluación como la dificultad". En el enfoque más simple de la IRT, los estudiantes se caracterizan por una sola estimación de $theta_s \in \mathbb{R}$ de su competencia, mientras que los elementos del curso se representan con una estimación de $beta_e \in \mathbb{R}$ de su dificultad.

Usando la notación proporcionada en 2.5, la función $\varphi_S = s \to \theta_s$ y $\varphi_E = e \to \beta_e$. Para estimar estos parámetros a partir de un conjunto de datos de las respuestas de los estudiantes $I_{st}$, suponiendo una independencia de $\theta_S$ y $\beta_E$, y un valor previo normal para ambos parámetros, se maximiza la siguiente probabilidad posterior:

$$\log P(\theta_S, \beta_E | I_{st}) = \sum_{s \in S, t} \sum_{(e,o) \in I_{st}} r \log \Phi(\theta_s - \beta_e) + (1 - r) \log \Phi(\theta_s - \beta_e) - \frac{1}{2}||\theta_S|| - \frac{1}{2}||\beta_E|| + C$$

donde la salida de una interacción es una variable binaria que indica si la pregunta fue respondida correctamente, $\Phi$ es la función de distribución acumulativa de la distribución normal estándar. Podemos ver que este modelo está definiendo explícitamente la relación entre las representaciones de los estudiantes y los elementos del curso. Establece que la diferencia entre ellos sigue una distribución normal estándar. Bajo esta suposición, la probabilidad de resolver el ejercicio aumenta a medida que el nivel de dominio del estudiante sobre la habilidad está más cerca del nivel necesario para resolver el ejercicio, o más alto.

Una extensión del método básico de la IRT, llamada Hierarchical IRT, explota información adicional como la estructura del conjunto de problemas (a veces organizada en categorías más generales) y las etiquetas de habilidades asignadas a los ejercicios. La dimensión temporal de los datos se incluye en una extensión del IRT Temporal, donde el conocimiento del estudiante se modela como un proceso de Wiener.

En [Wilson et al., 2016] los autores comparan la base IRT y las dos extensiones nombradas con el enfoque DKT tradicional, descripto en la siguiente sección, y obtienen un mejor rendimiento en tres conjuntos de datos diferentes, pero utilizando información más granular que el nombre de la habilidad siempre que sea posible.

Podemos ver que la IRT se centra únicamente en la respuesta de los estudiantes a los elementos del curso clasificados, y no se ha adaptado a los elementos no clasificados. Además, sólo puede modelar un aspecto del estudiante a la vez. Esta limitación puede ser ignorada en algunos escenarios como KT, donde la mayoría de los ejercicios involucran un pequeño conjunto de componentes de conocimiento similares, pero no puede modelar fenómenos más complejos. Aunque consideramos que estas representaciones son una simplificación excesiva tanto de los estudiantes como de los elementos del curso, el núcleo del modelo es el mismo que en nuestra propuesta. Las principales diferencias son:

- Utilizamos múltiples dimensiones para representar a los estudiantes y a los elementos del curso simultáneamente, mientras que ellos modelan una dimensión a la vez.
- Utilizamos una función a distancia para modelar la relación entre los estudiantes y los elementos del curso.
- En IRT, es posible modelar explícitamente una dimensión temporal, mientras que nuestro enfoque se basa en la capa recurrente para descubrir el efecto de la temporalidad a partir de los datos.

### 3.1.4 Deep Knowledge Tracing

Con la popularidad ganada por las redes neuronales en los últimos años, era imperativo modelar a los estudiantes en entornos de aprendizaje en línea utilizando un enfoque neural. Deep Knowledge Tracing (DKT) fue desarrollado por Piech et al. [2015]. Los autores utilizan una arquitectura neuronal recurrente para predecir la probabilidad de que un estudiante responda correctamente a los ejercicios del conjunto de datos. Proponen interpretar el estado oculto de una capa recurrente en este modelo neural como la representación del estudiante. Este estado mantiene un registro de toda la historia de las interacciones del estudiante con los elementos del curso, ya que se actualiza después de cada interacción. De ello se deduce que DKT aplica el concepto de *tarea de pretexto*, en este caso, prediciendo la probabilidad de que un alumno responda correctamente a los ejercicios, para construir la representación de los alumnos, aunque los autores no utilizan este vocabulario exacto para describirla. La representación obtenida es el equivalente al vector BKT de los niveles de dominio de habilidades, y puede ser utilizada para la recomendación de ítems.

En el siguiente capítulo detallaremos la implementación interna de las redes recurrentes. Durante esta introducción a DKT, sólo presentaremos cómo se interpretan la entrada y la salida, y cómo se modelan los elementos del estudiante y del curso.

Para una interacción dada en el momento $t$, el elemento del curso $e_t$ se representa como un vector de *one-hot encoding*. El one-hot encoding de un elemento con id $i$ es un vector donde todos los elementos son ceros, excepto el elemento i-ésimo, que tiene valor uno. Este es el método más simple para codificar un valor categórico en un vector numérico, y ha sido ampliamente utilizado en otras áreas como el procesamiento de lenguaje natural. Funciona bien en la práctica, con dos inconvenientes principales: la cantidad de memoria se escala linealmente con el número de elementos diferentes, y no hay relación entre dos vectores, ya que todos son ortogonales.

La salida de la interacción es un valor binario: $o_t \in \{0, 1\}$, representando si el estudiante completó correctamente el ejercicio en el primer intento.

La entrada para la red es la secuencia de interacciones $\{I_{st}\}_{t<T}$ representada como un conjunto de vectores $x_1, x_2, \ldots, x_n$. Cada vector $x_t$ es la concatenación del one-hot encoding del elemento y la multiplicación del mismo vector por la salida de la interacción. En otras palabras, el vector para el elemento $i$ tiene todos los ceros excepto las posiciones $i$ si el ejercicio fue contestado incorrectamente en el primer intento, o bien en las posiciones $i$ y $2i$ si fue contestado correctamente.

Como se mencionó anteriormente, la red está optimizada para predecir la probabilidad de que el estudiante pase un ejercicio en la siguiente interacción $t + 1$. Como resultado, la salida es un vector $y \in \mathbb{R}^{|E|}$, donde cada valor $y_i$ es la probabilidad de pasar el elemento de curso $i$ en el

primer intento.

El modelo neural, como cualquier arquitectura profunda, puede captar relaciones complejas entre las causas latentes de la tarea en cuestión. Como podemos ver, la única entrada necesaria es la identificación del elemento del curso y la salida de la interacción, convirtiéndose así en independiente de los componentes de conocimiento asignados manualmente. La arquitectura recurrente añade las siguientes propiedades:

- Procesa un elemento de la secuencia a la vez.
- Mantiene un estado interno oculto donde puede acumular la información relevante de la secuencia vista hasta ahora.
- Utiliza el valor de la nueva entrada en el momento $t$ junto con el estado oculto del tiempo $t-1$ para calcular la salida.

Este tipo de modelo proporciona un enfoque natural para procesar secuencias de tiempo y es lo suficientemente flexible para la mayoría de las tareas. Puede capturar las relaciones temporales entre los elementos, incluso cuando no están próximos en la secuencia. Los modelos que utilizan celdas más sofisticadas, como LSTMs y GRUs, pueden incluso distinguir entre la información importante que se debe conservar mientras se procesa toda la oración, y la información ruidosa que se debe descartar u olvidar. Dados sus buenos resultados, se han convertido en las arquitecturas recurrentes por defecto a utilizar.

Entendiendo la red neuronal como un modelo de aprendizaje de representaciones, Piech et al. [2015] propone interpretar el estado oculto de la capa recurrente como la representación del estudiante. Este estado hace un seguimiento del historial de interacciones desde el inicio del curso, almacenando sólo la información relevante. Se actualiza después de cada interacción con la nueva salida. Así, representa el estado del estudiante en cada paso temporal $t$ con la información óptima para inferir qué ejercicios está listo para resolver y cuáles no, construyendo así una estimación del nivel de conocimiento actual.

Este modelo tiene todas las propiedades que hemos identificado como necesarias para un tratamiento general del problema de modelado del estudiante. Sin embargo, no hay un modelado explícito del elemento del curso o de la relación entre los dos embeddings. Propondremos una extensión de la arquitectura DKT básica para establecer una relación explícita entre estos dos embeddings. Nuestro modelo es también más general que el DKT básico en el sentido de que la interpretación de Piech del estado oculto recurrente se propone sólo para la tarea de Knowledge Tracing, mientras que la nuestra también es útil para cualquier tarea secuencial de EDM.

Por muy conveniente que sea esta representación, tiene una limitación importante. En DKT, como en cualquier arquitectura neuronal, los parámetros del modelo no son directamente interpretables. Como alternativa a la interpretación del modelo, podemos analizar cómo los diferentes parámetros contribuyen al resultado final. Por ejemplo, Piech et al. [2015] también propone utilizar estas representaciones para mejorar los planes de estudio de los cursos y descubrir relaciones ocultas entre ejercicios.

### 3.1.5 Bayesian vs Deep Knowledge Tracing

Después de la primera propuesta de DKT, surgió una gran controversia en la comunidad KT, especialmente en comparación con modelos tradicionales como BKT e IRT. Parte de esta atención crítica se debió a un error en el trabajo de Piech et al. [2015], donde una parte del conjunto de datos de evaluación se utilizó también para el entrenamiento, lo que produjo puntuaciones de rendimiento más altas. Sin embargo, gran parte de las críticas se debieron a las propiedades intrínsecas del enfoque y a la falta de interpretación del modelo. En esta sección profundizaremos en las similitudes y diferencias entre DKT y BKT.

La tarea objetiva es muy similar para ambos métodos: utilizan el éxito o el fracaso de los intentos de los estudiantes de resolver ejercicios para calcular el conocimiento actual atribuido a cada habilidad, pero en entornos muy diferentes. BKT utiliza un Modelo Markov Oculto y se basa en la teoría del aprendizaje humano todo o nada (una habilidad se aprende o no), sin posibilidad de olvidar. DKT, por otro lado, utiliza una Red Neural Recurrente (RNN) alimentada con la secuencia de intentos de un estudiante y entrenada para predecir el desempeño del estudiante en el siguiente ejercicio. El estado oculto de la red se utiliza como representación del conocimiento del alumno, añadiendo naturalmente la expresividad al modelo de que una determinada habilidad ha sido parcialmente olvidada.

BKT y otros enfoques para esta tarea se basan en la información que se ha asignado manualmente a los elementos del curso, como las habilidades o los componentes del conocimiento, para ayudar a modelar el estado del conocimiento del estudiante. BKT puede ser entrenado sin información de habilidades, pero los resultados de la tarea de pretexto no son competitivos en este entorno.

La principal ventaja de DTK sobre BTK es que puede inferir automáticamente las habilidades involucradas en un ejercicio, sin necesidad de etiquetas manuales añadidas por un experto. Además, el dominio de una habilidad por parte del estudiante se representa como un estado continuo en lugar de un estado binario, lo que permite diferentes niveles de "conocimiento". Sin embargo, Xiong et al. [2016] analiza la DKT y encuentra varios inconvenientes en el método en comparación con los enfoques ITS. Incluso descubren que su rendimiento es comparable con los métodos de análisis de factores de rendimiento. Khajah et al. [2016] argumentan que la

evaluación de las competencias de los estudiantes no se beneficia de los modelos profundos: los modelos superficiales funcionan igual de bien y ofrecen una mayor interpretabilidad.

Sin embargo, incluso si estos puntos son ciertos para la tarea de KT, los otros modelos propuestos como BKT, SPARFA e IRT no son fácilmente adaptables a otras tareas de EDM como la predicción de la deserción escolar. Están diseñados para modelar al estudiante con respecto a su nivel de dominio sobre las habilidades, sin portabilidad a otras tareas. En contraste, DKT puede ser fácilmente adaptado se basa en una arquitectura neural que puede ser utilizada para una amplia variedad de tareas de pretexto.

### 3.1.6 Otros enfoques de modelado de estudiantes

Modelar estudiantes no es una tarea limitada al nivel de conocimiento [Chrysafiadi and Virvou, 2013]. Además del costo de etiquetar los ejercicios con habilidades, a veces la información recopilada de las interacciones no registra el resultado de la interacción. Existen entornos de aprendizaje, como algunos MOOCs, donde no hay información sobre el desempeño del estudiante en las pruebas, sino que la única información que se registra son los elementos del curso visitados por el usuario. En esos escenarios, todavía podemos explotar los registros en bruto para obtener un modelo de comportamiento de los estudiantes. Estos modelos pueden tener un impacto en varias tareas: recomendación de ejercicios, prevención de la deserción escolar, estimación del nivel de compromiso, etc.

El trabajo de Tang et al. [2016] también modela a los estudiantes usando Redes Neuronales Recurrentes, entrenadas para predecir la siguiente acción a ser realizada por el estudiante. Este enfoque no requiere que se califiquen los ejercicios. El autor no se centró en ninguna aplicación específica y sólo proporcionó una prueba de concepto de la idoneidad de los modelos neurales para aprender los datos.

El método propuesto por Reddy et al. [2016] genera embeddings de estudiantes, junto con embeddings de elementos del curso. En lugar de utilizar redes neuronales, los autores definen un conjunto de ecuaciones para actualizar el estado del estudiante después de cada interacción y, a continuación, maximizar la probabilidad de que se les proporcione un conjunto de datos de entrenamiento utilizando un método de Máximo a Posteriori (MAP).

Se contemplan dos tipos de elementos del curso: lecciones y evaluaciones. La salida de una interacción con una lección es siempre 1 (y por lo tanto ignorada), representando que el estudiante ha visto efectivamente el contenido, mientras que la salida de una interacción de evaluación es binaria. Formalmente, el estado de un estudiante en un momento dado se representa como un vector en el espacio de habilidades latente; una lección se representa como un vector de ganancias de habilidades y un vector de requisitos previos de habilidades; un módulo de evaluación

se representa sólo como un vector de requisitos de habilidades.

La probabilidad de un resultado positivo $o_t$ en la interacción en el momento $t$ con la evaluación $e_t$, dada la representación del estudiante $\varphi_S(s_t)$ está dada por:

$$P(o_t|s_t, e_t) = Bernoulli(f(\Delta(\varphi_S(s_{t-1}), \varphi_E(e_t))))$$

donde $\Delta$ es la distancia entre el estudiante y las incrustaciones de la evaluación y $f$ es una función logística (sigmoid).

El nuevo estado del estudiante se interpreta en términos de adquisición de habilidades. A partir del estado $\varphi_S(s_{t-1})$, después de la interacción $t$ con una lección con ganancias $l$ y prerrequisitos $p$, la función de embedding $\varphi_S$ se modela como:

$$\varphi_S(s_t) = \Phi(\varphi_S(s_{t-1}) + l \cdot f(\Delta(\varphi_S(s_{t-1}), p)) \tag{3.1}$$

donde $\phi$ es una distribución normal. De la ecuación anterior se deduce que la ganancia de conocimiento tiene una distribución normal alrededor del estado anterior más el producto de puntos entre la ganancia intrínseca de la lección y la distancia desde el estudiante hasta el prerrequisito de la lección.

Si el estado del estudiante está cerca del prerrequisito de las lecciones, la ganancia será mayor, mientras que si está lejos en cualquier dirección, la ganancia será menor. Esto añadiría al espacio un significado a la distancia entre los dos embeddings: ganancias menores pueden deberse a que los estudiantes saben muy poco y el ejercicio será demasiado difícil o a que saben demasiado y habrá aprendizaje. Esta intuición ya ha sido presentada anteriormente y es un concepto clave de nuestra propuesta. También es similar a la probabilidad de resolver el ejercicio explícitamente modelado sobre el método BKT.

Sin embargo, el modelo proporcionado por [Reddy et al., 2016] sólo permite cambiar la representación de los estudiantes en una sola dirección fija, variando sólo la magnitud del cambio. En la ecuación 3.1, la ganancia de habilidad asociada con la lección es ponderada por el resultado de la distancia *varDelta*, que es un escalar. Como resultado, la dirección de la lección es independiente del estado anterior del estudiante. Por lo tanto, no puede modelar el caso donde el estudiante tiene un fuerte dominio de algunas de las habilidades involucradas en la lección pero carece de otras, y debe ser actualizado sólo en las direcciones de las habilidades desconocidas, mientras que permanece constante en las otras. Este concepto es una distinción importante entre este modelo y nuestra propuesta.

## 3.2 Predicción de deserción

La segunda tarea de pretexto para evaluar las diferentes técnicas de embeddings propuesta en esta tesis es la predicción de la deserción escolar. El objetivo de esta tarea es predecir si los estudiantes abandonarán los cursos en lugar de completarlos. Este es un problema muy común en los MOOCs, y una fuente frecuente de críticas hacia los cursos masivos. La promesa de llegar a miles de estudiantes a través de plataformas en línea se ve disminuida por las bajas tasas de finalización. En algunos casos, hasta un 95% de los estudiantes no vuelven a un curso que han comenzado. Para los proveedores de cursos, un alto nivel de abandono puede indicar que el material es de mala calidad o ineficaz. Además, si el curso es gratuito y la institución sólo cobra por el certificado final, los altos índices de deserción escolar implican pérdidas económicas.

Como se explica en la sección 2.3, el fenómeno de la deserción escolar proviene de la propia naturaleza de los cursos de MOOC y no es trivial de resolver. También depende en gran medida de cómo se mide la deserción escolar y el objetivo final que se persigue detrás de la investigación. Si el estudiante debe inscribirse en un curso para ver su contenido, y después de la primera visita nunca regresa, ¿es realmente una situación de abandono? Se puede considerar que el estudiante no estaba inicialmente interesado en el curso. Como consecuencia, la personalización de una trayectoria dentro del curso puede no tener un impacto en ese tipo de estudiante.

En el resto de esta sección describiremos algunos aspectos importantes de esta tarea y los diferentes tipos de propuestas que la abordan.

### 3.2.1 ¿Qué es la deserción escolar?

La primera dificultad que hay que tener en cuenta al abordar la predicción de la deserción escolar es la falta de una definición generalizada de la deserción escolar. Por ejemplo, la competencia KDDCup 2015 consideró como abandono la ausencia de actividad estudiantil en los 10 días siguientes a la finalización del curso. Sin embargo, no estaba claro si se debía hacer una diferencia en el tipo de interacción que el estudiante tomaba, o si una actividad posterior estaba presente o no. Lee and Choi [2011] también mencionan otras situaciones estudiadas en investigaciones anteriores e identificadas como deserción: fracaso o ausencia en la última etapa de la evaluación, repetición del curso, retiro formal del curso o de la institución. También se mencionan otros conceptos y se contrastan con la deserción escolar: retirada informal, fracaso en la finalización, retención, no persistencia, etc. El análisis de Halawa et al. [2014] sólo considera que se ha producido una deserción escolar si el estudiante ha estado ausente durante más de un mes y ha visto menos del 50% de los vídeos del curso.

En [Fei and Yeung, 2015], se proponen y evalúan tres definiciones diferentes de abandono escolar: i) si un estudiante registrará la actividad hasta el final del curso, ii) si la semana actual es la

última semana en la que el estudiante tiene actividades, iii) si un estudiante tiene actividades en la próxima semana.

Es importante señalar que muchas de estas definiciones sólo pueden ser evaluadas una vez finalizado el curso. Lo que consideramos como deserción escolar tiene un efecto crítico en la forma en que se aborda la tarea. A su vez, esto también depende de qué problema en particular se necesita resolver y en qué momento se espera que actúe el modelo.

Halawa et al. [2014] identifica dos tipos de beneficios del estudio de predicción de deserción escolar: a corto plazo y a largo plazo. A corto plazo, el instructor puede utilizar el modelo para identificar quién está en riesgo de abandono y quién no. El objetivo principal es detectar los patrones que predicen la deserción escolar y tomar medidas lo antes posible. Sin embargo, para ello, los datos de la formación deben ser coherentes con este objetivo y estar disponibles durante el curso. Esto significa que un escenario de formación sensato no debería utilizar ninguna información futura que no estaría disponible cuando se ponga en producción.

A largo plazo, "la predicción de la deserción escolar puede proporcionar información valiosa sobre las interacciones entre el diseño del curso y los factores de los estudiantes" [Halawa et al., 2014]. Potencialmente, esto podría ayudar a diagnosticar el curso por defectos comunes o incompatibilidades con ciertos tipos de estudiantes.

Es posible entrenar un modelo utilizando datos históricos de cursos anteriores y utilizarlo para detectar casos de abandono casi total en un curso nuevo y actual. Sin embargo, la diversidad entre los MOOCs dificulta la posibilidad de transferir el aprendizaje entre los diferentes cursos. Más aún, hay casos en los que no está claro si las presentaciones posteriores del mismo curso comparten algunas o todas las características. Como se analizó antes, esto no es sólo para la predicción de la deserción escolar, es el problema genérico de la no comparabilidad entre los diferentes cursos.

En contraste con el rastreo de conocimiento, la tarea de predicción de deserción escolar no está tan claramente definida. Depende de los datos disponibles, de las características del curso que se está estudiando y de las necesidades de los diseñadores de instrucción que utilizarán el sistema. Por lo tanto, el análisis, los modelos y las conclusiones que funcionan muy bien en un entorno no garantizan que se apliquen a otros. Si queremos crear un modelo general de predicción de deserción escolar, éste debe ser adaptable a estas diferentes situaciones.

### 3.2.2 Factores de la deserción

La deserción escolar puede considerarse como un sistema complejo, que implica muchas causas latentes con diferentes orígenes. Se ha estudiado a fondo por qué los estudiantes abandonan los cursos que han empezado, pero no existe un solo modelo general que explique el fenómeno.

Entender, al menos a nivel intuitivo, qué tipo de factor oculto puede estar afectando a la situación, nos permitirá interpretar mejor los modelos automáticos y qué aspectos necesitan capturar.

El trabajo de Lee and Choi [2011] compila una extensa revisión de los posibles factores que influyen en la deserción escolar a partir de investigaciones anteriores. Los autores se centran en los cursos en línea que ofrecen los institutos de educación postsecundaria. La lista de factores resultante se agrupa en tres categorías:

- Factores del estudiante, incluyendo aspectos tales como antecedentes académicos, experiencias relevantes, habilidades relevantes y atributos psicológicos.
- Factores del curso/programa, incluyendo el diseño del curso (en términos de interactividad, calidad general y relevancia para las necesidades de los estudiantes), apoyo institucional e interacciones.
- Factores ambientales, incluyendo compromisos de trabajo, ambientes de estudio de apoyo.

La bibliografía revisada indica que el impacto de factores demográficos como la ubicación, la edad y el género son muy variables. En algunos estudios, se encuentran estadísticamente correlacionados con la deserción escolar, mientras que otros no encuentran ninguna distinción entre los grupos. Como resultado, la categoría no se incluyó como factor relevante.

Entre las conclusiones del trabajo mencionado, "los factores de deserción más distintivos en los cursos en línea fueron las características de ingreso de los estudiantes, incluyendo sus experiencias y desempeño académico y profesional previo, sus habilidades de aprendizaje y sus atributos psicológicos. Además, el diseño de los cursos y el apoyo institucional influyeron en las decisiones de los estudiantes que abandonaron la escuela. Los entornos de apoyo y estímulo son necesarios para que los estudiantes logren una integración social en la que puedan adaptar con éxito el estudio con su trabajo, su familia y sus compromisos sociales".

Este análisis indica que el análisis de la conducta de los estudiantes permite obtener una valiosa información. Siguiendo a Halawa et al. [2014], nos centraremos únicamente en los factores relacionados con el diseño del curso y las características de los estudiantes. Los factores externos son difíciles de registrar, sólo a través de cuestionarios externos, y aún más difíciles de alterar. Los factores internos, por otro lado, están en el campo de acción de los diseñadores de cursos. Se pueden hacer mejoras en la estructura y el contenido del curso, mientras que los estudiantes en riesgo pueden ser detectados a través de los patrones de sus trazas de acceso.

Sin embargo, no hay una forma perfecta de estructurar un MOOC. Incluso los patrones exhibidos por los estudiantes que abandonan la escuela y los que no lo hacen no son consistentes.

Muchos de los factores mencionados también dependen del curso en sí. Por ejemplo, los autores mencionan que los estudiantes exitosos en los cursos STEM pasan más tiempo viendo páginas de contenido, mientras que los estudiantes en ciencias sociales, inglés y cursos de comunicación pasan más tiempo participando en discusiones, ya sea leyendo, respondiendo a las preguntas de otros estudiantes o publicando las suyas propias.

Además, la interacción entre los estudiantes también puede afectar al comportamiento individual. Yang et al. [2013] construyen un modelo de supervivencia para evaluar los factores que aumentan y disminuyen la deserción escolar a lo largo del camino, concluyendo que los factores sociales sí afectan a la deserción escolar. Potencialmente podrían proporcionar información para el diseño de cursos que promuevan el compromiso a través de las interacciones sociales.

El trabajo de Halawa et al. [2014], más centrado en lo que se puede aprender de los datos de registro de los estudiantes, destaca las siguientes características como fuertes predictores de la deserción escolar:

- El rendimiento autopercibido del estudiante.
- La habilidad para auto-regular el proceso de aprendizaje.
- El nivel de interés de los estudiantes sobre el tema que están aprendiendo.

Los autores también mencionan las dificultades para distinguir cuál de estos factores está afectando a un estudiante en particular: ¿la deserción escolar se debe a una disminución del interés o a algunos factores externos? Por lo tanto, es importante detectar los momentos vulnerables en el camino del estudiante y tomar medidas preventivas que permitan, en primer lugar, diagnosticar la causa de la posible deserción escolar y, en segundo lugar, tratar de mitigar la situación.

Incluso sin un modelo completo y sistemático del fenómeno de la deserción escolar, está claro que todos estos factores intervienen en el proceso de diferentes maneras. Cualquier aproximación general a la predicción de la deserción escolar debe ser lo suficientemente flexible como para tratar los factores ocultos sin depender de reglas explícitas, características del curso o experiencia en el dominio humano.

### 3.2.3 Enfoques a la predicción de deserción escolar

Hemos analizado el fenómeno de la deserción escolar desde varios puntos de vista. Es claramente una tarea con muchos factores ocultos, que pueden no ser discernibles entre ellos. Al igual que con la tarea de rastreo de conocimiento, la predicción de deserción escolar puede ser abordada

con algoritmos de aprendizaje automático que aprenden de un conjunto de datos de las interacciones de los estudiantes. Dividimos esos enfoques en basados en características *handcrafted* o artesanales, donde la representación del estudiante o la entrada se construye utilizando reglas definidas por humanos, y arquitecturas de aprendizaje profundo, donde esas representaciones se optimizan automáticamente a partir del conjunto de datos de entrenamiento.

Entre el primer grupo, Halawa et al. [2014] propone un modelo integrado con dos predictores: uno para el estado activo del estudiante, y otro para el estado ausente. El predictor de modo activo analiza las interacciones del estudiante hasta que se detecta una ausencia prolongada. En ese momento, el control se pasa al predictor de modo ausente, que evalúa si es probable que el estudiante abandone la escuela o no. Las características utilizadas incluyen patrones de salto de vídeo y de tareas, tiempo de ausencia y rendimiento en las evaluaciones.

En la misma categoría, el trabajo de Kloft et al. [2014] utiliza modelos de SVMs para predecir la deserción usando trazas de flujo de clicks de bajo nivel. La deserción se define como la falta de actividad en esa semana y en cualquier semana futura. La predicción se hace semanalmente, y así se genera una nueva representación para el estudiante para cada período. Las características de la representación incluyen, entre otras, el número de días activos, el número de vistas de contenido (por período y por tipo de contenido), el número de sesiones y el número de interacciones por sesión. El modelo SVM aumenta la precisión de la predicción, pero se compara con una simple línea de base de la probabilidad general de abandono en esa semana.

Xing et al. [2016] proponen un método para predecir el abandono gradual de un curso, centrado en los participantes del foro. Los pronosticadores de deserción anteriores no distinguen entre deserción inmediata y deserción en algún lugar del resto del curso. Esto puede impedir que los instructores del curso tomen medidas en los casos más urgentes. El modelo determina los estudiantes en riesgo de abandonar la escuela exactamente en la semana siguiente, utilizando datos recolectados de semanas anteriores.

En este trabajo, la representación del estudiante se construye utilizando dos métodos. La primera sólo añade los valores de las características actuales con las características históricas. El segundo añade las características de las nuevas semanas a las anteriores, generando mayores representaciones a lo largo del tiempo. Para mitigar este aumento de la dimensionalidad, se utiliza el método de PCA para identificar un punto de interrupción para desactivar las características históricas y mantener sólo las más nuevas. Este segundo modelo logró resultados muy superiores, especialmente en las últimas semanas. Incluso si el objetivo principal es resaltar el aspecto temporal, ambas representaciones utilizan una visión estática del estudiante agregado durante un período de tiempo (menor). Los clasificadores utilizados fueron redes generales bayesianas y árboles de decisión.

En contraste con los enfoques basados en características artesanales, hay muchos trabajos en

los últimos años que aplican modelos de aprendizaje profundo a la tarea de predecir la deserción escolar.

Fei and Yeung [2015] utiliza una Red Neural Recurrente y reporta un aumento en el rendimiento en comparación con los modelos de Markov, regresiones logísticas y support vector machines. Sin embargo, en lugar de utilizar la secuencia completa de interacciones, los datos se agregan en períodos semanales.

Los autores de [Wang et al., 2017] también proponen descubrir automáticamente representaciones utilizando redes neuronales profundas. Nombran algunas de las limitaciones de la ingeniería de características, argumentando que este proceso requiere conocimiento del dominio y que es particular para cada conjunto de datos específico.

El modelo propuesto en este trabajo es una combinación de dos tipos de arquitecturas neurales: Redes Neuronales Convolucionales (CNN) y Redes Neuronales Recurrentes (RNN). En un primer paso, los registros de interacción en bruto se procesan con una CNN, compuesta por varias capas convolucionales y de agrupación. Cada capa convolucional sólo encuentra relaciones entre elementos adyacentes. Las capas inferiores pueden encontrar algunas y las capas superiores pueden encontrar patrones en toda la escala de la entrada. Una vez que la entrada ha sido procesada por la CNN, se obtiene una representación de la interacción y se alimenta en una RNN, que puede descubrir patrones en toda la secuencia.

Este modelo CNN-RNN es evaluado empíricamente sobre el conjunto de datos KDDCup, reportando resultados comparables a los obtenidos en este trabajo.

El trabajo de chun Feng et al. [2019] propone Context-aware Feature Interaction Network (CFIN), una combinación de arquitecturas neuronales que utilizan técnicas de suavización del contexto para suavizar los valores de las características de la actividad a través de una CNN. Antes de esto, los estudiantes se agrupan de acuerdo con las características de acceso a los diferentes tipos de elementos del curso. Los autores argumentan que los "usuarios" que estudian el comportamiento pueden ser agrupados en varias categorías, que corresponden implícitamente a diferentes "motivaciones que los usuarios estudian los cursos MOOC".

Este trabajo utiliza el modelo de CNN para construir una representación del contexto, que se refiere a la interacción entre el estudiante y el elemento del curso. Esta representación es utilizada posteriormente por el modelo predictivo. Se utiliza un mecanismo de atención para combinar esta representación del contexto con información estática utilizada, como el género, la ubicación y la edad. El predictor final de abandono es una red neural profunda tradicional con una activación sigmoide en la última capa. Las características se extraen de los registros históricos de los usuarios, pero los autores no especifican cómo o qué información se utiliza.

Los autores utilizan el mismo conjunto de datos que este trabajo, el conjunto de datos KDDCup

2015, junto con un conjunto de datos mucho más amplio.

Los autores utilizan el mismo conjunto de datos que este trabajo, el conjunto de datos KDDCup 2015, junto con uno mucho más grande proporcionado directamente por la plataforma XuetangX. El modelo propuesto supera otras aproximaciones básicas, pero no está claro si el modelo más simple (como los LRs y SVMs) fueron entrenados usando la representación neuronal generada o con otras características hechas a mano.

### 3.2.4 Diferencias en el contexto de aplicación

Como se ha señalado en [Whitehill et al., 2017] y [Xing et al., 2016], muchos predictores de deserción escolar son entrenados y evaluados usando el historial completo de las acciones de los estudiantes en el mismo curso. Incluso la competición KDDCup 2015 utiliza un formato similar. Naturalmente, este es el entorno de evaluación más sencillo, muchas veces inspirado por la evaluación en otras tareas como el Rastreo de Conocimiento, destinado a los ITS. Un modelo entrenado para KT usando las interacciones registradas en los primeros meses que ha estado en línea puede ser usado para refinar el mismo sistema después de este punto, ya que se espera que los ITSs funcionen durante mucho tiempo y sean usados sin cambios en varias cohortes.

Sin embargo, se espera que los MOOCs sean seguidos por todos los estudiantes al mismo tiempo, fijando fechas de entrega y publicando nuevos contenidos semanalmente. Si la meta es predecir la deserción escolar, cualquier sistema de aprendizaje automático supervisado necesitará ejemplos de interacciones estudiantiles que terminaron en éxito o fracaso. Estas interacciones no estarán disponibles hasta que el curso termine. Para que un sistema pueda ser utilizado en tiempo real, no puede depender de información futura. En esos casos, los modelos entrenados en diferentes escenarios deben ser adaptados al curso en cuestión. En contraste, un sistema entrenado usando el historial completo de interacciones puede ser usado para:

- Predicción de la deserción escolar en versiones posteriores del mismo curso o similares, asumiendo que las características utilizadas para representar las acciones de los estudiantes pueden ser traducidas de un escenario a otro.
- Mejorar la interpretabilidad de los datos del curso, explicando las causas latentes de la deserción o la desconexión. Este punto puede tener un gran efecto en el diseño de nuevos cursos, pero es un análisis exploratorio no dirigido al rendimiento de los clasificadores.
- Probar un modelo en particular es apto para las tareas de predicción de la deserción escolar, y proporciona un límite superior para el rendimiento del sistema.

En cualquier otra situación en la que sea necesaria la retroalimentación del predictor mientras el curso está en curso, se debe establecer un tipo diferente de escenario. En consecuencia, la selección de un predictor de deserción y la metodología de capacitación deben alinearse con el problema a resolver. Una evaluación de un modelo entrenado con secuencias previas completas de interacciones puede ser engañosa si se espera que detecte a los estudiantes sólo a partir de información parcial. Además, según Whitehill et al. [2017], la formación y las pruebas en el mismo conjunto de datos pueden sobrestimar significativamente el rendimiento de los clasificadores.

### 3.2.5 Predicción temprana

El problema de la clasificación temprana de secuencias, también llamado clasificación temprana de series temporales, consiste en predecir la etiqueta de una secuencia con la menor cantidad de información posible. La motivación detrás de modelar una situación como un problema de clasificación temprana es actuar tan pronto como el clasificador tenga suficiente información para hacer una predicción certera. En contraste con la predicción tradicional de series de tiempo, estos sistemas no leen la secuencia completa de acciones para producir una etiqueta.

Los escenarios en los que se utiliza la clasificación temprana son muy diversos, ya que se trata de un marco general y adaptable. Podemos encontrar ejemplos de pronósticos de series de tiempo, por ejemplo, pronósticos de consumo de energía, respuesta a la terapia farmacológica en pacientes con esclerosis múltiple utilizando un conjunto de datos de dominio médico que contiene valores de expresión génica, esclerosis precoz, hasta detección e identificación de llamadas de aves en un escenario de estudio de biodiversidad [Mori et al., 2016].

La detección temprana de riesgos es también un uso muy importante de la clasificación temprana. Por ejemplo, la tarea compartida eRisk [1] propone conjuntos de datos para la detección de anorexia, depresión y signos de autolesión a partir de textos en medios sociales.

Sin embargo, no en todas las aplicaciones el momento de la intervención es crítico, y la predicción temprana ha sido utilizada para modelar otros tipos de problemas también. Un ejemplo es el trabajo de Yu et al. [2017], donde el objetivo es clasificar una lectura de texto como menos palabras, no sólo parando antes sino también saltando algunas de ellas. El modelo utiliza un RNN que aprende a predecir el número de pasos de salto después de leer uno o varios tokens de entrada. Como este modelo es discreto y por lo tanto no totalmente diferenciable, la red está entrenada usando técnicas de Aprendizaje de Refuerzo.

La dificultad de la tarea de predicción temprana es decidir cuándo los datos procesados son suficientes para una buena predicción. Por un lado, el prefijo de la secuencia leída tiene que ser

[1] urlhttp://early.irlab.org/

lo más corto posible, pero por otro lado, si el prefijo es demasiado pequeño, el modelo puede no tener suficiente información para dar una respuesta correcta.

En el escenario EDM, este paradigma de clasificación ha sido utilizado para predecir la frustración de los estudiantes y el estrés sobre los ITSs [McQuiggan et al., 2007]. Este trabajo explora modelos secuenciales basados en n-gramas y modelos no secuenciales como Naïve Bayes, árboles de decisión y support vector machines. Para inferir el estado del estudiante, frustrado o no, se utilizan varias características: el tiempo transcurrido, la actividad del estudiante, el progreso en la plataforma de aprendizaje y una serie de señales fisiológicas recogidas de un aparato de *biofeedback* conectado a la mano del estudiante.

La predicción de deserción es también una tarea ideal para este tipo de enfoques, especialmente porque la detección del abandono es menos útil cuando el curso analizado ha terminado, donde no se puede tomar ninguna acción para prevenirla. La aplicación del modelo obtenido en un curso a otra repetición subsiguiente del mismo puede dejar algunos factores importantes particulares a esa cohorte de estudiantes, sin mencionar que restringe los cambios que se pueden hacer de una versión a la siguiente.

En segundo lugar, existen algunas limitaciones técnicas al utilizar un RNN para una tarea de clasificación de secuencias. La intensidad de la señal causada por la etiqueta final de la secuencia es difícil de propagar hasta el inicio de la secuencia. Este problema es comúnmente llamado *vanishing gradient*, y aunque existen arquitecturas modernas diseñadas para superar este problema, en la práctica no siempre es posible. Como resultado, si el modelo es entrenado con la secuencia completa de acciones del estudiante hasta el final del curso, aprenderá a predecir fuertemente basado en las últimas acciones del estudiante. Esto hace que el modelo dependa de tener disponible toda la secuencia de acciones, lo que no es posible mientras el curso esté en línea. Podríamos evitar esta situación con una configuración de predicción temprana, donde el modelo está entrenado para detectar patrones con porciones parciales y más pequeñas de la secuencia de acciones del estudiante.

Sin embargo, no existe ningún trabajo que utilice directamente una técnica de detección temprana para predecir la deserción escolar, hasta donde sabemos.

En nuestra propuesta, inyectamos en el clasificador información del dominio particular de los entornos de aprendizaje. Por lo tanto, es más probable que el modelo capture información con menos datos disponibles que un modelo de vainilla. Esto puede ayudar potencialmente a predecir con precisión secuencias más cortas. Por esta razón, proponemos también evaluar el impacto de los co-embeddings en la tarea de predicción de deserción, con un enfoque de detección temprana.

En este caso, utilizaremos sólo información parcial de la secuencia para predecir si se registrará alguna actividad del estudiante en la semana siguiente. Es posible entrenar y utilizar este

modelo a partir de la segunda semana del curso. Además, el modelo puede ser re-entrenado cada semana para incorporar la nueva información recopilada.

Vale la pena señalar que este no es un escenario de predicción temprana pura, porque el modelo no está aprendiendo cuándo hacer una predicción, sino sólo cómo hacer una predicción con información parcial disponible. Para evaluar realmente un enfoque de detección temprana, se debe utilizar una estrategia de aprendizaje diferente. En el Apéndice A describimos una posible solución utilizando el Aprendizaje de Refuerzo Profundo para optimizar no sólo la precisión de la clasificación, sino también cuando el clasificador decide emitir una advertencia de posible abandono.

## 3.3 Resumen de la propuesta

Ambas áreas, KT y predicción de deserción escolar, modelan el comportamiento de aprendizaje, pero desde diferentes perspectivas. KT se basa en modelos teóricos, mientras que la deserción escolar es más empírica y se centra en el escenario. Ante esto, proponemos utilizar un marco unificador que pueda ser aplicado en diferentes tareas de EDM.

Proponemos aprender una representación de los elementos y agentes de un escenario de aprendizaje basado en co-embeddings neuronales, utilizando el mismo tipo de arquitectura para ambos escenarios. Desde un punto de vista teórico, el modelo podría portarse a otras tareas de EDM. En el siguiente capítulo detallamos los aspectos técnicos de esta arquitectura, luego procedemos a describir nuestro entorno experimental y mostrar sus resultados.

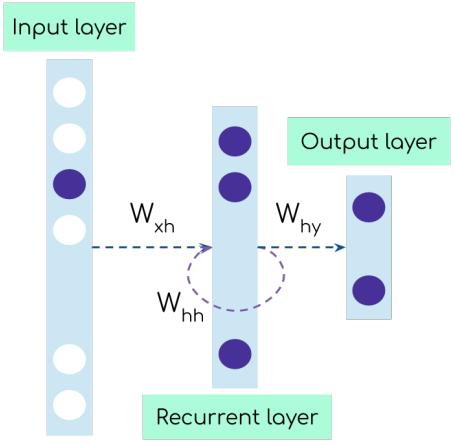
# Capítulo 4

# Co-embeddings neuronales para el modelar estados de estudiantes y contenidos

En este capítulo describiremos en detalle una arquitectura neural con capas recurrentes propuesta para optimizar las funciones de embeddings $\varphi_S$ y $\varphi_E$. Más adelante, describiremos cómo modificar la arquitectura para que se ajuste a los dos problemas particulares de clasificación de KT en ITS y la predicción de abandono en MOOCs.

El modelo propuesto se basa en el trabajo de Piech et al. [2015]. Consiste en una red neuronal con una capa de entrada, una capa recurrente y una capa de salida. Cada capa toma la salida de la capa anterior, la transforma con una combinación no lineal y genera una nueva salida. La figura 4.1 muestra una vista esquemática de una arquitectura general recurrente. En las siguientes secciones explicaremos las modificaciones a la estructura de base.

Figure 4.1: Arquitectura de una RNN básica



Algunas de las propiedades de RNN ya fueron introducidas en la sección 3.1.4. En resumen, las RNNs tienen un estado interno que hace un seguimiento de la entrada vista previamente. Como resultado, pueden utilizarse para procesar información secuencial, donde la interpretación de un elemento también depende de los elementos precedentes de la secuencia. El estado oculto es capaz de recordar selectivamente aspectos relevantes de la entrada procesada hasta el momento.

Para encontrar los parámetros del modelo, la red es entrenada con un proceso de optimización iterativo. Usaremos los conjuntos de secuencias $I_s$ para todos los estudiantes $s \in S$ para entrenar al RNN como modelo de clasificación, minimizando la pérdida de clasificación de las salidas de las interacciones (conocidas). Cada ejemplo de entrenamiento es una tupla con una representación vectorial de la interacción en el momento $t$, denominada $x_t$, y la representación vectorial de la salida correspondiente, denominada $o_t$.

Para el entrenamiento, tomaremos una secuencia de estudiantes a la vez y alimentaremos al clasificador con los ejemplos $(x_t, o_t)$ en el orden correcto. El clasificador predecirá una salida $y_t$ para cada entrada, llamada el paso *forward*. Una vez completada la secuencia, calcularemos el error entre la salida pronosticada $y_t$ y la salida real $o_t$, llamada función de pérdida. Minimizaremos el valor de la función de pérdida utilizando el algoritmo BackPropagation Through Time (BPPT), diseñado para la optimización de RNN. Esto actualizará los parámetros del RNN utilizando el descenso de gradiente estocástico o un algoritmo similar. El proceso se repite con todas las secuencias de estudiantes disponibles hasta que la red converge.

## 4.1 Capa recurrente

El vector de entrada $x_t \in \mathbb{R}^m$, en este caso las interacciones, son alimentadas a la red a través de la capa de entrada. Lo que caracteriza a un RNN es que el nuevo estado oculto se calcula utilizando la información de la capa de entrada y del estado oculto anterior $h_{t-1}$. Entonces, la salida $o_t$ se calcula como una transformación del estado oculto $h_{t-1}$.

Las ecuaciones de actualización específicas para una RNN básica son las siguientes:

$$\begin{aligned} y_t &= \sigma(W^{(hy)} h_t + b^{(y)}) \\ h_t &= tanh(W^{(xh)} x_t + W^{(hh)} h_{t-1} + b^{(h)}) \end{aligned} \tag{4.1}$$

Con esta definición, los parámetros finales del modelo se describen en la Tabla 4.1.

Table 4.1: Parámetros de una red neuronal recurrente

| Parámetro | Descripción | Dimensiones |
|---|---|---|
| $W^{(xh)}$ | Matriz de pesos de la capa de entrada al estado oculto | $\mathbb{R}^{m\times n}$ |
| $W^{(hh)}$ | Matriz de pesos de el estado oculto anterior al nuevo estado oculto | $\mathbb{R}^{n\times n}$ |
| $W^{(hy)}$ | Matriz de pesos de el estado oculto actual de la capa de salida | $\mathbb{R}^{n\times o}$ |
| $b^{(h)}$ | Vector de bias añadido al estado oculto | $\mathbb{R}^{n}$ |
| $b^{(y)}$ | Vector de bias añadido a la capa de salida | $\mathbb{R}^{o}$ |

## 4.2 Integración de embeddings en la arquitectura RNN

Siguiendo a [Piech et al., 2015], interpretaremos el estado oculto después de ver la secuencia de entrada $\{(x_t, o_t, o_t)|t \leq T\}_S$ del clasificador RNN como el embedding del estudiante en el momento $T + 1$. Una vez entrenados, la red ha aprendido qué información de la entrada es útil mantener durante el tiempo, y una codificación de dicha entrada se registra en el estado oculto. El estado oculto $h$ es nuestra función $\varphi_S$ que toma, como se define en la Sección 2, la representación de la interacción y el embedding del estudiante en el tiempo anterior.

Para integrar el embedding $\varphi_E$ en el clasificador RNN, sustituimos la capa de entrada por una nueva capa de embedding. Una capa de embedding no es más que una matriz que funciona como una tabla de búsqueda, donde la columna $j$ es el embedding del elemento con el índice $j$. El único cambio en la arquitectura que se muestra en la Figura 4.1 es que reemplazamos la primera capa con entrada esparsa por esta capa de embedding, generalmente densa y mucho más pequeña.

La ecuación 4.1 puede reescribirse como:

$$\varphi_S(s_{t-1}, \varphi_E(x_t))_t \triangleq h_t = tanh(W^{(xh)}\varphi_E(x_t) + W^{(hh)}h_{t-1} + b^{(h)} \tag{4.2}$$

En este punto tenemos una arquitectura que proyecta a los estudiantes y a los elementos del curso en vectores de la misma dimensionalidad, pero aún no hemos asegurado que pertenezcan al mismo espacio de habilidades latente. En otras palabras, no hemos dado el mismo significado a las dimensiones de dicho espacio. Para ello, no utilizaremos el elemento del curso embebido como una entrada directa a la capa oculta. En su lugar, calcularemos el nuevo estado oculto como una combinación del estado anterior y la distancia entre el embedding actual del estudiante y el embedding del elemento del curso.

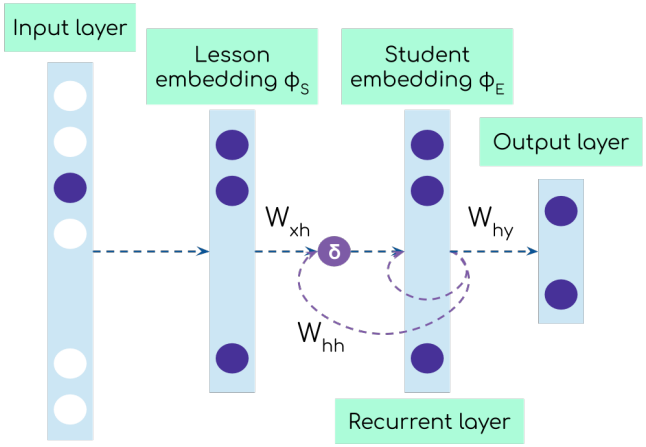
Las ecuaciones de actualización final que representan el modelo son:

$$y_t = \sigma(W^{(hy)}h_t + b^{(y)})$$

$$\varphi_S(s_{t-1}, \varphi_E(x_t))_t \triangleq h_t = tanh(W^{(xh)}\delta(\varphi_E(x_t), h_{t-1}) + W^{(hh)}h_{t-1} + b^{(h)}) \tag{4.3}$$

donde $\delta$ es la función puntual que calcula la distancia. Esta arquitectura se presenta en la figura 4.2.

Figure 4.2: Arquitectura RNN con co-embeddings




Proponemos utilizar una función $\delta$ para incluir en los embeddings de los elementos del curos la intuición de Reddy et al.. En este escenario, si los estudiantes están lo suficientemente cerca del elemento del curso, entonces obtendrán el mayor beneficio posible de interactuar con el elemento del curso. Si están demasiado lejos, entonces la lección tendrá un contenido demasiado fácil o demasiado difícil para esta interacción. Podemos ver esto reflejado en la Ecuación 4.3. El nuevo estado del estudiante es la suma de dos partes, el estado anterior y el embedding de la lección. Cuando el valor de $\delta(\varphi_E(x_t), h_{t-1})$ es pequeño, el estado del estudiante cambia menos y la lección particular de la interacción tiene poco impacto en el nuevo estado.

En necesario tener en cuenta que, si la tarea de pretexto es distinta de Knowledge Tracing, la interpretación de Reddy et al. no es válida. Las RNN son agnósticas al significado de la distancia entre el estudiante y los embeddings de los elementos del curso. No obstante, optimizará ambos para la tarea de pretexto. Podemos usarlo como un método general que determinará qué aspectos de la lección deben impactar más en el embedding del estudiante. La representación de los estudiantes, a su vez, será mejorada para traer la información necesaria para resolver la tarea de pretexto. En el caso de la predicción de deserción escolar, por ejemplo, los factores latentes estarán más relacionados con la participación que en KT.

El aspecto puntual de la función también es importante, ya que queremos distinguir la influencia de cada dimensión en el estado resultante. Esta es una distinción importante entre nuestro modelo y Reddy et al. [2016], que utiliza la longitud de la proyección del embedding del estudiante sobre el embedding de la lección. Como resultado, nuestro enfoque no compensa un pequeño valor en una dimensión (habilidad) con un gran valor en otra, sino que preserva la relación entre el estudiante y el elemento del curso para cada dimensión.

Proponemos las siguientes de funciones de *delta*:

$$\delta(x, y) = (x - y)^2 \quad \text{(sq)}$$

$$\delta(x, y) = |(x - y)| \quad \text{(abs)}$$

$$\delta(x, y) = norm(x - y, 0, std) \quad \text{(norm)}$$

$$\delta(x, y) = sigmoid(x - y) \quad \text{(sigmoid)}$$

$$\delta(x, y) = tanh(x - y) \quad \text{(tanh)}$$

Todas las operaciones son punto a punto. En la última función, *norm* se refiere a la probabilidad de que el vector siga una distribución normal centrada en 0 y con desviación estándar *std*.

Las funciones *sq* y *abs* se comportan de forma similar, proyectando valores positivos y negativos en el mismo rango. La función *sq* maximiza valores que son demasiado grandes, lo que corresponde a las dimensiones en las que los elementos del curso y los estudiantes están muy alejados unos de otros.

La función *norm* sólo toma valores en el intervalo[0, 1], siendo cercana a 1 en las dimensiones donde los elementos del curso y los estudiantes están cerca, y cayendo rápidamente a 0 cuando se separan.

Las funciones *sigmoid* y *tanh* tienen una forma similar, la primera en el intervalo $[0, 1]$ y la segunda en el intervalo $[-1, 1]$. Estas dos funciones guardan información sobre el signo de la diferencia entre el elemento estudiante y el elemento del curso. Los valores extremos se introducirán en un extremo de cada rango.

Las dos primeras definiciones son adecuadas para el rastreo del conocimiento, donde las dimensiones del espacio compartido se interpretan como el nivel de conocimiento de los estudiantes en ciertas habilidades latentes (aunque es probable que la red también esté modelando otros aspectos). Aplicando *sq* y *abs* a la diferencia entre estudiantes y lecciones, se da menos importancia a las habilidades donde el estudiante está más cerca de los prerrequisitos de la lección, en cuyo caso el éxito es más seguro. Disponer de información detallada sobre la distancia que separa al alumno de la lección puede ayudar al clasificador a discriminar mejor los casos menos certeras.

Para la tarea de predicción de abandono, esperamos que las dimensiones del espacio latente estén más relacionadas con los niveles de compromiso. El escenario más probable es que la deserción escolar no esté vinculada a una sola lección, sino a un largo proceso de retirada. Todas las funciones presentadas podrían presentar patrones importantes, maximizando las diferencias cuando las lecciones y los estudiantes están cerca o lejos. Esperamos obtener alguna información basada en su impacto en el rendimiento, si es que hay alguna diferencia entre ellos. No

debemos olvidar que los modelos neurales no son interpretables, y en consecuencia cualquier interpretación del espacio latente es sólo especulativa y no descriptiva.

### 4.2.1 Embeddings pre-entrenados

Una posible variación en este modelo es inicializar la capa de embedding inicial con embeddings de elementos de curso pre-entrenados obtenidos con un método no supervisado. En otras palabras, podemos obtener una representación general de los elementos del curso utilizando sólo la secuencia de elementos que ven los estudiantes.

Los embeddings pre-entrenados tienen la ventaja de incluir información sobre el orden en el que los estudiantes acceden a los elementos del curso, utilizando algoritmos especializados para este tipo de tareas. Caracterizan los elementos del curso en función de su contexto, compuestos por otros elementos que fueron vistos de cerca en la secuencia, antes y después. De hecho, métodos como word2vec [Mikolov et al., 2013] y GloVe [Pennington et al., 2014] han demostrado modelar adecuadamente la distribución subyacente de secuencias de eventos a partir de muestras incompletas. Estos modelos parecen capturar causas latentes que explican comportamientos observables, como la semántica de palabras en el modelado del lenguaje.

Esperamos que los modelos pre-entrenados capturen aspectos de los conjuntos de datos pertinentes a la relación entre los elementos del curso independientemente de su impacto en la tarea de pretexto. En [Pardos and Horodyskyj, 2017], el autor modeló secuencias de estudiantes usando sólo word2vec y analizó la utilidad de las visualizaciones de esos embeddings, evaluados por los creadores del curso. Los resultados muestran que la organización de alto nivel del contenido del curso fue capturado por los embeddings, e incluso agruparon a los estudiantes exitosos y a los que no lo fueron.

Además, estos embeddings pueden optimizarse o afinarse junto con la optimización del modelo para la tarea de pretexto. Si no se permite el ajuste fino, el impacto de los embeddings pre-entrenados en los embeddings de los alumnos será mayor. Otra ventaja importante es que los embeddings pre-entrenados, como método no supervisado, también pueden ser entrenados con instancias del mismo dominio pero sin etiquetas para la tarea de pretexto.

Hasta ahora hemos presentado un método general que se puede aplicar a muchas tareas, con las correspondientes alteraciones. En las dos secciones siguientes describiremos la configuración particular del modelo cuando se aplica a Knowledge Tracing y a predicción de deserción escolar.

### 4.2.2 Co-embeddings neuronales para Knowledge Tracing

Para la tarea de rastreo del conocimiento, la red RNN se utilizará para producir un resultado para cada elemento de la secuencia. Esta configuración también se denomina etiquetado de secuencia.

La función de pérdida seleccionada es el promedio de la entropía cruzada binaria o *log loss* entre la predicción del modelo y la etiqueta real en todos los ejemplos. Para la tarea KT, para calcular la pérdida de un estudiante sumamos la entropía cruzada de todas las interacciones de la secuencia, como en Piech et al. [2015]. Si definimos el índice de ejercicio en la interacción en el momento $t$ como $e_t$, la pérdida para un estudiante es:

$$loss = \sum_t log(y_{t,e_t})o_t + log(1 - y_{t,e_t})(1 - o_t) \tag{4.4}$$

La representación de los datos de entrada también se modifica con respecto al modelo base. Esta tarea proporciona dos valores para cada interacción: la identificación del ejercicio y si el estudiante lo resuelve con éxito en el primer intento. Para modelar ambos aspectos, usamos la suma de dos embeddings para cada ejercicio: una que corresponde al ejercicio base, y otra para el resultado exitoso de la interacción. Si el alumno no resuelve el ejercicio en el primer intento, sólo se utiliza el embedding base.

### 4.2.3 Co-embeddings neuronales para predicción de deserción

Las incrustaciones y co-incrustaciones neuronales pueden ser útiles también para tareas menos definidas como la predicción de la deserción escolar. En este caso, no esperamos que modelen sólo el conocimiento, sino principalmente otras variables relacionadas con el compromiso.

No es necesario para esta tarea utilizar ninguna otra información que no sea la identificación del elemento del curso visitado en ese momento. La capa de entrada sólo toma este id, pero si hay más información disponible, puede ser adaptada para tomar múltiples valores como en el caso de KT.

Hay dos configuraciones posibles en una red neuronal para predecir una variable binaria como el abandono: calcular sólo la probabilidad de la capa positiva, o calcular la probabilidad de ambas clases como una tarea de clasificación multi-etiqueta. Esto afecta al tamaño de la capa de salida, ya que se necesita una neurona por clase de salida. La función de pérdida es en ambos casos el promedio de la entropía cruzada para todas las secuencias del conjunto de datos de entrenamiento.

## 4.3 Discusión

En este capítulo hemos detallado los aspectos técnicos de nuestra propuesta para aprender representaciones conjuntas de estudiantes y elementos del curso. Se basan en la interpretación de los estados ocultos. Después de Piech et al. [2015], este estado oculto captura los cambios en el estado del estudiante. Nuestra propuesta amplía la de Piech al relacionar explícitamente estos cambios con las contribuciones que los diferentes elementos del curso pueden hacer o han hecho en los estados de los estudiantes.

Por último, hemos desarrollado cómo se utiliza la misma arquitectura para la predicción de KT y de la deserción escolar, que son tareas con diferentes configuraciones de clasificación. En los capítulos siguientes evaluamos el impacto de este modelo en estas dos tareas.

# Capítulo 5

# Experimentos

En capítulos anteriores, hemos presentado el problema del modelado de estudiantes en EDM, introducido un modelo teórico para abordarlo, y dos posibles tareas para evaluar el impacto de los co-embeddings. Durante este capítulo, desarrollaremos un entorno experimental para obtener resultados que puedan responder a las preguntas de investigación propuestas en la Sección 2.6. Hay dos conjuntos principales de experimentos que realizar, uno para cada tarea.

Los primeros temas a considerar son los aspectos prácticos de cómo construir y entrenar los modelos presentados en las Secciones 4.2.2 y 4.2.3. La optimización de las redes neuronales es una tarea no trivial y, muy a menudo, la mayor barrera a superar para tener un modelo exitoso. Presentamos aspectos clave del proceso de optimización que deben tenerse en cuenta para entender los resultados de los experimentos. Esto incluye diferentes hiperparámetros que definen la red, el proceso para seleccionarlos y, finalmente, la optimización iterativa que se debe seguir para encontrar parámetros de buen desempeño.

También es importante diseñar cómo se evaluarán y compararán los modelos. Esto incluye no sólo las métricas de evaluación, y lo que captan y destacan, sino también la relevancia de los resultados. En el aprendizaje profundo, es práctica común mostrar sólo el mejor resultado después de una búsqueda muy extensa de hiperparámetros, sin mostrar la distribución de los valores obtenidos o dar una intuición de por qué una arquitectura determinada debe funcionar, basada en el conocimiento del dominio o del problema.

La mayoría de estos aspectos son comunes a ambas tareas, KT y predicción de deserción escolar. En los casos en que no lo sean, se explicará en qué se diferencian los modelos y las consideraciones especiales que se deben tener en cuenta. En la última sección se detallan los experimentos a realizar, explicando las diferentes implementaciones a comparar.

## 5.1 Proceso de optimización

La mayoría de los modelos de aprendizaje de máquina se entrenan usando algún algoritmo iterativo que minimiza una función de pérdida, como se explica en la sección 2.5.1. Sin embargo, el desempeño del clasificador no se mide a través de la función de pérdida, sino a través de alguna otra métrica que sea más adecuada para describir el problema. Se espera que la minimización de la pérdida del modelo aumente su rendimiento, pero no se garantiza la velocidad a la que aumenta. Además, se desconoce la verdadera distribución de probabilidad del problema, sólo aproximada por una muestra incompleta de ejemplos. Según Goodfellow et al. [2016], esa es la principal diferencia entre el entrenamiento (para modelos de aprendizaje automático) y la optimización pura de problemas.

> "Los algoritmos de aprendizaje profundo implican optimización en muchos contextos" [Goodfellow et al., 2016, p. 274].

Existen dos procesos de decisión principales a la hora de entrenar redes neuronales: la selección de hiperparámetros y la optimización de parámetros. Ambos tienen un impacto crítico en el rendimiento final.

Otros algoritmos de aprendizaje automático tienen un pequeño conjunto de hiperparámetros que los describen. Esto permite aplicar procesos de selección basados en la exploración de todas las combinaciones mediante búsqueda en grillas. Pero las redes neuronales tienen tantas configuraciones posibles que es necesario aplicar mejores estrategias. Además, con el paso de los años, se proponen cada vez más modificaciones y nuevas arquitecturas, por lo que el número de combinaciones sigue aumentando.

La selección de los parámetros tampoco es sencilla. La función de pérdida es altamente no convexa, con muchos mínimos locales, regiones planas y puntos de silla (*saddle-points*). También hay otros problemas relacionados con el mal estado de las funciones hessianas, explosiones y gradientes inexactos, irregularidades en la función de pérdida, entre otros, descriptos por Goodfellow et al. [2016]. Estas propiedades, junto con el uso del descenso de gradiente estocástico a partir de un punto de partida inicializado aleatoriamente, no garantizan que una configuración dada produzca resultados comparables cada vez que se ejecuta. Es necesario realizar varios experimentos idénticos para tener en cuenta las exploraciones atípicas.

El último factor que afecta a la formación de los modelos de aprendizaje profundo es la falta de una comprensión teórica sólida de los mismos. Hasta donde sabemos, el por qué las redes neuronales funcionan tan bien y qué puede mejorarlas sigue siendo una cuestión sin resolver en el área, aunque muchos trabajos se han ocupado de ella. Esto también conduce a una falta de certeza sobre si el modelo obtenido es el mejor modelo posible.

Como resultado, el proceso de formación se basa en gran medida en probar cientos, si no miles, de redes diferentes y seleccionar las mejores. Hay reglas generales a seguir y recomendaciones de lo que funciona mejor para los diferentes dominios, pero el última instancia, sigue siendo una rutina de prueba y error. En las siguientes secciones describiremos algunos conceptos clave a tener en cuenta mientras se entrenan las redes neuronales, y nuestro procedimiento experimental.

### 5.1.1 *Overfitting* y *underfitting*

El entrenamiento es un proceso propenso al *overfitting* o sobreajuste, una situación en la que el modelo funciona muy bien en el conjunto de datos de entrenamiento, pero mal en los ejemplos no vistos. Esto sucede porque el modelo tiene demasiada capacidad de representación, suficiente para memorizar el conjunto de datos de entrenamiento sin ninguna generalización. Por el contrario, el modelo debe ser capaz de aprender eficazmente y capturar buenas generalizaciones, abordando así las causas ocultas de los fenómenos que se manifiestan en los ejemplos. Si esto no ocurre, se produce el problema opuesto, es decir, la falta de adaptación o *underfitting*.

La optimización de modelos es un intercambio entre estos dos conceptos. Para los modelos de aprendizaje profundo en particular, es más común experimentar el sobreajuste, dado el gran número de parámetros que tienen, lo que significa que son muy complejos y con una gran expresividad.

Se debe tener una correlación entre la complejidad del modelo y la complejidad del problema que se está resolviendo [M. Bishop, 1995]. En este caso, el proceso de aprendizaje humano tiene muchas causas ocultas, pero no todas ellas son capturadas en los conjuntos de datos educativos. Puede faltar información crucial de los conjuntos de datos para representar, por ejemplo, los antecedentes sociales. Al elegir una arquitectura de aprendizaje profundo, debemos añadir suficientes parámetros para modelar tantas causas ocultas como sea posible capturar de la información dentro del conjunto de datos, pero no muchas más, para evitar sobredimensionar los ejemplos.

En las siguientes secciones, describiremos diferentes aspectos del proceso de optimización con un enfoque en la prevención del sobreajuste sin disminuir el poder de representación de la red neuronal.

#### La importancia del conjunto de datos de validación

El overfitting suele estar relacionado con la complejidad intrínseca del modelo durante la selección de sus parámetros. Sin embargo, en las redes neuronales, también es posible el sobrea-

juste indirecto, durante la selección de los hiperparámetros.

Durante el proceso de exploración se probarán muchos modelos diferentes. Los hiperparámetros se elegirán progresivamente para maximizar el rendimiento en un conjunto de datos reservado. Este uso del conjunto de datos reservados implica que, eventualmente, no habrá ninguna porción de datos para evaluar el verdadero poder de generalización del modelo sobre los datos no vistos. Es de vital importancia en el aprendizaje profundo utilizar un conjunto de datos de validación para la selección de hiperparámetros, y un conjunto de datos de prueba diferente para la evaluación del modelo después de que se haya determinado la arquitectura final.

### Tamaño del modelo

"La complejidad de un modelo de red neuronal está gobernada por el número de grados de libertad, que a su vez está controlado por el número de parámetros adaptativos (pesos y sesgos) en la red." [M. Bishop, 1995, p. 2]. El número de capas, y el número de neuronas en cada capa, determinará el poder de representación de la red. Como se explicó en las secciones anteriores, el modelo debe tener suficientes neuronas y capas para aprender el problema.

### Regularización

Regularizar un modelo implica reducir su complejidad, siguiendo el principio de la navaja de Occam, y puede ayudar a prevenir el sobreajuste Bhlmann and van de Geer [2011].

En los modelos de aprendizaje profundo, la regularización a menudo adopta tres formas: penalizaciones de la norma de los parámetros, capas de dropout y detención temprana. Goodfellow et al. [2016] menciona varios más, pero nos centraremos sólo en estos tres en este trabajo.

Las penalizaciones de la norma de parámetros añaden la norma del vector de parámetros a la función de pérdida. De esta manera, los parámetros con altas normas aumentan las pérdidas y se evitan durante el proceso de optimización. Por lo general, sólo se regularizan los pesos de las capas, excluyendo los sesgos. En términos generales, la regularización del peso reduce la varianza y desalienta los modelos demasiado complejos. Según M. Bishop [1995], los regularizadores de esta forma fomentan el buen funcionamiento de la red. El uso de normas diferentes para la regularización tendrá efectos diferentes sobre el modelo, pero un análisis más profundo excede el alcance de esta tesis.

Dropout es una técnica de regularización que asigna el valor de una celda a cero con una probabilidad $p$, independientemente de otras celdas. Esto sólo ocurre durante el entrenamiento. El objetivo es prevenir coadaptaciones entre diferentes neuronas que no se generalizan con datos no vistos. El dropout de las neuronas también puede ser visto como una forma de inyectar ruido

aleatorio en las capas ocultas del modelo, un conocido método de smoothing [Srivastava et al., 2014].

La parada temprana es la técnica de interrumpir el proceso de entrenamiento cuando ocurre algún fenómeno no deseado. Por ejemplo, la pérdida en el conjunto de datos de validación empieza a aumentar. Otro criterio es entrenar sólo durante iteraciones $n$ más a partir de que la pérdida en el conjunto de validación deja de disminuir, lo que también se conoce como el hiperparámetro de la paciencia. Este proceso no sólo evita el sobreajuste, sino que también reduce el tiempo y los costes de entrenamiento. En algunos aspectos, es comparable a la regularización de las penas de la norma. No obstante, se recomienda utilizar varios métodos de regularización independientes cuando se entrena a las redes neuronales M. Bishop [1995].

### 5.1.2 Descenso por el gradiente estocástico y otros optimizadores

Para entrenar las redes neuronales necesitamos minimizar la función de pérdida. El método de optimización más común es el Descenso de Gradiente Estocástico (SGD) o algún algoritmo derivado, como Adam o RMSEProp El algoritmo SGD se basa únicamente en el cálculo del gradiente, sin necesidad de la segunda derivada. Es más lento que otros métodos, pero en la práctica funciona bien.

En términos generales, SGD actualiza los parámetros del modelo de manera iterativa buscando disminuir el valor de la pérdida con cada iteración. El proceso se detiene cuando el modelo converge y los valores de pérdida no cambian significativamente.

Para disminuir la pérdida, necesitamos mover los parámetros del modelo en cualquier dirección en que la función de pérdida vaya hacia "abajo", llamada dirección de descenso. En particular, lo contrario del gradiente es la dirección de mayor disminución de la función de pérdida. SGD o *stochastic gradient descent* siempre utiliza esta dirección para actualizar los parámetros. El grado de actualización de los parámetros es controlado por un hiperparámetro llamado velocidad de aprendizaje *learning rate*.

En SGD, el ritmo de aprendizaje se mantiene constante durante toda la optimización. Este valor tiene un impacto crítico en el entrenamiento del modelo: si es demasiado pequeño, las actualizaciones de los parámetros serán cortas y se necesitarán muchas iteraciones. Si el valor es demasiado grande, entonces es posible que el modelo se desvíe y "pase por encima" del punto mínimo. En otros algoritmos, como Adam, la velocidad de aprendizaje se adapta durante la optimización.

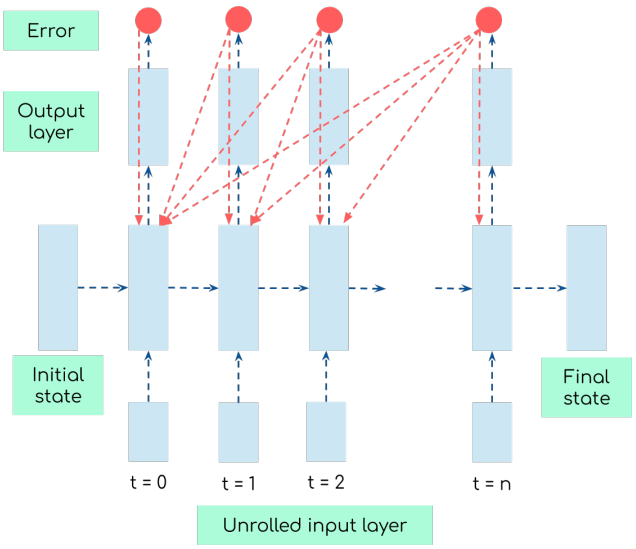
El cálculo del gradiente de la pérdida con respecto a cada parámetro de la red no es una tarea sencilla. Un algoritmo especial, llamado *Backpropagation*, fue desarrollado por Rumelhart et al.

[1988]. Este algoritmo calcula la actualización de cada parámetro utilizando la actualización de los anteriores, reduciendo drásticamente el tiempo de cálculo.

### Retropropagación a través del tiempo

En arquitecturas recurrentes, el algoritmo de retropropagación (Backpropagation) no puede utilizarse directamente, ya que el número de pasos de la secuencia es desconocido de antemano. Una modificación llamada Backpropagation Through Time (BPTT) se utiliza para calcular los gradientes en estos casos. La red se desenrolla una vez por cada paso de tiempo, creando nuevas capas con los mismos parámetros. Los errores se acumulan a través de los pasos de tiempo para la actualización final de los parámetros.

Figure 5.1: Diagrama de propagación de errores sobre una RNN desenrollada.




Presentamos una intuición del algoritmo en la Figura 5.1[1], que ilustra cómo se propagan los errores y cómo afectan a los parámetros. Los errores generados por los últimos pasos de tiempo se resumen y se aplican a pasos de tiempo anteriores, porque se supone que todos los datos introducidos en el pasado contribuyen al error actual.

### Entrenamiento por mini-batchs

La pérdida de problemas de aprendizaje profundos, por ejemplo el de la ecuación 4.4, es la suma del error individual producido por el modelo en cada ejemplo del conjunto de datos de entrenamiento. Sin embargo, calcular predicciones para todos los ejemplos es prohibitivamente

[1]Todas las imágenes de BPTT están inspiradas en `https://r2rt.com/styles-of-truncated-backpropagation.html`

costoso, especialmente cuando tiene que ser recalculado para cada iteración. Además, muchos ejemplos serán redundantes y no contribuirán información adicional al gradiente.

El algoritmo SGD se llama estocástico porque aproxima el gradiente de la pérdida utilizando el error de uno o unos pocos ejemplos, llamado *batch*. El proceso de optimización divide ahora el conjunto de datos de entrenamiento en lotes y actualiza el modelo con cada uno de ellos. Una vez que se ha visto todo el conjunto de datos de entrenamiento, se completa una época. Si el modelo no ha convergido todavía, el conjunto de datos de formación se mezcla y el proceso comienza de nuevo.

El tamaño del lote también influye en la convergencia de la optimización. Si el tamaño del lote es pequeño, las iteraciones serán rápidas pero utilizando una estimación inexacta del gradiente. Por el contrario, el procesamiento de muchos ejemplos a la vez aprovecha las arquitecturas paralelas como las GPUs. Si el lote es demasiado grande, el modelo convergerá lentamente y también puede crear un cuello de botella si el lote no cabe en la memoria de la GPU.

#### Desvanecimiento y explosión de gradientes

En teoría, una RNN puede modelar casi cualquier problema secuencial. En la práctica, es un desafío aprender tal modelo de un conjunto de datos cuando los elementos de una secuencia se relacionan a través de grandes períodos de tiempo. Esto se debe a dos problemas de origen similar: *vanishing* y *exploding gradients*.

La función de pérdida de una red neural es un problema altamente no convexo. Además, puede tener secciones de acantilados con valores de pendiente extremos. Cuando el optimizador se encuentra con tales secciones, puede producir una gran actualización de los parámetros, haciendo que el modelo aterrice en una sección completamente diferente de la pérdida en lugar de seguir el acantilado [Goodfellow et al., 2016]. Esta situación se denomina explosión de gradientes. Para evitarlo, se puede utilizar el heurístico muy simple llamada *gradient clipping*: el valor del greadiente se corta a un máximo si es demasiado alto.

El problema del desvanecimiento de gradientes es más complejo de resolver. Su origen se encuentra en el algoritmo de retropropagación, que estima el impacto de cada neurona en el error final. Este impacto se diluye a medida que las neuronas se encuentran más lejos de la capa de salida. En redes neurales muy profundas, esto se convierte en un problema porque sin un gradiente fuerte, la actualización de los pesos es muy pequeña. Esto implica que en aquellas regiones alejadas de la capa de salida la red no está aprendiendo nada. Para redes recurrentes, Bengio et al. [1993] demostró que la magnitud de la derivado del estado recurrente en el momento $t$ con respecto al estado en el momento 0 disminuye exponencialmente a medida que aumenta el tiempo.

Como resultado, cuando el error en la capa de salida en el tiempo t depende de un evento que ocurrió muchas veces antes en la secuencia, el gradiente generado por el evento de origen será exponencialmente pequeño. Esto no significa que la red no sea capaz de capturarla, sólo que se necesitarán muchas iteraciones para aprenderla.

La celda LSTM ha sido diseñada para superar el problema de desvanecimiento de gradientes Hochreiter and Schmidhuber [1997], mediante la adición de una compuerta de olvido especializada, una compuerta de entrada y una compuerta de salida. Las compuertas utilizan funciones de activación tangencial hiperbólica y sigmoide. En términos generales, la celda LSTM vincula el valor del gradiente con la activación de la compuerta del olvido, permitiendo el flujo de información y evitando que el gradiente desaparezca.

### Retropropagación truncada

Por muy exitoso que sea, el uso de celdas LSTM no garantiza un aprendizaje eficiente y correcto en sí mismo. La red todavía necesita modelar las dependencias a largo plazo y aprender de las señales escasas de los datos. Cuando se entrena una LSTM es importante entender el alcance de las relaciones con el modelo, la fuerza de las señales y la capacidad de la red para capturarlas. Si las relaciones no son demasiado largas, entonces no hay necesidad de propagar más los gradientes porque no se encontrará nueva información. Hacerlo llevará mucho tiempo e incluso puede impedir que el modelo converja.

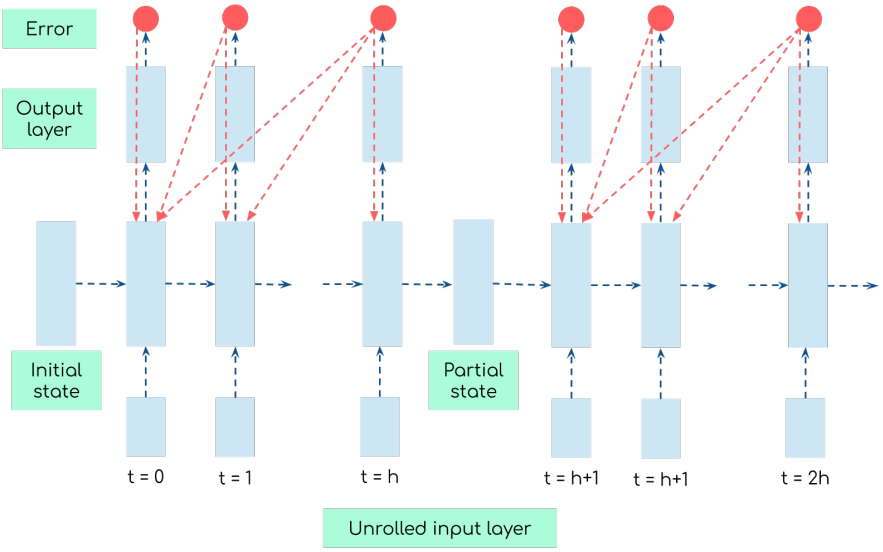
Williams and Peng [1990] propone el algoritmo *Truncated Backpropagation Through Time* (TBPTT) para optimizar el entrenamiento de redes recurrentes. En lugar de desenrollar la red hasta el tiempo cero, sólo desenrolla pasos de tiempo de $h$. Como los autores describen, este proceso inyectará un término de error en el cálculo de los gradientes, pero también evitará otros errores originados en la parte anterior de la secuencia. El hiperparámetro $h$ debe ser elegido de acuerdo con la tarea que se está modelando.

Evaluamos el rendimiento del TBPTT en las arquitecturas propuestas, con el objetivo principal de reducir los costes de formación. Sin embargo, utilizamos un enfoque diferente en el que la secuencia se divide en porciones de pasos temporales de $h$ y el error se propaga completamente en esa sección. Cuando se inicia una nueva sección de la misma secuencia, el último estado oculto de la capa recurrente de la sección anterior se utiliza como estado inicial. Una ilustración del proceso se presenta en la Figura 5.2.

### Etiquetado de secuencias contra clasificación de secuencias

Es importante enfatizar en este punto la diferencia entre tareas como la predicción de la deserción escolar y el Rastreo del Conocimiento. En la predicción de abandonos, se espera que el

Figure 5.2: Esquema de propagación de errores sobre una RNN desenrollada, con gradientes truncados.




clasificador procese toda la secuencia y emita un valor. Este escenario se llama *clasificación de secuencia*. El Rastreo del Conocimiento requiere un escenario diferente, llamado *etiquetado de secuencia*, donde se asigna una etiqueta a cada elemento de la secuencia.

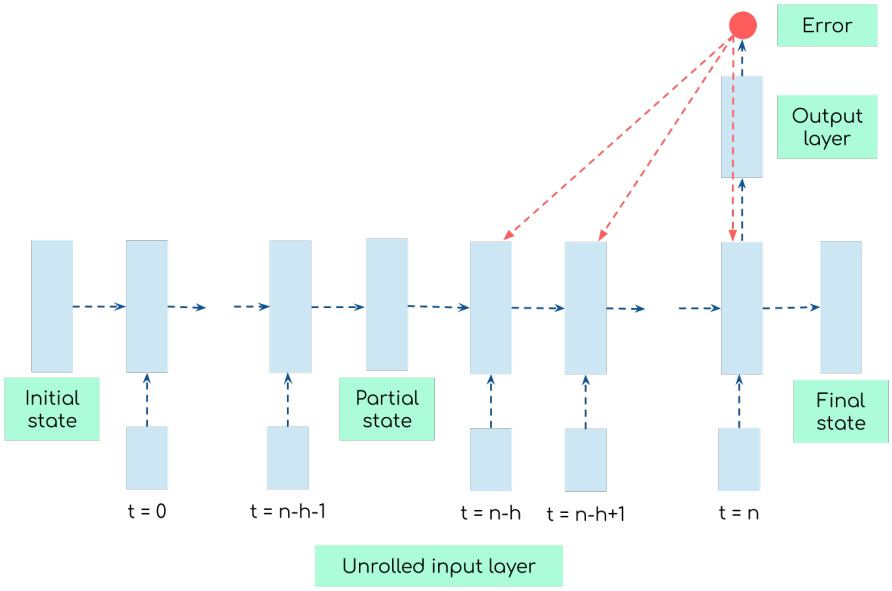
Esto tiene consecuencias decisivas durante la optimización. En la clasificación de secuencia, sólo hay una señal de error para propagarse a través de la red desenrollada, mientras que en el etiquetado de secuencia están presentes muchas señales de error. En ambos casos se puede utilizar el TBPTT, pero tiene un impacto diferente en la clasificación de la secuencia, como se muestra en la figura 5.3. Podemos observar que la red se utiliza para calcular todos los estados ocultos, pero el error sólo se propaga durante los pasos de tiempo de $h$, y los desdoblamientos anteriores no reciben ninguna actualización.

La pregunta que surge es, ¿cómo se pueden optimizar los pasos iniciales? En este sentido, tenemos la hipótesis de que, si se elige $h$ de tal manera que se procese una cantidad suficiente de secuencias desde el principio, el modelo aprenderá de esos ejemplos cómo representar los pasos iniciales del tiempo.

## 5.2 Entorno experimental

El objetivo principal de estos experimentos es responder a la pregunta de investigación 3, que aborda el impacto de los co-embeddings en el desempeño de cada tarea. Para ello, necesitamos comparar el rendimiento de los modelos con y sin co-embeddings, en el entorno para el que fueron diseñados. El desarrollo de este tipo de arquitecturas estaba dirigido a tareas en las que no había anotaciones manuales de conceptos de alto nivel, tales como componentes de

Figure 5.3: Diagrama de propagación de errores sobre una RNN desenrollada, con gradientes truncados, en un escenario de clasificación de secuencia.




conocimiento asociados con los ejercicios. Por lo tanto, tiene sentido evaluar el modelo en esos escenarios, incluso si el conjunto de datos incluye más información.

Es importante señalar que no pretendemos superar el estado del arte en estas tareas, aunque las compararemos con ellas cuando sea posible. Nuestro objetivo final es proporcionar una buena representación de los fenómenos, y el impacto de dicha representación en una aplicación como ésta se toma como un indicador de que la representación está capturando y generalizando los fenómenos adecuadamente.

Proponemos comparar el rendimiento de tres modelos, que detallaremos en el siguiente apartado:

- (**LSTM**) Una red neuronal recurrente con celdas LSTM. Esto equivale a la propuesta de Piech et al. [2015]. El embedding del estudiante es el estado oculto de la red, y no hay ningún embedding para los elementos del curso.
- (**E-LSTM**) Una red neuronal recurrente con una capa de embedding para los elementos del curso. Sin embargo, los estudiantes y los elementos del curso no están obligados a compartir el mismo espacio. Por lo tanto, llamamos a este enfoque "embeddings disjuntos".
- (**CoE-LSTM**) Una red neuronal recurrente con una capa de embedding para los elementos del curso, donde comparten el mismo espacio con la embedding del estudiante 4.

Se selecciona el modelo con el mejor puntaje de validación y se reportan los resultados sobre el conjunto de datos de evaluación.

### 5.2.1 Implementación de las arquitecturas propuestas

Todas las arquitecturas neuronales que probamos tienen una capa recurrente. La más simple (LSTM) tiene una sola capa de celdas LSTM. La red procesa las secuencias de los estudiantes una interacción a la vez. Los modelos están implementados en la biblioteca de Tensorflow 1.0 para Python [2].

La entrada es una representación *one-hot encoding* del elemento del curso involucrado en la interacción. Si se utiliza más de un aspecto del elemento, como el id y el tipo de interacción, cada uno se codifica individualmente como un vector *one-hot encoding* y posteriormente se concatenan en un único vector.

La capa recurrente se conecta a una capa de dropout, y más tarde a la capa densa que calcula el resultado final de la red. En todos las capas se aplica una regularización L2. El tamaño y la activación de la capa de salida está determinado por la tarea a resolver. Para KT, hay una neurona de salida para cada posible ejercicio, con una activación sigmoide. Para la predicción de deserción, la capa de salida está compuesta por dos neuronas con activación softmax, una para cada clase.

El segundo modelo (E-LSTM) tiene la misma estructura que el modelo LSTM, pero la capa de entrada se sustituye por una capa de embeddings. La capa de embeddings es seguida por una capa de dropout identica a la capa de dropout posterior. En la predicción de dropout, la capa de embeddings es sólo una matriz en la que las columnas corresponden a representaciones del elemento de curso. En KT, como se indica en la sección 4.2.2, la capa de embeddings es más compleja e implica varias operaciones. Para esta última tarea, los ejemplos de entrada son una concatenación de los one-hot encodings para el id de módulo y el id de evento. También experimentamos usando sólo la id del módulo, con resultados consistentemente peores.

El modelo CoE-LSTM final es el que se describe en la sección 4, implementando los co-embeddings propuestos. Tiene las mismas capas de embeddings, dropout y salida que el E-LSTM. La única diferencia es la aplicación de la función $\delta$ a la salida de la capa de embeddings antes de que entre en la capa recurrente.

Para los dos últimos modelos, también exploramos el uso de embeddings pre-entrenados, como se desarrolló en 4.2.1. Para obtener dichos embeddings usamos las secuencias de entrenamiento como entrada al algoritmo SkipGram de word2vec. Los parámetros utilizados son: tamaño de la ventana de 5 eventos, frecuencia mínima de 5 eventos, una tasa de aprendizaje inicial alfa de 0,01 y un muestreo negativo de 5 ejemplos. Se exploran ambos parámetros con y sin re-ajuste (*finetunning*).

[2] `https://www.tensorflow.org/`

### 5.2.2 Métricas

Para evaluar el rendimiento de los clasificadores necesitamos algunas métricas que midan la diferencia entre el resultado del modelo, es decir, la probabilidad de aprobar un ejercicio, y el resultado real de la interacción. Como el clasificador devuelve un resultado probabilístico, también podemos utilizar métricas de regresión para penalizar tanto una predicción incorrecta como una incierta. Las métricas más relevantes utilizadas son:

#### Precisión

La precisión (*accuracy*) es una métrica de clasificación estándar. Mide cuántos de los ejemplos dados fueron clasificados correctamente. Sin embargo, el resultado de los modelos propuestos es probabilístico, no categórico. Para hacer una clasificación binaria a partir de ellos, es necesario definir un umbral de corte para decidir si el ejemplo es positivo o negativo. En este trabajo, siempre hemos utilizado un umbral de 0,5, ya que la métrica ROC de la AUC tiene en cuenta otros posibles valores.

La precisión nos permite ver qué tan bien que se desempeña el clasificador, a costa de perder información de la salida probabilística. Por esta razón, lo consideraremos sólo como un indicador aproximado y compacto, confiando en la métrica ROC de AUC para un análisis más fino.

#### AUC ROC

El Área Bajo Curva de la Característica Operativa del Receptor (AUC ROC por sus siglas en inglés, o simplemente AUC) es una métrica de clasificación para clasificadores probabilísticos, en tareas de clasificación binaria. La métrica de la curva ROC traza la Tasa de falsos positivos (FPR) en función de la Tasa de Verdaderos Positivos (TPR), para varios umbrales posibles de la clase positiva. La TPR es la proporción de ejemplos positivos que se clasificaron correctamente. La FPR es la proporción de ejemplos negativos que fueron incorrectamente clasificados como positivos. El análisis de los diferentes umbrales contempla todos los casos posibles, por ejemplo, clasificadores estrictos que sólo asignan la clase positiva cuando el modelo predice con una probabilidad superior a 0,95, o modelos relajados que asignan la clase positiva con una probabilidad mínima de 0,2.

Para dar un ejemplo, tomemos como clasificador un generador de números aleatorios y un umbral de decisión de uno. Podemos suponer que no se detectaron ejemplos positivos y que todos los ejemplos negativos son correctos. Esto corresponde a que la TPR y la FPR son cero. Cuando disminuyamos el umbral, más ejemplos positivos serán clasificados correctamente, pero algunos negativos serán clasificados erróneamente. En el otro extremo, con un umbral de cero

todo se etiqueta como positivo, lo que corresponde a TPR y FPR igual a uno. En particular, la curva ROC de un clasificador aleatorio estará muy cerca de la función de identidad.

Esperaremos un clasificador (con suerte) mejor que una suposición aleatoria, es decir, concentrar la probabilidad de salida en torno a 0 y 1. En este contexto, cuando el umbral empiece a disminuir, la proporción de positivos que se identifiquen frente a los negativos que se clasifiquen erróneamente debería ser mayor que uno. La curva ROC debería mostrar valores altos de TPR vs. valores bajos de FPR. En palabras más intuitivas, el clasificador identifica las clases positivas sin generar falsas alarmas. El área bajo la curva ROC es la suma de la proporción entre las dos cantidades para todos los umbrales.

Utilizamos el AUC como nuestra principal métrica de comparación porque: i) se ha utilizado tradicionalmente en trabajos anteriores, y permite la comparación con otros métodos, ii) proporciona más información que la exactitud, como se ha explicado, iii) tiene dos propiedades deseables para comparar el rendimiento de los clasificadores:

- AUC es invariante con respecto a la escala de los resultados del modelo. Mide cuán bien se clasifican las predicciones, en lugar de sus valores absolutos.
- AUC es invariante con respecto a umbral de clasificación. Mide la calidad de las predicciones del modelo independientemente del umbral de clasificación elegido.

#### RMSE

El Root Mean Square Error (RMSE) es una métrica de regresión que penaliza el cuadrado de la diferencia entre la probabilidad prevista y la probabilidad binaria verdadera.

Aunque esta métrica se utiliza en un contexto de regresión, también puede esclarecer algunos aspectos en tareas de clasificación, siempre que el resultado del clasificador sea probabilístico. Estimará la distancia entre las predicciones y los valores cero y uno, representando así la certeza del clasificador.

#### R2

La puntuación R2 o coeficiente de determinación estima la proximidad del rendimiento del clasificador a un modelo constante que siempre devuelve el mismo valor, independientemente de la entrada. Hay varias definiciones de este coeficiente, aplicamos la implementación de la biblioteca de Scikit-learn
urlhttps://scikit-learn.org, formalizada en Ecuación 5.1.

$$R^2 = 1 - \frac{\sum_i (y_i - \hat{y}_i)^2}{\sum_i (y_i - \bar{y})^2} \tag{5.1}$$

donde $\hat{y}_i$ es la etiqueta verdadera del ejemplo $i$, $y_i$ es la etiqueta obtenida por el modelo, y $\bar{y}$ es la media de todas las etiquetas verdaderas $\hat{y}$.

La métrica R2 captura la relación entre el error del clasificador y la varianza de los datos. Puede interpretarse como una medida de qué parte de la varianza del problema, estimada a través de la varianza del conjunto de datos, está siendo capturada por el modelo. Un clasificador constante, uno que siempre devuelve la media $\bar{y}$, tendrá R2 cero porque no explica la varianza de ninguno de los ejemplos.

Utilizando esta fórmula, el valor de R2 puede ser negativo, porque las predicciones pueden tener infinitamente más varianza que los datos. Por supuesto, esto indicará un modelo no deseable.

### 5.2.3 Análisis de rendimiento

En la mayoría de los problemas de aprendizaje automático, la evaluación se lleva a cabo explorando una gran parte de los posibles valores para los hiperparámetros, seleccionando el modelo que tenga mejor rendimiento sobre los datos de entrenamiento, e informando de su capacidad de generalización utilizando el rendimiento sobre los datos de las pruebas. Este enfoque no es suficiente cuando se trabaja con modelos de aprendizaje profundo, como argumenta Reimers and Gurevych [2017], por varias razones.

Las redes neuronales son procesos estocásticos afectados por más de una variable aleatoria. Por un lado, tenemos más de una docena de hiperparámetros, con un número exponencial de combinaciones. Por otro lado, la red se inicializa con pesos aleatorios, lo que se ha demostrado que impacta en el rendimiento [Erhan et al., 2010]. Las señales de la red siguen después un camino que puede tener más de una capa de dropout aleatorio. Por último, el conjunto de datos de entrenamiento debe ser mezclado con un nuevo orden aleatorio después de cada época, para ayudar al optimizador a evitar los mínimos locales y el sobreajuste.

Como consecuencia, encontrar el mejor modelo depende más de variaciones aleatorias que de la propia arquitectura. Un resultado puede ser excepcionalmente bueno para una tarea particular debido a la inicialización, o la activación de las capas de dropout, que no son reproducibles o adaptables a otras tareas. Además, el rendimiento se calcula sobre el conjunto de datos de la prueba, que suele ser pequeño, y puede disminuir cuando se realizan pruebas con datos diferentes.

Por lo tanto, es importante analizar no sólo el resultado máximo, sino también la distribución de todos los resultados obtenidos durante la búsqueda de hiperparámetros. Esta información

cubre el mejor y el peor de los casos, y ayuda a determinar cuán robusta es una arquitectura propuesta, independientemente de las variaciones aleatorias o de los hiperparámetros no relevantes. Si la arquitectura produce un modelo que supera el estado del arte en 0,01 puntos, pero se han buscado cientos de modelos de menor rendimiento para encontrar este modelo superior, ¿es realmente una arquitectura mejor? Bajo estas circunstancias, es probable que el éxito no se reproduzca en un escenario ligeramente diferente. Para cada escenario de aplicación, otros investigadores deben tener la información necesaria para decidir qué tan probable es que encuentren un modelo similar.

# Capítulo 6

# Rastreo de conocimientos

Hemos mencionado varios aspectos a tener en cuenta al entrenar y evaluar los modelos neuronales, y hemos presentado el esquema de los experimentos a realizar. En el presente capítulo se describen aspectos particulares de los experimentos para la tarea de rastreo de conocimientos y se resumen los resultados obtenidos.

En la primera sección, describimos el conjunto de datos de *ASSISTments 2009-2010* empleado para el Rastreo del Conocimiento. Este conjunto de datos ha sido ampliamente utilizado por investigaciones anteriores, que ya lo han descrito y analizado. Sólo incluimos información relevante para los experimentos realizados.

En las siguientes secciones, describimos los métodos base (*baselines*) contra los cuales comparamos el rendimiento y el proceso de optimización de hiperparámetros, terminando el capítulo con los resultados de los diferentes escenarios evaluados.

## 6.1 ASSISTments 2009-2010

Para la tarea de pretexto de Rastreo de Conocimiento, exploramos el desempeño de nuestro enfoque en el conjunto de datos de ASSISTments 2009-2010. El sistema ASSISTment es un software inteligente de tutoría en línea (ITS por sus siglas en inglés) creado en 2004. Los estudiantes interactúan con el sistema en un entorno de clase, dirigido por un profesor. Se les presenta un problema, si responden correctamente se les da uno nuevo. Si se equivocan, se les proporcionan preguntas que dividen el problema en pasos.

Este conjunto de datos es un ejemplo típico de los datos generados por un sistema ITS y contiene una cantidad bastante grande de interacciones de los estudiantes. Además, es un conjunto de datos de referencia para el campo del EDM, ya que ha sido utilizado para la experimentación

por varios trabajos. Lo usamos como referencia para evaluar el rendimiento de la arquitectura con co-embeddings en comparación con los embeddings disjuntos, así como con investigaciones anteriores.

La tarea del pretexto es predecir la columna *correct*, que indica si el estudiante resolvió el problema en la primera presentación. Para ello, se registran múltiples aspectos de las interacciones, como el id de usuario, el id de problema, el recuento de intentos, el tipo de respuesta y el id de habilidad involucrada en el ejercicio.

Como se ha señalado en Xiong et al. [2016], las interacciones entre un estudiante y un ejercicio etiquetado con múltiples habilidades se almacenan duplicando la fila de la interacción y etiquetando cada una de ellas con una habilidad diferente. Esto conduce a la duplicación de datos y puede afectar el rendimiento de cualquier clasificador si hay información filtrada entre los ejemplos. Siguiendo Xiong et al. [2016], representamos ejercicios con múltiples habilidades con una nueva habilidad que representa la fusión de las originales.

Con estas modificaciones, obtenemos 346.860 registros únicos, correspondientes a 4.217 alumnos. El número de interacciones por estudiante sigue una distribución exponencial, con la mayoría de los estudiantes con menos de 50 interacciones, y un máximo de 889.

En el conjunto de datos de ASSISTments 2009-2010 hay 113 habilidades únicas, mientras que hay 26.688 problemas únicos. Como en otros fenómenos humanos, la frecuencia de aparición de un problema es también exponencial, con sólo unos pocos ejercicios que aparecen más de 10 veces. Como ya se ha mencionado, la principal dificultad de utilizar el identificador único del ejercicio como entrada es la explosión del número de parámetros. Con tantos ejercicios únicos, su representación mediante vectores one-hot encoding es costosa, tanto en requisitos computacionales como en el tamaño del modelo. Explicaremos en el siguiente capítulo los desafíos de entrenar arquitecturas con demasiados parámetros. Los embeddings incorporados mitigan este problema, creando un vector denso de tamaño fijo para cada problema. Sin embargo, debe haber suficientes ejemplos para que estos vectores se actualicen, de lo contrario permanecen con su valor aleatorio inicial.

Muchos problemas de gran dimensión siguen la ley empírica Zipf, que en términos generales establece que la frecuencia de un elemento es inversamente proporcional a su clasificación en una tabla de frecuencias. Hay un conjunto limitado de elementos que se producen la mayoría de las veces, y una gran cola de elementos que se producen con una frecuencia muy pequeña. Como resultado, un buen método de reducción de la dimensionalidad es eliminar los elementos que ocurren menos de $n$ número de veces, donde el valor de $n$ es lo suficientemente pequeño como para mantener la pérdida de información al mínimo, pero lo suficientemente grande como para que una gran parte de los elementos sean descartados. Sin embargo, esos enfoques no funcionaron para este conjunto de datos:

- Eliminar los ejercicios poco comunes: Si quitamos los ejercicios que se repiten menos de 3 veces, el número resultante de ejercicios es de 21.111. Si tomamos un umbral de 5 veces, el número de ejercicios sigue siendo 17.305 de los 26.688 originales.

- Muestreo de estudiantes: Si tomamos 500 del total de 4217 estudiantes, entonces descartamos 291.375 de las 346.860 filas del conjunto de datos (84%). Sin embargo, todavía tenemos 12.796 problemas únicos en nuestro conjunto de datos, incluso después de filtrar los que parecen menos de 5 veces. Una reducción tal no produce un decrecimiento en el tamaño del modelo que justifique la pérdida de información.

También exploramos otros métodos de filtrado y muestreo basados en la similitud de los ejercicios, agrupando ejercicios similares, sin que se produzcan nuevas mejoras en la reducción de la dimensionalidad. Los problemas no podían ser divididos en conjuntos disjuntos sin estudiantes en común. Como resultado, el único preproceso que aplicamos al conjunto de datos fue filtrar los ejercicios con menos de 5 ocurrencias.

Para todos los experimentos, las secuencias de acciones de los estudiantes en el conjunto de datos se dividieron en tres partes, entrenamiento (70%), evaluación (20%) y validación (10%). Los rendimientos reportados se obtienen aplicando el modelo de entrenamiento sobre el conjunto de datos de evaluación, después de que los mejores hiperparámetros han sido elegidos usando el conjunto de datos de validación.

Para resolver el problema de procesar secuencias de longitud variable, rellenamos las secuencias demasiado cortas con ceros hasta que coincida con la longitud de la secuencia más larga del batch (lote). Utilizando una implementación dinámica de RNN en TensorFlow, la librería ignora los elementos añadidos de cada secuencia en el cálculo de la función de pérdida. Las secuencias se agrupan en batchs según su longitud para reducir la cantidad de relleno.

## 6.2 Clasificadores base

Comparamos nuestro clasificador con los siguientes modelos base y del estado del arte:

- Resultado más común: asignamos a cada interacción el resultado más común (modo) para el ejercicio o el estudiante.

- Resultado promedio: asignamos a cada interacción el resultado promedio del ejercicio o del estudiante.

- La implementación del DKT [Xiong et al., 2016] y nuestro propio modelo **LSTM**.

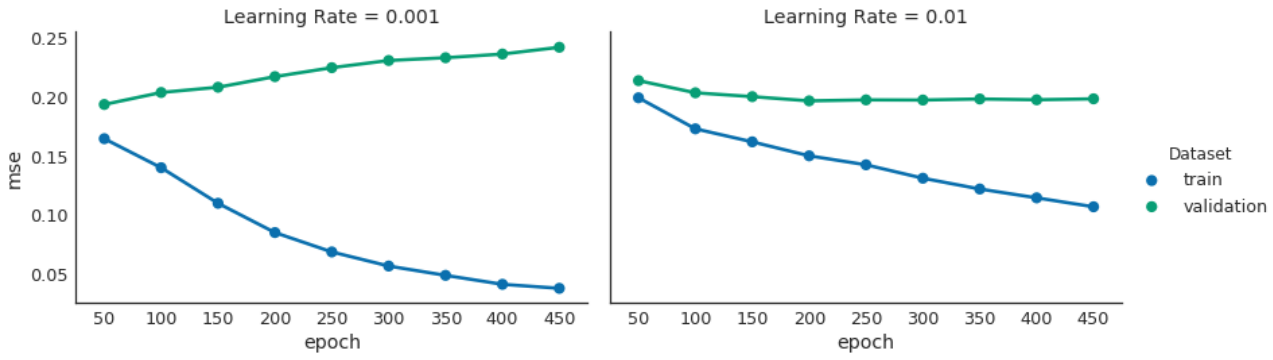
Además, hemos realizado experimentos con arquitecturas recurrentes utilizando la información de los componentes de conocimiento para representar cada ejercicio. Con este enfoque, estamos utilizando información adicional introducida por un experto anotador, y puede considerarse como un límite superior para el desempeño de otros métodos que utilizan representaciones totalmente no etiquetadas.

Para nuestro enfoque propuesto, todos los experimentos se realizaron utilizando una red RNN con celdas LSTM construidas en TensorFlow v1.2. La optimización se realizó con el optimizador Adam, y todas las capas tenían parámetros de regularización.

## 6.3 Hiperparámetros

Todas las arquitecturas fueron optimizadas con el optimizador Adam, con una tasa de aprendizaje estándar de 0,001. Un valor más bajo, como 0,01, aumenta la tendencia del clasificador hacia el sobreajuste. En la figura 6.1 podemos ver cómo la curva de aprendizaje con respecto al error cuadrado medio (*mse*) en el conjunto de datos de entrenamiento cae drásticamente para la tasa de aprendizaje de 0,01, mientras que la curva de aprendizaje de validación aumenta con las épocas de entrenamiento. Los clasificadores mostrados son arquitecturas E-LSTM con hiperparámetros idénticos.

Figure 6.1: Comparación de las curvas de aprendizaje para diferentes tasas de aprendizaje, para dos modelos E-LSTM con parámetros idénticos.




Para explorar los hiperparámetros, usamos un algoritmo de búsqueda aleatoria con los posibles valores mostrados en la Tabla 6.1. Todos los modelos fueron entrenados durante 300 épocas.

El hiperparámetro de pasos máximos controla el número de veces que se desenrolla la red para propagar los gradientes, como se explica en la sección 5.1.2. Esta es una diferencia clave entre esta implementación y los trabajos de Piech et al. [2015] y Xiong et al. [2016].

Table 6.1: Valores de hiperparámetros explorados.

| Hiperparámetro | Valores posibles |
|---|---|
| Pasos máximos | 30, 50, 100, y 300 |
| Número de celdas LSTM | 30, 50, 100, y 200 |
| Proporción de dropout | 0.1, 0.2, 0.3, 0.4, y 0.5 |
| Tamaño de batch | 30, 50, y 100 |

## 6.4 Resultados

En esta sección presentamos de las cifras de rendimiento obtenidas durante la experimentación. Todos los valores se calculan usando la concatenación de todos los resultados, en contraste con el promedio del rendimiento en cada secuencia individual de los estudiantes.

| Método | AUC ROC | R2 | RMSE | Exactitud |
|---|---|---|---|---|
| Resultado más común por estudiante | 0.602 | -0.346 | 0.555 | 0.691 |
| Resultado promedio por estudiante | 0.504 | -1.792 | 0.799 | 0.360 |
| Resultado más común por problema | **0.658** | **-0.231** | **0.531** | **0.717** |
| Resultado promedio por problema | 0.566 | -1.442 | 0.747 | 0.440 |
| Resultado más común por habilidad | 0.529 | -0.483 | 0.577 | 0.666 |

Table 6.2: Resultados para modelos base en el conjunto de datos no filtrado.

Los resultados de los modelos estadísticos base utilizados resultados promedios y más comunes se muestran en la tabla 6.2. Los resultados de rendimiento de los modelos de aprendizaje automático se resumen en la Tabla 6.3. Para facilitar la comparación con el estado del arte, mostramos los resultados reportados por Xiong et al. [2016] para el mismo conjunto de datos, marcados con un asterisco. Los autores encontraron que el conjunto de datos original contenía registros duplicados, e informan también de nuevos resultados usando la implementación presentada por Piech et al. [2015] sobre el conjunto de datos corregido. Estos valores de rendimiento no son directamente comparables porque no utilizan el mismo conjunto de datos para las pruebas. En particular, se utiliza todo el conjunto de problemas, en lugar de filtrar problemas menos comunes como lo hacemos nosotros.

La redacción en Xiong et al. [2016] parece indicar que el método está usando DKT con el ID de la habilidad. De hecho, cuando utilizamos nuestro LSTM con ID de habilidades (cuarta fila de la tabla), obtenemos un rendimiento comparable. Sin embargo, la intención de Piech et al. [2015] y nuestra propuesta es evitar confiar en esta información añadida manualmente, usando el ID del problema como entrada.

Cabe mencionar que un modelo LSTM entrenado con Id del problema alcanzó 0,736 puntos de la métrica ROC de la AUC. Fue descartado porque mostraba evidencia de sobrejauste y tenía una puntuación R2 cercana a cero.

| Model | Identifier | AUC ROC | R2 | RMSE | Accuracy |
|---|---|---|---|---|---|
| Resultado más común problema | Id problema | 0.658 | -0.231 | 0.531 | 0.717 |
| Xion BKT* | Id habilidad | 0.63 | 0.07 | - | - |
| Xion DKT* | Id habilidad | 0.75 | 0.18 | - | - |
| Piech DKT* | Id habilidad | 0.73 | 0.14 | - | - |
| LSTM (nuestro DKT) | Id habilidad | 0.746 | 0.176 | 0.432 | 0.720 |
| LSTM (nuestro DKT) | Id problema | 0.722 | 0.127 | 0.444 | 0.696 |
| E-LSTM | Id problema | 0.752 | 0.146 | 0.437 | 0.718 |
| CoE-LSTM | Id problema | 0.754 | 0.116 | 0.447 | 0.717 |

Table 6.3: Rendimiento de modelos de aprendizaje automático, modelo del estado del arte (filas superiores) y modelos propuestos (filas inferiores).

### 6.4.1 Analizando el área bajo la curva

La métrica AUC ROC es la más utilizada para evaluar el rendimiento en las tareas KT. Seleccionamos los modelos de mejor rendimiento de acuerdo con este valor, y a menudo lo usamos como sinónimo de rendimiento.

La primera interpretación que se puede hacer es que los modelos bayesianos funcionan de manera similar a la utilización del resultado más común por problema. Esto sugiere que el modelo bayesiano no es lo suficientemente complejo como para representar aspectos individuales de los estudiantes o elementos del curso.

El rendimiento del clasificador LSTM con identificaciones de habilidades se presenta como un límite superior de rendimiento, un punto de comparación del modelo en el caso óptimo: cuando la información etiquetada está presente. Podemos ver que usando sólo información no supervisada (la identificación del problema), el rendimiento de los métodos incorporados es comparable al rendimiento usando información etiquetada manualmente (identificación de las habilidades). Estos resultados indican que los modelos embebidos tienen un impacto positivo con respecto a la red básica de LSTM. La representación desarticulada (E-LSTM) obtiene resultados ligeramente mejores que la representación conjunta, pero las diferencias son tan pequeñas que es necesario seguir explorando para descubrir sus causas.

Siguiendo el argumento presentado en la sección 5.2.3, llevaremos a cabo un análisis más profundo de los resultados obtenidos. En la figura 6.2 vemos la distribución del rendimiento de todos los clasificadores explorados. Los gráficos boxenplot son similares a los gráficos de caja, pero muestran más información sobre diferentes cuantiles. Estos modelos no usan embeddings preentrenados, esos resultados serán analizados en la Sección 6.4.3.

Los modelos E-LSTM obtienen resultados superiores similares al modelo Co-ELSTM, pero las arquitecturas que usan la función *tanh* $\delta$ muestran un rango menor de resultados observados. Esto sugiere que la representación conjunta de los estudiantes y las embeddings de los cursos

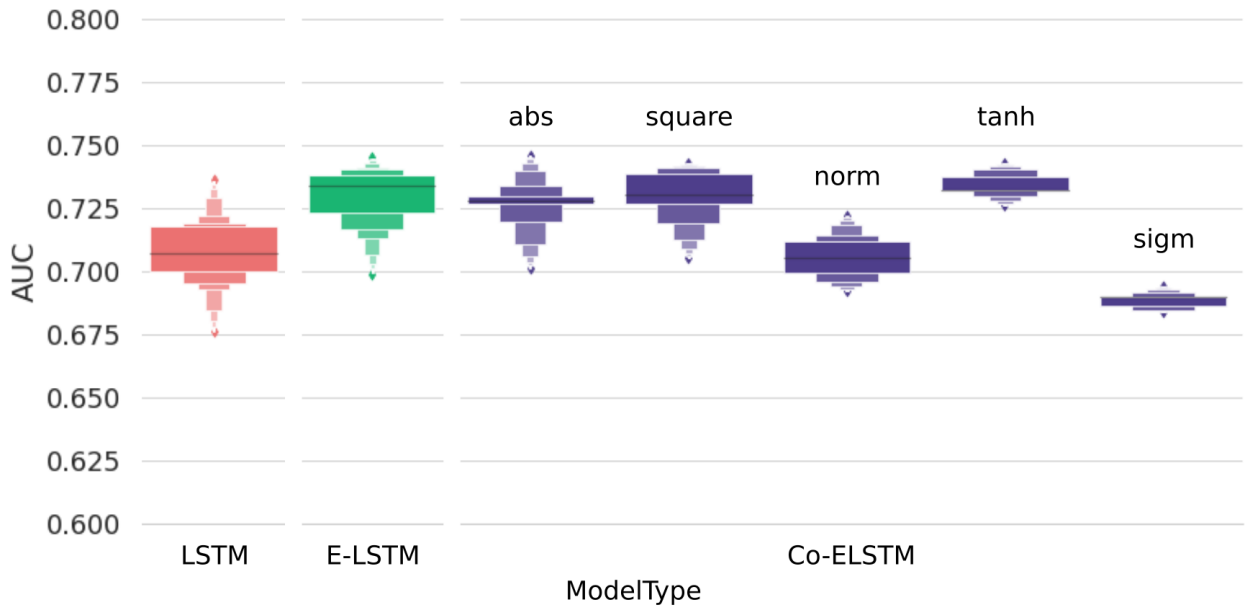



Figure 6.2: Distribución del rendimiento obtenido por las arquitecturas exploradas durante la búsqueda de hiperparámetros, usando diagramas boxenplot. Los modelos Co-ELSTM están separados por la función de distancia aplicada. Todos los modelos utilizan el ID del problema como identificador del elemento del curso. Los modelos E-LSTM y Co-ELSTM usan embeddings generadas aleatoriamente.

no mejora el rendimiento en esta tarea, medida por la métrica ROC de la AUC, pero produce clasificadores más estables, con una mayor probabilidad de no producir un mal clasificador cuando se entrena.

Podemos observar, en segundo lugar, que las diferentes funciones $\delta$ tienen un gran impacto en el rendimiento de los clasificadores Co-ELSTM. Basándonos en esta información, descartamos el uso de las funciones *norm* y *sigmoid*.

Nuestra hipótesis inicial sugería que las funciones *sq* y *abs* serían las que mejor funcionen, ya que mantendrían o aumentarían las señales cuando la diferencia entre embeddings sea grande. La función sigmoide sólo puede tomar valores entre 0 y 1, polarizando las diferencias entre estos dos valores (interpretados como lejos y cerca). Sin embargo, la función *tanh*, que tiene las mismas propiedades que la sigmoide pero en el intervalo $[-1, 1]$, fue la distancia de mejor rendimiento. ¿Qué está causando una diferencia de rendimiento tan grande entre funciones similares? Esto puede tener una respuesta no trivial: si los estudiantes y los elementos del curso empiezan cerca, como esperaríamos con un sistema de tutoría que recomienda ejercicios, entonces las señales de la distancia siempre serán apagadas por la función *sigmoid*. Si ese fuera el caso, el clasificador no podría aprender nada de las celdas con valores cero, porque tienen gradientes nulos. Esta explicación también se apoya en el hecho de que los modelos *sigmoid* funcionan mal también en el conjunto de datos de entrenamiento, lo que sugiere un problema

de subajuste.

En cuanto a la función *norm delta*, si el estudiante está demasiado lejos de la lección, la probabilidad de que se produzca por una distribución normal será pequeña. Esto apagaría efectivamente las señales provenientes del embedding del curso, es decir, dejaría sólo la información del estado del estudiante para su clasificación. El hecho de que esta función haya obtenido resultados de bajo rendimiento apoya la intuición de que el valor de la distancia es necesario para el clasificador.

Las funciones *abs* y *square* no difieren en el rendimiento de los modelos que producen, sino en su distribución. No tenemos más hipótesis para este fenómeno, pero recomendamos el uso de la función cuadrática, ya que es suave y convexa, más fácil de optimizar.

### 6.4.2 Otras métricas

Junto con el área bajo la curva, discutiremos el efecto de las diferentes arquitecturas en las otras métricas seleccionadas para la evaluación.

Los valores de la puntuación de R2 indican cuánto de la varianza del problema se explica por el modelo. En la tabla 6.3, podemos ver que los modelos que utilizan el ID de la habilidad como identificador tienen un rendimiento moderadamente mejor que los clasificadores del ID del problema, y el impacto es mayor que en la métrica de AUC. Esto indica que el ID del problema no proporciona suficiente información para que el modelo tenga en cuenta toda la variabilidad de los datos. Pueden ser necesarios más ejemplos y/o más capas ocultas para superar este problema.

Dado que hemos observado un impacto severo de sobreajuste para esta tarea, buscamos una métrica que pueda capturar ese fenómeno. Para ello calculamos una "puntuación de sobreajuste" definida como la diferencia entre el rendimiento en los conjuntos de validación y entrenamiento, en la última época de entrenamiento. Encontramos que la métrica de R2 está correlacionada negativamente con la puntuación de sobreajuste, como podemos ver en la figura 6.3. El gráfico nos muestra el valor de las dos métricas para una serie de experimentos con diferentes hiperparámetros de la arquitectura CoE-LSTM. El color indica dos cúmulos, de modelos buenos y modelos que no funcionan lo suficientemente bien.

Por último, las métricas de RMSE y Exactitud no muestran patrones diferentes a los mencionados anteriormente. Las diferencias entre las cifras de rendimiento, con la excepción del modelo LSTM, no son lo suficientemente grandes como para fundamentar más conocimientos.

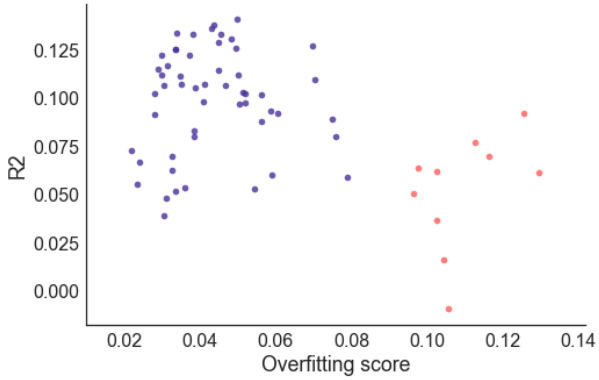



Figure 6.3: Correlación entre la puntuación de sobreajuste y la métrica R2.

### 6.4.3 Embeddings preentrenados

En esta tarea, no había una diferencia considerable entre el uso de embeddings de elementos de curso preentrenados y aleatorios, con y sin finetunning. Los resultados se presentan en la tabla 6.4.

| Tipo de modelo | Embedding EC | AUC | R2 | RMSE | Exactitud |
|---|---|---|---|---|---|
| LSTM (our DKT) | - | 0.722 | 0.127 | 0.444 | 0.696 |
| E-LSTM | Aleatorio | 0.745 | 0.122 | 0.447 | 0.710 |
| E-LSTM | Pre-entrenado | 0.751 | 0.142 | 0.443 | 0.710 |
| E-LSTM | Pre-entrenado+Finetune | 0.752 | **0.146** | 0.437 | 0.718 |
| CoE-LSTM | Aleatorio | 0.746 | 0.138 | 0.443 | 0.712 |
| CoE-LSTM | Pre-entrenado | 0.745 | 0.121 | 0.444 | 0.714 |
| CoE-LSTM | Pre-entrenado+Finetune | **0.754** | 0.116 | 0.447 | 0.717 |

Table 6.4: Comparación de desempeño entre modelos con embeddings de elementos del curso (EC) aleatorios, preentrenados, y con finetune

A primera vista, el impacto en el rendimiento no es alto, con una pequeña variación de menos de 0,01 puntos de AUC ROC. El puntaje de R2 aumenta cuando se usan embeddings preentrenadas en modelos E-LSTM y disminuye cuando se usan en modelos Co-ELSTM. Si analizamos la distribución de los clasificadores, como se muestra en la Figura 6.4, observamos que el uso de embeddings preentrenadas tiene un efecto diferente dependiendo de la función de distancia. Sin embargo, hay una tendencia general: los modelos que no utilizan el ajuste fino no logran rendimientos más altos.

Analizando las arquitecturas E-LSTM, las distribuciones de los embeddings con y sin finetune no parecen ser diferentes. Sin embargo, hay un cambio importante en los valores de rendimiento en los cuantiles superiores para los modelos Co-ELSTM. Nuestra hipótesis es que el modelo E-LSTM no tiene suficiente información para afinar los embeddings, y debe basarse en la información capturada por el algoritmo word2vec durante el preentrenamiento. Esta es ya una representación adecuada de los elementos del curso. Cuando las señales débiles de gradiente

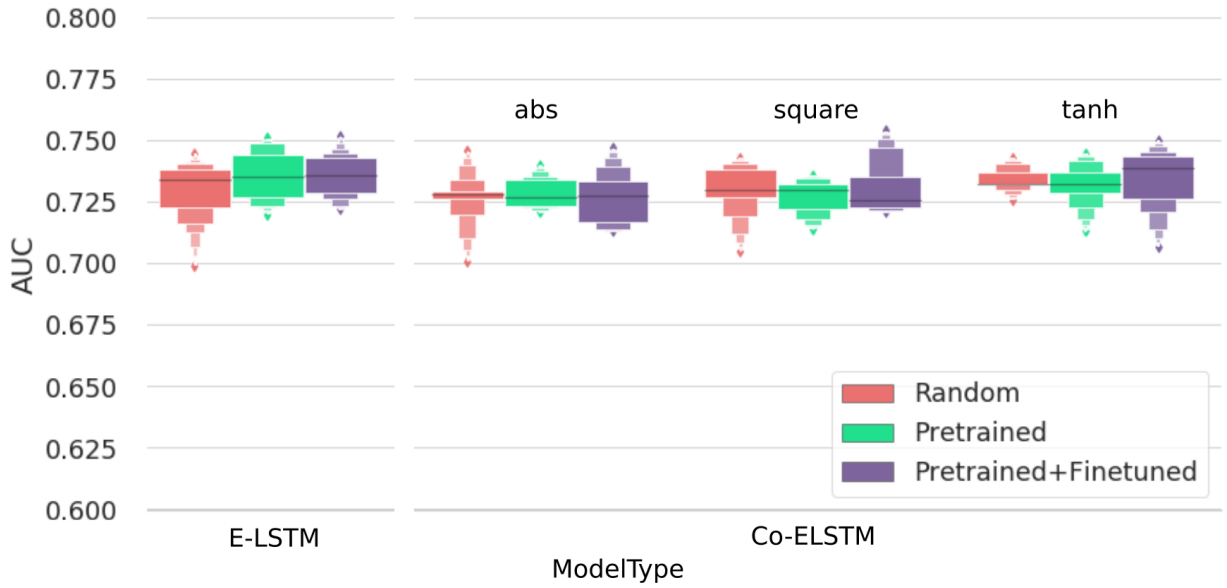



Figure 6.4: Distribución de la actuación de la AUC ROC en los modelos E-LSTM y Co-ELSTM usando embeddings de elementos de curso aleatorios, preentrenados y con finetune (CE)

procedentes de la optimización de la función de pérdida pasan a través de la capa recurrente, no contienen suficiente información para desplazar los embeddings hacia valores más adecuados.

Por otro lado, los modelos Co-ELSTM obligan a que los embeddings de los elementos del curso estén en el mismo espacio que los embeddings de los estudiantes. Es necesario permitir un ajuste fino en este tipo de arquitecturas, ya que el método utilizado para generar los embeddings preentrenadas fue diseñado originalmente para lenguaje natural. Si no permitimos el ajuste, los embeddings del estudiante deben adaptarse completamente a los embeddings de los elementos del curso, sin tener en cuenta la tarea de clasificación. La representación original de los elementos del curso puede no ser óptima para el rastreo de conocimientos, como lo demuestra el menor rendimiento de esos modelos.

## 6.5 Impacto de los hiperparámetros

En aras de la reproducibilidad, en la tabla 6.5, describimos los hiperparámetros que produjeron las arquitecturas de mejor rendimiento.

Aunque esta exploración incluye más de 250 arquitecturas evaluadas, se basa en la intuición y no consideramos que abarque todo el espacio hiperparamétrico. Sin embargo, la optimización de un modelo no es el objetivo principal de este trabajo, en el que buscamos una prueba de

| Modelo | Tipo EEC | Distancia | MaxPasos | LSTM | Dropout | Batch | Tamaño EEC |
|---|---|---|---|---|---|---|---|
| LSTM | - | - | 50 | 100 | 0.5 | 30 | - |
| E-LSTM | Aleatorio | - | 200 | 200 | 0.1 | 50 | 50 |
| E-LSTM | Pre | - | 200 | 100 | 0.1 | 50 | 100 |
| E-LSTM | Pre+Fine | - | 300 | 100 | 0.3 | 50 | 200 |
| CoE-LSTM | Aleatorio | abs | 200 | 200 | 0.5 | 50 | 200 |
| CoE-LSTM | Pre | tanh | 100 | 200 | 0.2 | 30 | 200 |
| CoE-LSTM | Pre+Fine | square | 200 | 200 | 0.1 | 100 | 200 |

Table 6.5: Hiperparámetros de las arquitecturas con mejor rendimiento. Utilizamos EEC para denostar embedding de elementos del curso, Pre para embeddings preentrenados, Fine para embeddings con ajuste fino, MaxPasos para el número de pasos de retropropagación, LSTM para el tamaño de la capa recurrente, Dropout para la proporción de la capa de dropout y Batch para el tamaño de cada lote de entrenamiento.

concepto sobre el uso de embeddings conjuntos. En ese sentido, en esta sección sólo extraeremos algunas ideas de los hiperparámetros para comprender las propiedades del modelo.

En términos generales, las variables con mayor impacto en el rendimiento de la arquitectura fueron el tipo de modelo, el preentrenamiento y la distancia utilizada. Otros hiperparámetros no determinaron fuertemente el rendimiento, excepto en algunos valores extremos. Por ejemplo, en el caso del número de pasos para retropropagar, el modelo LSTM no podía ser entrenado con más de 100 pasos y un tamaño de lote mayor de 30 debido a las limitaciones de memoria. Utilizamos la métrica R2 para evaluar la bondad del ajuste para los modelos, ya que representa lo bien que el clasificador está captando la varianza del conjunto de datos. En la figura 6.5, podemos observar que los modelos más exitosos requieren 100 o 200 pasos para retropropagar, lo que sugiere que el problema del rastreo del conocimiento está mejor modelado teniendo en cuenta las dependencias a largo plazo.

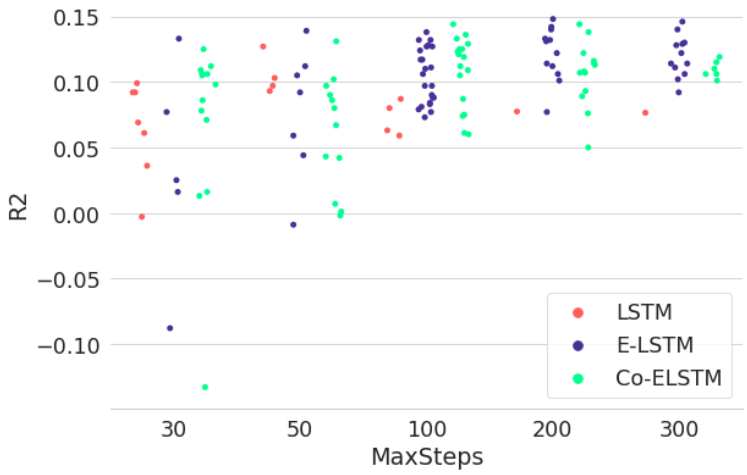



Figure 6.5: Distribución de los valores de rendimiento de R2 con respecto al número de pasos para retropropagar (MaxPasos), dividido por el tipo de modelo

Se pueden encontrar patrones similares en otros hiperparámetros. El tamaño del lote es mejor

con valores pequeños más cercanos a 50. El número preferido de unidades LSTM es 100 para los modelos Co-ELSTM, y 30, 50 ó 100 para los modelos E-LSTM. Este resultado sugiere que la arquitectura Co-ELSTM es más compleja, y requiere más neuronas para aprender adecuadamente la relación entre el estudiante y el elemento del curso. El tamaño de la capa LSTM restringe el tamaño de los embeddings del curso, que también funciona mejor con el tamaño 100. Esta puede ser una explicación diferente para el tamaño de las unidades LSTM requeridas con los modelos Co-ELSTM.

En cuanto al uso de dropout, los mejores valores son 0,5 para los modelos LSTM, entre 0,1 y 0,4 para E-LSTM, y 0,1 para Co-ELSTM. Los modelos LSTM usan una capa de one-hot encodings, que necesita una alta probabilidad de abandono para apagar la única célula activa. En el caso de los modelos Co-ELSTM, la caída en el rendimiento cuando se utiliza un valor de abandono superior a 0,1 puede indicar que toda la información de los embeddings de los elementos del curso es necesaria para calcular la distancia entre el estudiante y los embeddings de los elementos del curso.

## 6.6 Discusión

Hemos evaluado el impacto de una representación conjunta de los estudiantes y los elementos del curso en la tarea de Rastreo de Conocimientos. Las arquitecturas con embeddings funcionan mejor que las arquitecturas sin embeddings, alcanzando un rendimiento de vanguardia. El uso de co-embeddings no parece perjudicar el rendimiento, aunque tampoco se pueden encontrar mejoras notables.

El factor más importante de variación en el rendimiento es la función $\delta$, con *tanh* como la función de mejor rendimiento. Aunque alcanza un valor AUC comparable, la distribución de los diferentes modelos aleatorios entrenados con esta arquitectura tiene un rango menor de resultados. Otros hiperparámetros no parecen tener un impacto importante.

# Capítulo 7

# Predicción de la deserción escolar

Siguiendo la misma estructura del capítulo anterior, comenzamos introduciendo el conjunto de datos utilizados para evaluar el impacto de los embeddings, más específicamente los embeddings conjuntos en la tarea de predicción de la deserción. A continuación, describimos los métodos de base utilizados para la comparación y el proceso de optimización de los hiperparámetros. Presentamos diferentes escenarios experimentales que destacan los puntos fuertes y débiles de nuestra propuesta.

## 7.1 Conjunto de datos KDDCup 2015

El concurso KDDCup 2015 [1] propuso la tarea de predecir la deserción de los estudiantes en MOOCs. Los datos fueron proporcionados por XuetangX, una plataforma MOOC de aprendizaje china iniciada por la Universidad de Tsinghua y socia de EdX. Para esta tarea en particular, el evento de abandono se definió como la ausencia de actividad de los estudiantes en los diez días siguientes, aunque no está claro si los cursos se completaron en ese momento.

Aunque la información ya no está disponible en el sitio web del concurso, este conjunto de datos es uno de los pocos ejemplos disponibles públicamente de registro detallado en un entorno MOOC.

Los datos proporcionados incluyen información de 38 cursos diferentes. Todos los cursos tienen una duración de 29 días, comenzando en diferentes fechas desde octubre de 2013 hasta agosto de 2014. En este trabajo, sólo pudimos utilizar la parte de entrenamiento del conjunto de datos, que describimos a continuación. Los organizadores de la competición sólo publicaron las secuencias de interacciones de la parte de evaluación, y no sus etiquetas. Por lo tanto, no podemos usarlas para evaluar el clasificador.

[1] `https://biendata.com/competition/kddcup2015/`

Figure 7.1: Distribución de las inscripciones únicas por curso.

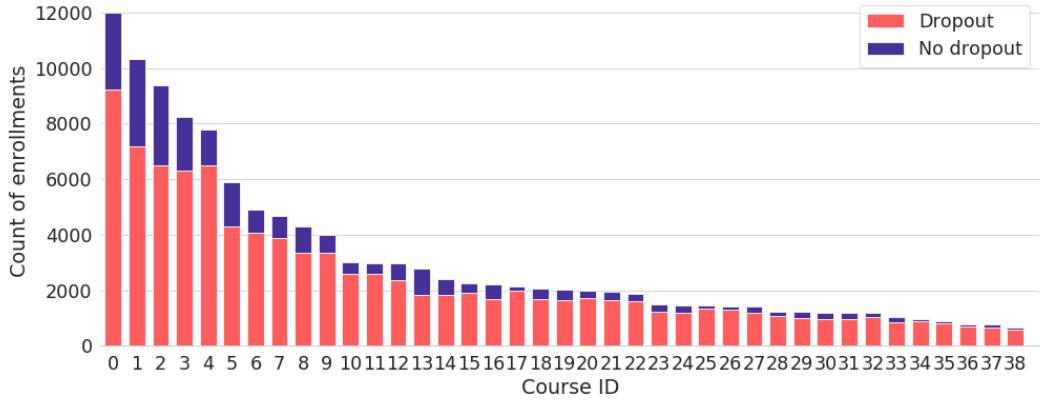
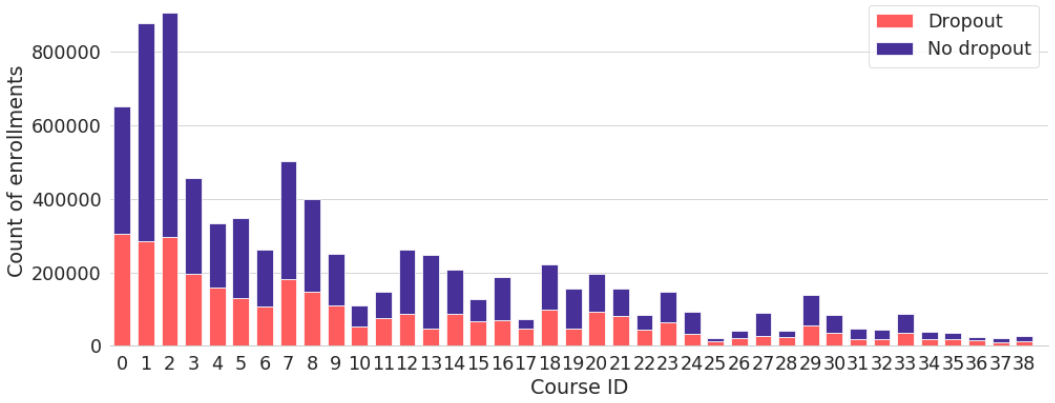



Figure 7.2: Distribución de las interacciones únicas por curso.

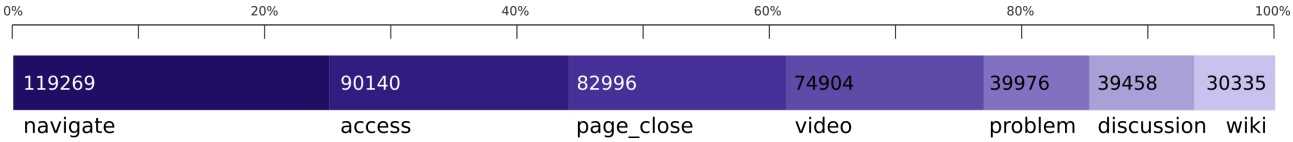



El conjunto de datos comprende 120.542 inscripciones etiquetadas, correspondientes a 79.186 usuarios únicos. El total de 8.157.278 interacciones incluye información como el acceso a contenidos de video, la resolución de un problema, el acceso a una wiki del curso, etc. Los eventos están etiquetados con la hora e identificados con el estudiante y el curso correspondiente (inscripción). Además de las actividades de los estudiantes, el concurso proporcionó información sobre la organización jerárquica de los elementos del curso.

Los cursos no tienen la misma cantidad de inscriptos, variando entre 12004 y 645 (curso 0 y 38 respectivamente), con la distribución que se muestra en la Figura 7.1. El número de interacciones únicas no se corresponde directamente con el número de inscripciones, con un máximo de 907.118 para el curso 1 y un mínimo de 21.216 para el curso 25. La distribución detallada se presenta en la figura 7.2.

En todo el conjunto de datos, el 80% de las inscripciones corresponden a una deserción, aunque esta distribución cambia entre los cursos. Como podemos observar en las figuras 7.1 y 7.2, la mayoría de las inscripciones corresponden a deserciones, pero la mayoría de las interacciones se

Figure 7.3: Distribución de las interacciones de acuerdo al tipo de evento.




producen por inscripciones sin deserción. Esto es de esperar, ya que la intuición apunta a que las secuencias de abandono son más cortas.

### 7.1.1 Detección de valores anómalos

Es común encontrar en los conjuntos de datos ejemplos que destacan y no siguen la distribución general, denominados comúnmente outliers. Este tipo de datos puede provenir de diversas fuentes: fallos en las herramientas de recopilación de datos, errores en la introducción de datos, informe incorrecto de un fenómeno de distinta naturaleza, entre otros. Además, también pueden ser verdaderos valores atípicos en la distribución, que corresponde a observaciones válidas, aunque poco frecuentes.

Algunos modelos de aprendizaje automático, como las regresiones lineales, son demasiado sensibles a las distribuciones desequilibradas o a la presencia de ejemplos anómalos. Por esta razón, es una práctica común curar los conjuntos de datos antes del entrenamiento, eliminando cualquier registro sospechoso.

Las redes neuronales se consideran robustas en este sentido, pero las redes neuronales recurrentes en particular pueden verse afectadas por secuencias demasiado largas. Dado que las secuencias se rellenan hasta la secuencia más larga del lote, si una secuencia es excesivamente larga afectará también a otros ejemplos, lo que provocará un aumento del tiempo y los recursos de entrenamiento. Además, i) la mayoría de las secuencias largas están etiquetadas como no abandonadas, ii) podrían corresponder a otros agentes, como bots o los programas automáticos, en lugar de estudiantes reales.

En este conjunto de datos encontramos varias secuencias demasiado largas, que decidimos descartar por las razones mencionadas anteriormente. El criterio exacto fue eliminar las inscripciones con un número de eventos superior al percentil 0,99, dejando 7.171.235 interacciones.

#### Distribución de eventos

Hay más de 26.000 objetos descriptos en el conjunto de datos, pero sólo 5.900 de ellos aparecen en interacciones. Podemos ver en la Figura 7.3 la distribución aproximada de las interacciones en el conjunto de datos según su tipo de evento.

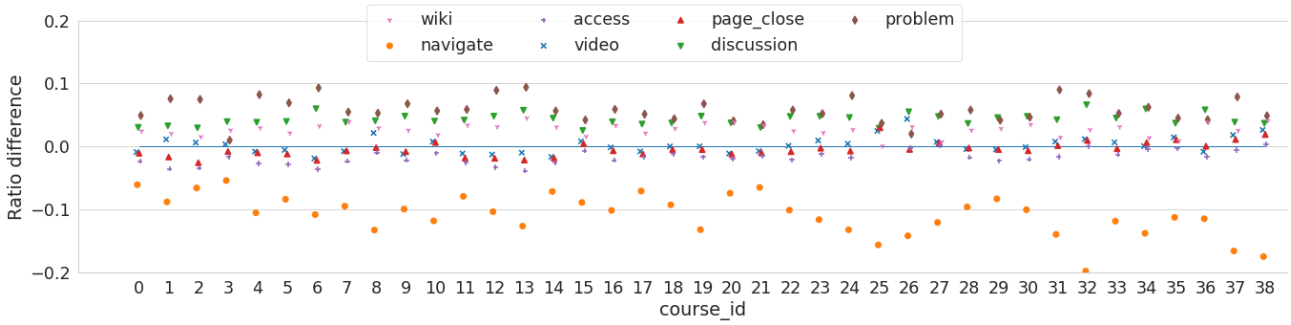



Figure 7.4: Diferencia en la proporción de tipos de acción entre los estudiantes que no abandonan y los que abandonan, separados por curso.

A partir de una exploración preliminar, observamos en varios cursos que la proporción de acciones *access* y *navigate* son más prominentes en los estudiantes que abandonan. Por otro lado, los eventos del tipo *problem* son más comunes en los estudiantes que no abandonan. Para verificar que este fenómeno se encuentra en todos los cursos, creamos una visualización más extensa.

En la figura 7.4 podemos observar la diferencia entre las proporciones de los tipos de eventos en los estudiantes sin deserción y con deserción. Si la diferencia es positiva, es decir, por encima de la línea, indica que los estudiantes que no abandonan acceden proporcionalmente a ese tipo de evento más a menudo que los estudiantes que sí abandonan. Por ejemplo, en el curso 1, los estudiantes que no abandonan tienen una proporción de interacciones de acceso 9% más baja que los estudiantes que abandonan. Podemos ver que, en casi todos los cursos, los eventos *problem* y *discussion* son más comunes en estudiantes que no abandonan, mientras que lo contrario es cierto para eventos *navigate* y *access*. La diferencia en el resto, *video*, *page close* y *wiki*, es casi nula en la mayoría de los cursos, lo que indica que pueden no estar relacionados con la etiqueta del estudiante. Esto apoya la intuición de que los estudiantes que abandonan los estudios están menos comprometidos en las actividades y estudian el contenido de una manera más superficial.

### Análisis por sesiones

Para comprender mejor los patrones en los datos, es necesario agregarlos en unidades más interpretables. Para esta tarea elegimos usar *sesiones*, definidas como una secuencia de acciones en las que el tiempo entre ellas no supera el límite de dos horas.

Al trazar diferentes distribuciones, pudimos notar algunos patrones interesantes. El número de sesiones por estudiante tiene una distribución diferente entre la clase que abandona y la que no lo hace, como se puede ver en la Figura 7.5. Este patrón es consistente entre los diferentes cursos. El número de acciones y la duración de la sesión es ligeramente mayor en los estudiantes

sin abandono, mientras que el promedio de minutos entre las acciones es muy similar.

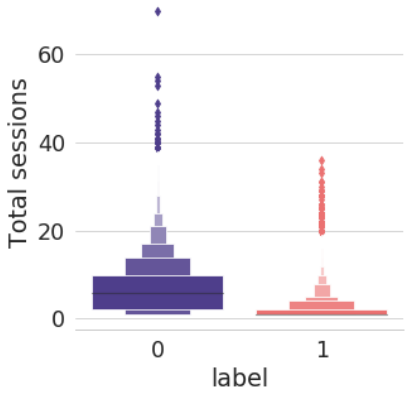



Figure 7.5: Distribución del número de sesiones por inscripción.

La información codificada en las sesiones es una característica importante que puede ser indicativa de la deserción, junto con la longitud de la secuencia. Sin embargo, no es posible saber el número final de sesiones hasta el final de la secuencia y, en consecuencia, esta característica es menos útil para las predicciones tempranas. Observamos que el lapso de tiempo entre las sesiones es mayor para los estudiantes con abandono, lo que podría utilizarse como un fuerte predictor.

Realizamos exploraciones adicionales para determinar si hay un declive en el número de sesiones en el tiempo hasta que se alcanza el abandono, si tienen menos interacciones en promedio o si están más espaciadas en el tiempo. No pudimos encontrar ningún patrón significativo que difiera entre los estudiantes que abandonan y los que no lo hacen en ninguno de estos aspectos.

Basándonos en este análisis exploratorio, podemos concluir que el fenómeno de la deserción sigue en general principios intuitivos, estando estrechamente relacionado con el número y el tipo de interacciones. Queda por ver si un simple modelo basado en características extraídas manualmente es suficiente para predecirlo, o si es necesario utilizar modelos más complejos y profundos.

La aplicación de los co-embeddings introducirá un tipo diferente de información en el modelo, tratando de optimizar las relaciones entre cada lección particular y el compromiso del estudiante. Queda por ver si la adición de esta información ayudará al clasificador a tener un mejor rendimiento. El otro aspecto a evaluar es si las arquitecturas recurrentes pueden capturar los patrones a largo plazo evidenciados por las secuencias.

**Particiones de datos para experimentos**

Para cada curso, seleccionamos el 80% de los estudiantes para el entrenamiento y el 20% restante se utiliza para las evaluación. De los estudiantes de entrenamiento, el 15% se utiliza como validación de las curvas de aprendizaje y del sobreajuste.

Un clasificador diferente es entrenado para cada curso. El número de cursos y el número de hiperparámetros dificultan la búsqueda de la combinación óptima, ya que entrenar y evaluar 39 modelos lleva varias horas. Para acelerar este proceso, durante la exploración utilizamos el rendimiento de sólo 7 cursos (1, 6, 11, 16, 21, 26 y 31), y una vez determinada la mejor arquitectura, la evaluamos en todos los cursos.

## 7.2 Clasificadores base

Para estimar la dificultad de la tarea, utilizaremos como base ficticia un clasificador que siempre asigna la etiqueta más frecuente. Para todos los cursos esto corresponde a la deserción.

En contraste con la tarea KT, hay pocas evaluaciones previas realizadas en los conjuntos de datos de KDDCup 2015. Además, hasta donde sabemos, no hay métodos establecidos para la predicción del abandono. Para tener un punto de comparación de base para nuestros clasificadores neuronales, proponemos utilizar otros modelos de aprendizaje automático.

El enfoque de aprendizaje automático más simple para una tarea de clasificación es, en general, la aplicación de un modelo lineal. Aunque sean poco profundos, los modelos lineales pueden arrojar mucha luz sobre un problema, y ayudar a comprender las relaciones entre sus muchas variables ocultas y observadas. Los modelos simples también son útiles para evaluar la complejidad de un problema, y para evaluar la necesidad de modelos más sofisticados, o más profundos.

Nos centraremos en dos tipos de clasificadores: regresión logística y árboles de decisión. No tienen la capacidad de procesar datos secuenciales, por lo que es necesario encontrar una representación adecuada de las secuencias para utilizarlas como entrada. Esta representación se basa en gran medida en características extraídas manualmente, y debe destacar sólo los aspectos importantes de los datos para que un clasificador lineal alcance un rendimiento adecuado. En este sentido, los clasificadores lineales pueden utilizarse para estimar la calidad de una representación con respecto a una tarea de clasificación.

En particular, buscamos responder a la pregunta: ¿las estadísticas y características calculadas sobre la secuencia son suficientes para predecir la deserción? A diferencia de las tareas KT, hay menos investigación sobre el tipo de características que son predictores de abandono. Los

conjuntos de datos son más variados, y la definición de abandono cambia. Es significativo entender, para este conjunto de datos en particular, las relaciones entre las características y las etiquetas de abandono.

Para construir el vector de características que representan a cada estudiante, usamos las conclusiones obtenidas del análisis presentado en la Sección 7.1. La representación final se compone de las siguientes características:

- El número total de interacciones
- El número promedio de interacciones por sesión, dividido por tipo
- El recuento de interacciones por sesión, dividido por tipo
- El número de sesiones diferentes
- La duración media de las sesiones

El resultado son 18 columnas de características de valores densos. También evaluamos el mismo clasificador, entrenado sin ninguna información sobre las sesiones. Esto nos permitirá evaluar hasta qué punto esos rasgos son esenciales para la tarea. En este escenario, utilizamos sólo el número total de interacciones y el recuento de interacciones por tipo.

El único paso de preproceso aplicado para todas las líneas de base fue la estandarización, para evitar la sobreponderación de una sola característica. Los valores que faltaban fueron reemplazados por ceros. Utilizamos el 80% de las filas para entrenamiento y el 20% para evaluación, ya que no había necesidad de validación. Los hiperparámetros utilizados fueron los predeterminados para la biblioteca Spark ML.

## 7.3 Hiperparámetros para modelos neuronales

Se exploraron todos los tipos de modelos con los mismos rangos de valores para cada hiperparámetro, cuando correspondía. El tamaño de embedding se movió entre 20 y 100, la proporción de abandono entre 0,3 y 0,5, el tamaño del lote entre 50 y 100, la tasa de aprendizaje de 0,01 a 0,001, y la cantidad máxima de pasos en la secuencia para propagarse de 20 a 300. El tamaño de los embeddings se eligió teniendo en cuenta dos razones. En primer lugar, el número de componentes únicos de cada curso es significativamente menor que el conjunto de datos de ASSISTments y, por lo tanto, un número menor de parámetros debería ser suficiente para representarlos. Además, la cantidad de ejemplos puede no ser suficiente para entrenar

un modelo con capas extensas. Sin embargo, esperamos que la deserción esté determinada por interacciones muy lejanas en el tiempo, y permitimos que el error se propague muchos pasos.

Además, realizamos experimentos con celdas GRU para la capa recurrente y el optimizador RMSEProp, que no mejoraron el rendimiento.

La selección de los hiperparámetros no se realizó de forma aleatoria debido a las limitaciones de tiempo, ya que una iteración exploratoria sobre los 7 cursos mencionados requiere varias horas, frente a los 40 minutos de media que se tarda en entrenar un modelo KT en todo el conjunto de datos. Cada paso de la exploración se seleccionó manualmente, cambiando uno o dos hiperparámetros en el momento. Aunque se entrenaron más de 1200 redes con resultados exitosos, esto abarca aproximadamente 60 arquitecturas diferentes, que tuvieron que ser entrenadas y evaluadas a lo largo de varios cursos. Para mitigar este hecho y obtener una buena comprensión del rendimiento del clasificador, cada modelo fue entrenado dos o tres veces, y los valores reportados son el promedio de todas las iteraciones.

#### Longitud de las secuencias

Encontramos una importante diferencia entre el uso de la BPTT (BackPropagation Through Time) tradicional y la BPTT truncada. Como se presenta en la Sección 5.1.2, la BPTT tradicional puede tener algunos problemas con secuencias muy largas, por ejemplo con más de 500 elementos. Por un lado, para realizar esta operación en una GPU debemos rellenar todas las secuencias a la longitud de la secuencia más larga, lo que requiere una cantidad significativa de memoria. Por otro lado, los gradientes se hacen más pequeños cuando los propagamos en el tiempo, cayendo en el problema de desaparición del gradiente. Para superar este problema utilizamos la estrategia propuesta en la sección 5.1.2, y la comparamos con el uso de sólo los últimos $h$ pasos de tiempo de la secuencia.

Esperábamos que el truncado de la BPTT funcionara mejor, ya que tiene acceso a la historia completa de las acciones. Sin embargo, vimos una caída significativa del rendimiento en comparación con la estrategia más simple de limitar el historial del estudiante.

Los resultados fueron consistentes a través de los ajustes y modelos de los experimentos, mostramos en la tabla 7.1 algunas métricas de rendimiento como ejemplos:

Una cosa importante a notar es la caída de la métrica R2, que indica que los modelos truncados están funcionando muy cerca de un clasificador por promedio.

En cuanto a la razón de este fenómeno, tenemos dos hipótesis. En primer lugar, la información contenida en las acciones al principio de las secuencias, demasiado lejos de la última acción vista, no son relevantes para la predicción de deserción. Además, los resultados indican que

Table 7.1: Diferencias en las actuaciones de Truncado vs. BPTT tradicional. El modelo no truncado utiliza las últimas 100 acciones de cada secuencia.

| Model0 | Truncado | Curso 1 | | Curso 6 | | Curso 21 | | Curso 26 | |
|---|---|---|---|---|---|---|---|---|---|
| | | AUC | R2 | AUC | R2 | AUC | R2 | AUC | R2 |
| Co-ELSTM | Sí | 0.708 | -0.028 | 0.700 | -0.060 | 0.627 | -0.067 | 0.594 | -0.127 |
| Co-ELSTM | No | 0.882 | 0.439 | 0.816 | 0.328 | 0.712 | 0.114 | 0.737 | 0.121 |

están añadiendo ruido y limitando la capacidad de aprendizaje del clasificador. En segundo lugar, como el algoritmo truncado realiza un pase de optimización hacia atrás después de ver cada trozo de eventos, puede estar aprendiendo algunos patrones incorrectos. Este sería el caso si las secuencias que terminan en abandono y sin abandono no difieren en sus acciones iniciales, y luego el punto en que el abandono se hace evidente está más cerca de la última acción.

#### Filtrado de instancias

Cuando las secuencias son demasiado cortas, puede ser difícil para un RNN generalizar a partir de tan pocos datos. Experimentamos entrenando los modelos con todas las secuencias, y también sólo con las secuencias de longitud superior a 5. Siguiendo los resultados generales, en los cursos grandes la diferencia de rendimiento es apenas perceptible. Sin embargo, en los cursos pequeños, el no filtrar los ejemplos cortos conduce a un aumento del rendimiento.

Table 7.2: Comparación de rendimiento entre los modelos con y sin filtrar las secuencias más cortas de 5 interacciones, dividido por el curso.

| Modelo | Filtrado | Curso 1 | | Curso 6 | | Curso 21 | | Curso 26 | |
|---|---|---|---|---|---|---|---|---|---|
| | | AUC | R2 | AUC | R2 | AUC | R2 | AUC | R2 |
| E-LSTM | Sí | 0.884 | 0.442 | 0.813 | 0.322 | 0.731 | 0.043 | 0.688 | 0.051 |
| E-LSTM | No | 0.887 | 0.461 | 0.850 | 0.366 | 0.783 | 0.184 | 0.699 | -0.069 |
| Co-ELSTM | Sí | 0.881 | 0.440 | 0.825 | 0.308 | 0.752 | 0.076 | 0.679 | 0.035 |
| Co-ELSTM | No | 0.884 | 0.441 | 0.853 | 0.382 | 0.814 | 0.277 | 0.749 | 0.062 |

## 7.4 Resultados

Los resultados de los clasificadores base se presentan en la tabla 7.3. La primera conclusión a la que podemos llegar es que los modelos de aprendizaje automático están aprendiendo efecti-

vamente el problema, y logran un rendimiento mucho mayor que una línea de base ficticia. La diferencia es más aguda en el caso de AUC ROC y R2, lo que proporciona cierta comprensión de la solidez de estas métricas con respecto a las demás.

En segundo lugar, los modelos de Regresión Logística superan a los Árboles de Decisión. El rendimiento del modelo, incluidas las características relacionadas con la sesión, no disminuye significativamente, lo que indica que esas características no codifican ninguna información nueva útil para el clasificador. Aunque los Árboles de Decisión no superan a la regresión logística, son clasificadores interpretables, ya que se basan en reglas basadas en características. Pudimos observar, a través de la inspección manual, que en general las características más importantes eran el número de sesiones, el número de interacciones totales y el recuento de interacciones por tipo. El orden entre las tres características variaba de un curso a otro. Presentamos una visualización de un Árbol de Decisión para el curso 6 en la Figura 7.6, donde se puede observar claramente esta distribución de características.

| Modelo | AUC ROC | R2 | RMSE | Accuracy |
|---|---|---|---|---|
| Valor más común | 0.5 | -0.249 | 0.446 | 0.800 |
| DT | 0.837 | 0.331 | 0.326 | 0.860 |
| LR | 0.840 | 0.344 | 0.323 | 0.865 |
| LR + sesión | 0.848 | 0.357 | 0.320 | 0.866 |

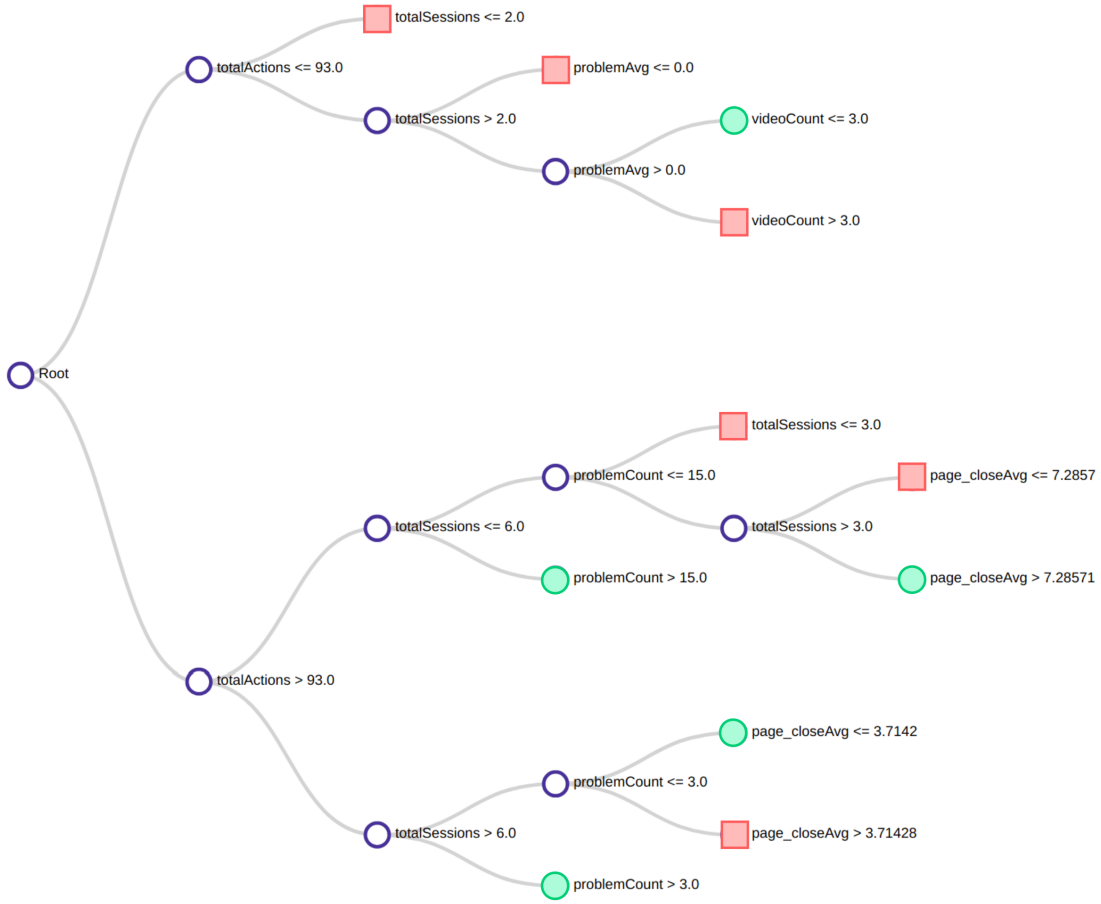
Table 7.3: Rendimiento de los modelos de base para la predicción de la deserción, calculado sobre todos los cursos. LR significa Regresión Logística, DT significa Árbol de Decisión, y sesión significa que se incluyeron características basadas en sesiones.

Los resultados de la evaluación general se muestran en la tabla 7.4, incluyendo el modelo LR con características de sesión, y todos los modelos neuronales, con y sin embeddings preentrenados. Las métricas que se muestran se calculan utilizando la predicción de todas las arquitecturas de mejor rendimiento para cada curso individualmente, ya que no se puede esperar que la misma arquitectura sea óptima para diferentes conjuntos de datos.

Podemos observar que hay un aumento en la AUC ROC y R2 cuando añadimos embeddings con respecto al modelo LSTM, pero sorprendentemente el modelo LR supera a los modelos LSTM y E-LSTM en la misma métrica. Esto parece indicar que el uso de una red neuronal recurrente no está captando características simples como el número de interacciones, y es necesario modificar el modelo para usar co-embeddings para obtener un rendimiento similar.

El RMSE aumenta ligeramente para los modelos LSTM y E-LSTM, mientras que la exactitud sigue siendo la misma. El aumento en la ROC y la R2 de la AUC indica que el modelo está captando eficazmente los fenómenos, mientras que el impacto positivo de las arquitecturas con co-embeddings en el RMSE puede apuntar hacia un clasificador con más certeza, polarizando los valores de predicción cerca de 0 y 1.

Figure 7.6: Visualización de las reglas internas de un Árbol de Decisión entrenado en el curso 6. Los nodos indican la característica y el valor utilizado para la decisión. Los nodos sombreados corresponden a los puntos de decisión, con cuadrados rojos para la deserción y círculos verdes para la etiqueta de no deserción.




Vale la pena notar que el equipo ganador de KDDCup 2015 alcanzó un AUC de 0.91 con un modelo ensamblado. Esos resultados se obtuvieron con el conjunto de datos de prueba oficial, al que no tenemos acceso. Además, la solución ganadora utilizó toda la información disponible, mientras que nosotros sólo utilizamos el identificador de cada elemento del curso, con el fin de evaluar los modelos en un escenario de información mínima. En este trabajo, no nos hemos centrado en el rendimiento, la optimización de hiperparámetros o el ensamblaje de diferentes modelos, sino en la evaluación del impacto de los embeddings con modelos más simples. Nuestro objetivo es obtener una visión del tipo de información capturada por los modelos neuronales. Basándonos en esto, podemos distinguir en qué escenarios pueden ser recomendados, en términos de complejidad de la tarea e información disponible para el entrenamiento.

El hecho de que una arquitectura neuronal sea superada por una regresión logística indica que no es el modelo correcto para esta tarea. Aunque el uso de los embeddings conjuntos aumenta el rendimiento con respecto a los embeddings disjuntos, no consideramos que estos resultados sean competitivos con otros métodos de vanguardia. Sin embargo, todavía nos permite comparar

| Modelo | AUC ROC | R2 | RMSE | Exactitud |
|---|---|---|---|---|
| LR | 0.848 | 0.357 | **0.320** | 0.866 |
| LSTM | 0.831 | 0.331 | 0.333 | 0.860 |
| E-LSTM | 0.843 | 0.356 | 0.327 | 0.864 |
| Co-ELSTM | **0.852** | **0.368** | 0.322 | **0.867** |

Table 7.4: Desempeño promedio para predicción de deserción, calculada sobre todos los cursos.

los diferentes tipos de embeddings.

Otro punto a considerar son las representaciones obtenidas en sí mismas. La regresión logística no produce una representación, utiliza una diseñada por humanos y que requiere un análisis previo de los datos. Por el contrario, un modelo neuronal podría ser utilizado tal cual, y portado entre diferentes plataformas. Teniendo en cuenta la mínima diferencia de rendimiento con los modelos neuronales, podemos concluir que los embeddings producidos por la capa recurrente no representan mejor para los fenómenos de deserción que las características artesanales. Las embeddings se construyen utilizando los elementos del curso elegidos por los estudiantes. Parece que la información capturada por la secuencia de interacciones no es relevante para la predicción de la deserción, o el modelo no ha sido capaz de generalizar a partir de datos de bajo nivel. Basándonos en el análisis preliminar, nuestra hipótesis es que el primer escenario es más probable para este conjunto de datos, aunque la situación puede ser diferente para otros conjuntos de datos.

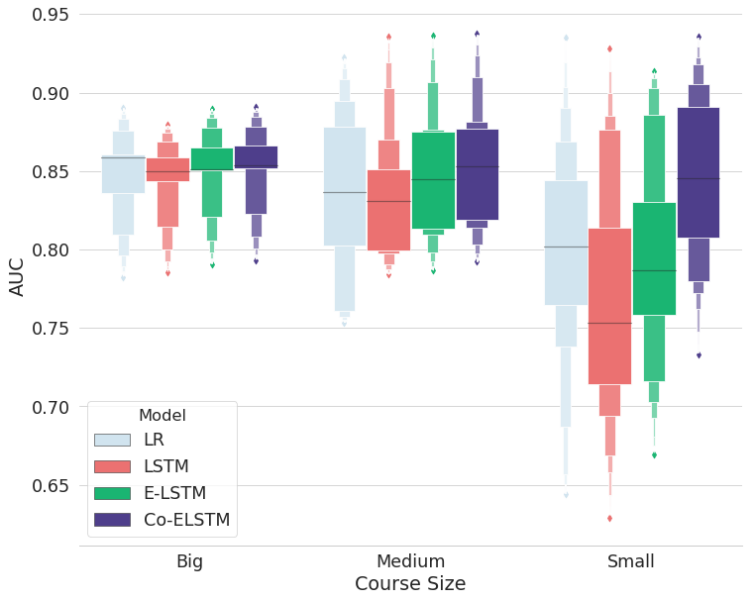
Como resultado, podemos concluir que i) los modelos neuronales están aprendiendo efectivamente a clasificar a los estudiantes, ii) hay un incremento en el rendimiento cuando se aplican representaciones conjuntas, y iii) el modelo produce una representación densa de estudiantes y elementos del curso en el mismo espacio.

### Análisis por curso

Es importante señalar que los resultados presentados se calculan sobre las predicciones de 38 cursos, lo que puede ocultar información valiosa. Proponemos hacer un análisis más profundo separando el rendimiento de los múltiples cursos. El principal factor de variación que encontramos fue la cantidad de estudiantes del curso. Por lo tanto, decidimos distinguir los resultados en tres segmentos diferentes de cursos: 5 cursos grandes con más de 6000 alumnos de formación, 9 cursos medianos con entre 5000 y 2000 alumnos y 24 cursos pequeños con menos de 2000 alumnos.

Cuando desagregamos los cursos por tamaño, podemos ver que los co-embeddings tienen un rendimiento mucho mejor en los cursos más pequeños, incluso si el rendimiento general en tales cursos es peor que en los cursos más grandes. En la figura 7.7 podemos ver que para

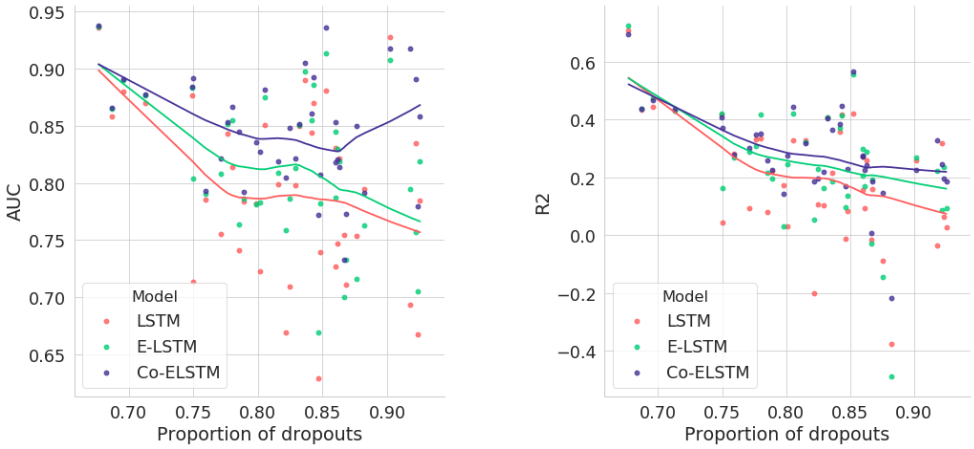
Figure 7.7: AUC para predicción de deserción, agrupados por tamaño del curso




los cursos más grandes todos los clasificadores se desempeñan indistintamente, con pequeñas variaciones, de ahí la menor dispersión del boxenplot. Los modelos LR siempre superan la arquitectura LSTM, y muestran mayores puntuaciones máximas que el E-LSTM, a costa de un mayor rango de distribución. Para algunos cursos, son mejores, pero para otros son peores, independientemente del tamaño.

Para los cursos más pequeños, donde los datos son más escasos y el rendimiento es peor, los co-embeddings parecen proporcionar generalizaciones útiles sobre los datos de bajo nivel. No sólo en comparación con las otras arquitecturas neuronales, sino también contra la LR. La información proporcionada por las secuencias parece ser insuficiente para encontrar una buena aproximación lineal del límite de decisión. Cabe señalar que, si bien los co-embeddings no parecen repercutir positivamente en el rendimiento en los cursos más grandes, tampoco lo hacen negativamente.

Explorando las diferencias de rendimiento para cada curso encontramos otro factor de correlación. La proporción de estudiantes que abandonaron, es decir, el desequilibrio de las etiquetas, influye en la dificultad de la tarea. Cuantos menos ejemplos haya en una de las clases, más difícil será aprender las diferencias con la clase mayoritaria. En la figura 7.8 a) trazamos los valores del AUC para cada tipo de modelo en cada curso, de acuerdo con la tasa de abandono en el conjunto evaluación. Se ha añadido una estimación LOWESS, para facilitar la interpretación. Vemos que, en promedio, los modelos CoE-LSTM tienen un mayor impacto en el rendimiento en los cursos con más desequilibrio de clase. Por otro lado, en la figura 7.8 b) podemos ver que el puntaje de R2 permanece más invariable con respecto a la proporción de abandonos, pero el modelo Co-ELSTM sigue superando a los otros.




(a) AUC ROC (b) R2

Figure 7.8: Rndimiento para predicción de deserción, en relación con la tasa de no-deserción del curso

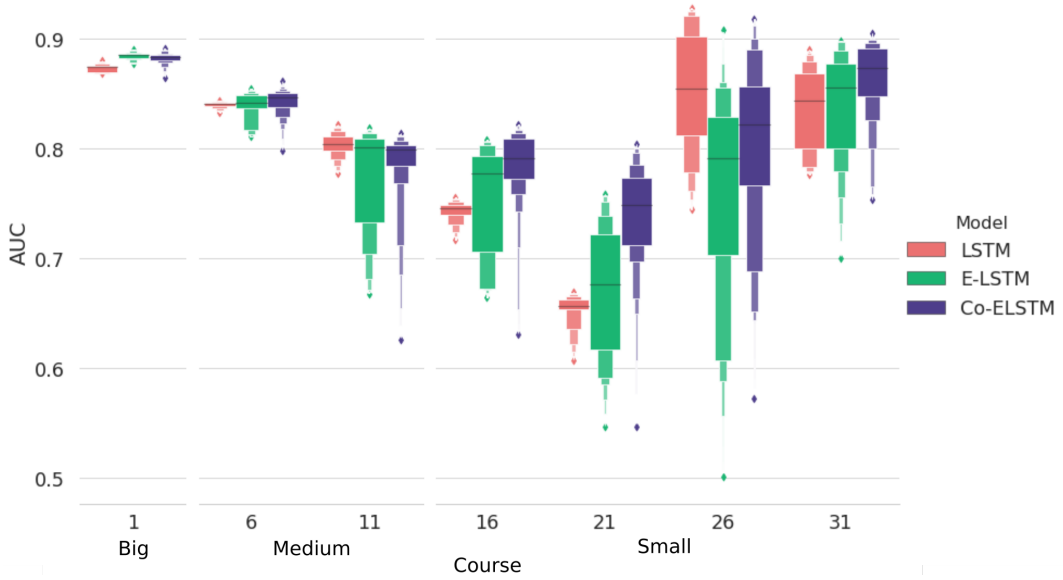
### Distribución de resultados

Debido al costo computacional de optimizar los hiperparámetros en todo el conjunto de datos, hay menos evaluaciones en todos los cursos. La mayoría de las arquitecturas fueron evaluadas sólo en los cursos de exploración. La falta de puntos de muestra impide un análisis profundo del impacto de los hiperparámetros en el rendimiento del modelo, como se hizo para la tarea KT. Sin embargo, todavía podemos tener una estimación de la estabilidad de los tres tipos de modelos utilizados, analizando la distribución de los resultados obtenidos durante las exploraciones. En la figura 7.9 presentamos varios boxenplots con los valores AUC ROC, separados por el tipo de modelo y el curso. El curso 1 se considera Grande, el curso 6 y 11 son Medianos y los restantes son Pequeños.

De la figura podemos deducir que las arquitecturas con co-embeddings, en cursos pequeños, obtienen modelos más adecuados. En el curso grande y mediano, la distribución es similar, con menor alcance para los modelos E-LSTM. Con la excepción del curso 26, el modelo LSTM es superado. Este es el único curso en el que este fenómeno ocurre, y en particular las puntuaciones R2 de los modelos Co-ELSTM superan la distribución del LSTM tanto en valor promedio como en valor máximo.

Un punto interesante para analizar es el impacto de la función de distancia utilizada. Su distribución de resultados se analiza en la Figura 7.10. A diferencia de la tarea KT, las funciones *norm* y *sigmoid* no disminuyeron el rendimiento, siendo las arquitecturas más exitosas. No observamos un impacto importante de las otras funciones de distancia, con la excepción de la función *sq* (cuadrado), que produjo modelos de rendimiento anormalmente bajo. Este resultado confirma la importancia de la función de distancia, y su impacto en el modelado de los estudiantes.

Figure 7.9: Distribución de AUC para los diferentes cursos de exploración, dividido por el tipo de modelo.

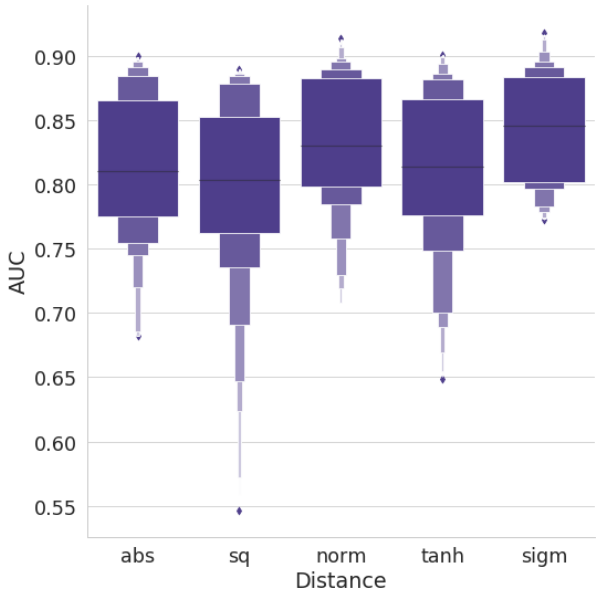



Nuestra hipótesis para este fenómeno se basa en la naturaleza de las dos tareas, y la distribución de los embeddings en el espacio latente. Al predecir el rendimiento de los estudiantes en el conjunto de datos de ASSISTments, esperamos que los estudiantes y los problemas estén cerca. Las funciones *sigmoid* y *norm* están limitadas al rango de valores [0, 1], polarizando los valores de los embeddings demasiado cercanos o demasiado alejados. En particular, son las dos únicas funciones que proyectan valores cercanos a cero a valores no nulos. Esto indica la importancia de distinguir cuándo los estudiantes comienzan a estar cerca de ciertas lecciones. Intuitivamente, si un estudiante ha alcanzado cierto punto en el curso, es decir, ciertas lecciones, entonces la probabilidad de abandono es menor. La secuencia de interacciones también será mayor, lo que es uno de los indicios más fuertes de abandono.

En cuanto a los hiperparámetros que producen los mejores modelos, varían de un curso a otro. Los tamaños de embedding de 50 y 20 son más exitosos para los cursos pequeños, 50 y 100 para los cursos medianos, y 100 para los cursos grandes. Estos resultados se ajustan a la intuición explicada anteriormente: los cursos más pequeños tienen menos elementos de curso, y por lo tanto no hay necesidad de un gran número de neuronas para representarlos. Otro punto interesante es que el rendimiento de los modelos E-LSTM no parece estar afectado por los diferentes tamaños de embedding de los cursos.

El número de pasos para retropropagar (MaxPasos) más exitosos fue de 300, aunque en los cursos grandes la diferencia no fue notable. Los modelos LSTM no podían entrenarse con este número de pasos debido a las limitaciones de memoria, el valor de mejor rendimiento era también el máximo: 100. Esta evidencia apoya nuestra hipótesis inicial de que las secuencias largas son necesarias para la predicción de la deserción. Otro factor, además de las características del

Figure 7.10: Distribución de AUC para diferentes funciones de distancia.




problema en sí, que podría explicar este fenómeno es el hecho de que las interacciones en el conjunto de datos de KDDCup son más granulares que el conjunto de datos de ASSISTments, y por lo tanto contienen menos información individualmente. Debemos ver muchas interacciones para percibir el patrón general.

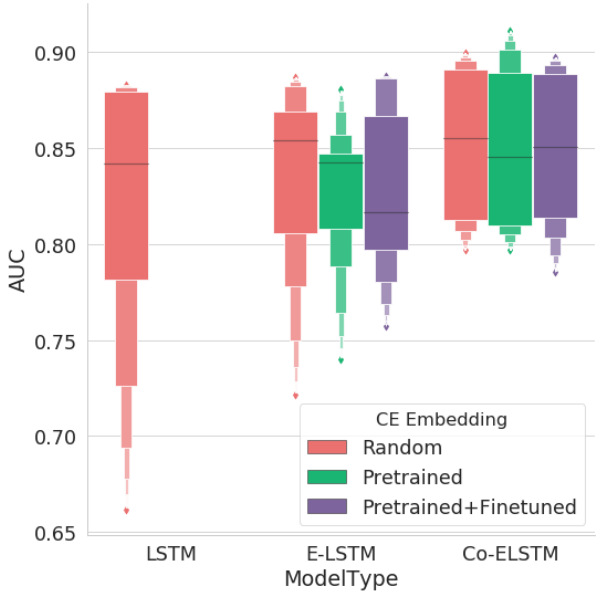
La tasa de aprendizaje de 0,01 impactó negativamente el rendimiento, especialmente en los cursos más pequeños. Estos resultados son consistentes con la tarea de KT. Las variaciones de los tamaños de los lotes no mostraron impacto en el rendimiento. Finalmente, el mejor valor de abandono fue de 0,3. Aunque no realizamos una búsqueda exhaustiva de hiperparámetros, los resultados presentados esclarecen ciertos aspectos del rendimiento del clasificador y los requisitos de un modelo neuronal que resolvería el problema con éxito.

### Impacto de los embeddings preentrenados

Para completar este análisis, en esta sección comparamos en detalle el rendimiento de los modelos inicializados con embeddings preentrenados y con embeddings aleatorios. En la figura 7.11 presentamos las puntuaciones de la AUC divididas por el tipo de embedding de elemento del curso (EEC), para comparar el rendimiento general de los métodos con embeddings con y sin un paso de preentrenamiento. Cada boxenplot muestra la distribución del modelo de mejor rendimiento usando ese EEC en particular. No mostramos la división entre los tamaños de los cursos, ya que las tres categorías siguen los mismos patrones.

La primera observación que podemos hacer es que los embeddings de elementos de curso aleatorios funcionan mejor que los preentrenados. Las diferencias en el impacto de los embeddings

Figure 7.11: Distribuciones del AUC para embedding de elemento del curso (EEC) para predicción de deserción, agrupados por tipo de modelo




preentrenados entre las dos tareas de pretexto pueden deberse a la diferencia en el tipo de clasificación. La predicción de deserción es una tarea de etiquetado de secuencias, mientras que el rastreo de conocimientos predice un resultado para cada elemento de la secuencia. La información necesaria para detectar la deserción está codificada en toda la secuencia, no en las propiedades de los elementos individuales del curso. Los métodos como word2vec no tienen por objeto caracterizar los elementos con respecto a la secuencia completa sino más bien en relación con los elementos que los rodean. Es coherente entonces que inyectar ese tipo de información no ayuda, e incluso dificulta el aprendizaje. Además, también hemos observado en los resultados de los modelos LR que el uso de la información individual de los elementos de la secuencia no es necesario para obtener buenas predicciones.

Los modelos E-LSTM tienen el mismo comportamiento que en la tarea KT, perdiendo rendimiento con el uso del paso de finetune, mientras que los modelos Co-ELSTM no se ven muy afectados por la incorporación de valores iniciales. Esto aboga por una superioridad de los embeddings conjuntos sobre los embeddings disjuntos, en el sentido de que producen modelos más robustos.

## 7.5 Predicción por períodos

Buscando un escenario más realista, hemos implementado una predicción por períodos, analizada en la sección 3.2.5. El objetivo es emular las condiciones en las que un sistema de detección temprana podría ser utilizado para la prevención del abandono escolar. Para ello, dividimos cada curso en períodos de 7 días. Después de esto, tomamos como instancias de entrenamiento

la secuencia de interacciones hasta el final del período. Asignamos a cada instancia una etiqueta que representa si el estudiante tiene actividad en el siguiente período. En este conjunto de datos, todos los cursos tienen un lapso de 4 semanas, lo que permite evaluar 3 períodos.

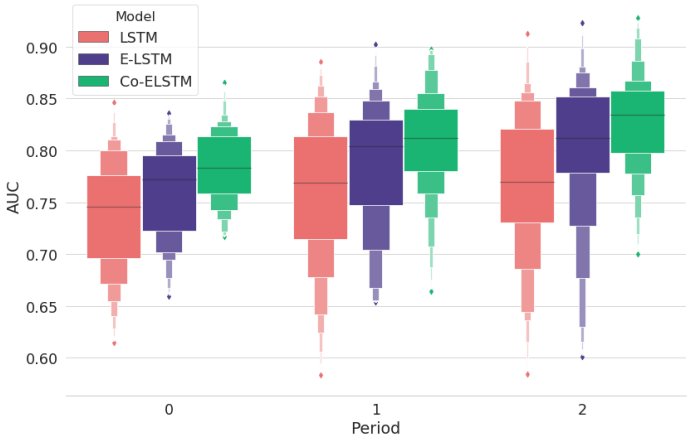
Para predecir la actividad en la segunda semana, el clasificador se entrena con todas las secuencias de la primera semana. Para predecir la tercera, el modelo se entrena desde cero con las secuencias de los dos períodos anteriores, en lugar de volver a entrenar el clasificador anterior con el nuevo período. Se utiliza lo mismo para todos los períodos siguientes.

En este escenario, es imperativo que no entrenemos los modelos con información futura no disponible mientras se enseña el curso. Por esta razón, no es posible utilizar embeddings preentrenados.

### 7.5.1 Resultados

En esta sección, presentamos los resultados de la evaluación propuesta en la sección anterior. El índice del período indica el número de la última semana utilizada para el entrenamiento.

Figure 7.12: AUC para predicción por períodos

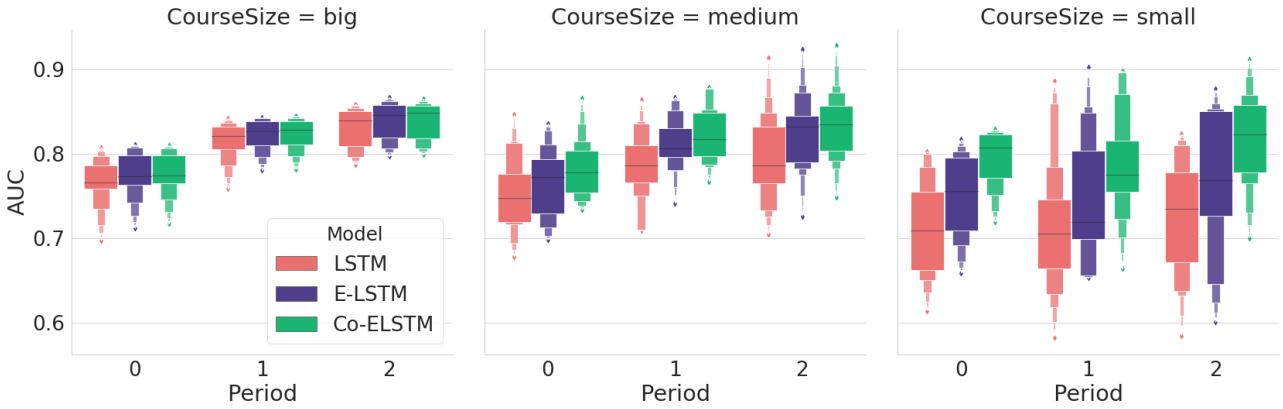



En la figura 7.12 mostramos los resultados de la predicción por períodos, presentando el AUC de los tres tipos de clasificadores en todos los cursos, divididos por períodos. Como era de esperar, el rendimiento de todos los clasificadores mejora en los períodos posteriores, donde hay más información para predecir el abandono.

Podemos observar que los modelos con embeddings superan a los modelos LSTM en todos los períodos, por lo que la generalización proporcionada por los embeddings es útil para esta tarea también. La mediana de los valores de rendimiento en los embeddings disjuntos y conjuntos son cercanos en los dos primeros períodos. Sin embargo, la diferencia en los cuartiles indica que los modelos conjuntos están superando de hecho a las representaciones disjuntas.

Además, inspeccionando la figura 7.13 se puede ver que la diferencia entre los modelos es más pronunciada en los cursos pequeños, pero también es notable en los cursos grandes.

Figure 7.13: AUC para la predicción del período, separado por el tamaño del curso




## 7.6 Discusión

Congruentemente con los resultados para KT, vemos que los embeddings mejoran el rendimiento y los co-embeddings no perjudican el rendimiento en la tarea de predicción de la deserción. Además, hemos encontrado algunos casos en los que los co-embeddings mejoran el rendimiento: en cursos con pocos estudiantes, en cursos con más desequilibrio de clase y utilizando sólo información parcial (secuencias más cortas). Cuando se dispone de información completa, los embeddings conjuntos no suponen una diferencia con respecto a los embeddings más sencillas, pero sí en contextos importantes, como los cursos que están siendo dictados al momento de la predicción o los cursos con una población más pequeña.

Parece que los co-embeddings son capaces de captar sutilezas que escapan a otros clasificadores utilizando representaciones generadas manualmente y basadas en información de dominio, como la regresión logística. Esto apunta a que los co-embeddings están aprendiendo una representación que es hasta cierto punto útil para la tarea de predicción de la deserción escolar.

# Capítulo 8

# Visualización de representaciones vectoriales

A lo largo de este trabajo, nos hemos centrado exclusivamente en las interpretaciones de las métricas de rendimiento. En este capítulo, exploraremos brevemente las posibilidades de interpretación de las representaciones obtenidas, buscando patrones relevantes. Esta etapa proporcionará una visión sobre la pregunta de investigación 3b, que se refiere al impacto de los co-embeddings en la interpretabilidad.

Antes de presentar cualquier visualización, es importante destacar que las redes neuronales no pueden, en la medida de nuestro conocimiento, ser directamente interpretadas. En consecuencia, el objetivo de este análisis no es desenmarañar el comportamiento interno de los modelos, ni explicar en términos intuitivos qué criterios se utilizan para la clasificación. Buscamos, en cambio, comprender cómo el modelo está proyectando los embeddings hacia un espacio que es útil para la clasificación. En particular, buscamos la diferencia introducida en este espacio cuando se utilizan embeddings conjuntos. La modificación del modelo mediante la inyección de conocimiento de dominio, como en el caso de los embeddings conjuntos, es un intento de ayudar a los clasificadores a representar los ejemplos en un espacio más adecuado. Nuestra hipótesis es que la visualización de la representación obtenida ayudará a comprender el impacto de las modificaciones propuestas.

La interpretación de los vectores en un espacio de altas dimensiones, especialmente aquellos que representan procesos complejos, no es una tarea trivial. Lo ideal sería contar con expertos en el tema que revisen los embeddings, determinen qué patrones son relevantes o novedosos, actúen sobre la nueva información y evalúen el impacto en la calidad de la educación en un escenario real, como lo hace Pardos and Horodyskyj [2017]. Desafortunadamente, no contamos con tales recursos o acceso a un curso en vivo, sin embargo, visualizaremos los embeddings desde una perspectiva similar, aunque más metodológica. El objetivo de esta sección es entender qué tipo

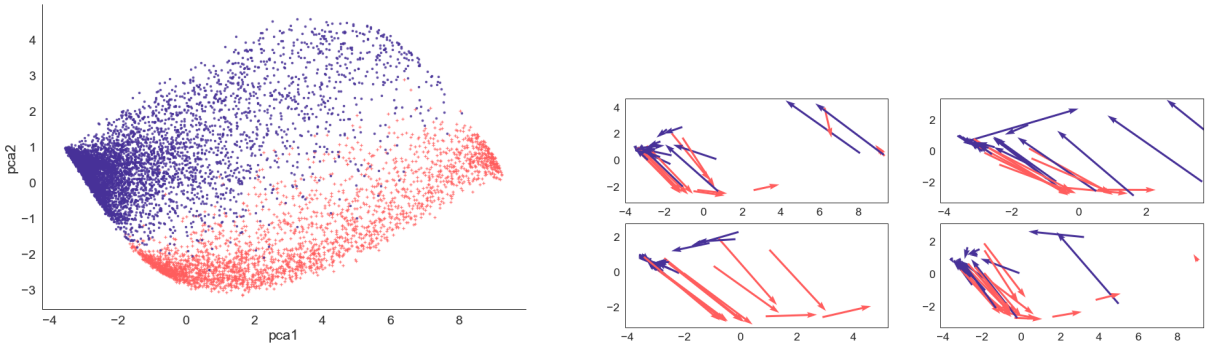
de información puede ser visualizada usando embeddings, y para qué podría ser usada.

Para crear diagramas de dispersión con embeddings que tengan más de 3 dimensiones, es necesario utilizar algún algoritmo de reducción de la dimensionalidad. Estos algoritmos proyectan embeddings a 2 ó 3 dimensiones con una mínima pérdida de información, y preservando la mayor cantidad posible de la variabilidad original. En este trabajo, evaluamos dos opciones: el primer método es el Análisis de Componentes Principales (PCA), una técnica basada en transformaciones ortogonales para obtener los componentes principales de una matriz [Pearson, 1901]. Los componentes principales son conjuntos de bases ortogonales no correlacionados, ordenados según su varianza. Es una técnica común utilizada para visualizar vectores de alta dimensión.

El segundo método que usamos es t-sne, propuesto por van der Maaten and Hinton [2008] y ampliamente utilizado para la visualización desde entonces. A diferencia de PCA, t-sne puede capturar relaciones no lineales en el espacio de origen, ya que se basa en distribuciones gaussianas. Utilizando tanto la estructura local como la general del espacio original, no comprime los puntos del espacio de destino, lo cual es un problema común en PCA.

Hay un último aspecto al que hay que prestar atención cuando se intenta interpretar un espacio de altas dimensiones. Las visualizaciones en dos dimensiones no proporcionan suficiente información de que un cierto patrón no está presente en el espacio. Sólo pueden mostrar la información presente en los componentes principales, u otros aspectos seleccionados por el algoritmo de proyección.

## 8.1 Rastreo de conocimiento




(a) Embeddings de estudiantes coloreados por etiqueta

(b) Cambios producidos por elementos del curso

Figure 8.1: Visualización de los embeddings de los estudiantes producidas por una arquitectura E-LSTM con embeddings inicializadas al azar. Los vectores se proyectan usando PCA.

Analizaremos los primeros resultados de la tarea de Rastreo de Conocimientos (KT). En la figura 8.1 (izquierda) podemos observar embeddings de una muestra aleatoria de 100 estudiantes del

conjunto de evaluación, proyectada con el algoritmo PCA. El modelo utilizado para obtener los embeddings es la mejor arquitectura E-LSTM, con embeddings inicializadas aleatoriamente. Para calcular cada punto, utilizamos la salida de la capa recurrente para toda la secuencia. El color y la forma de los puntos representan la etiqueta, con cruces rojas para la etiqueta negativa y puntos azules más oscuros para la etiqueta positiva. Podemos ver efectivamente que las interacciones etiquetadas como "incorrectas" tienden a agruparse hacia la parte inferior y derecha del gráfico, y lo contrario sucede para la otra clase. Proyectado con PCA hay una clara separación lineal entre ambas clases, lo que indica que el espacio está optimizado para la tarea de predicción.

En la figura 8.1 (derecha) vemos el cambio entre los estados después de una interacción. En otras palabras, se muestra cómo los estados de los estudiantes se mueven en el tiempo. Los datos provienen del mismo clasificador, así que ambas figuras están trazando el mismo espacio. Cada gráfica de campo representa los cambios producidos por una sola acción, y cada flecha comienza en la posición original de un estudiante, y termina en la posición del estudiante después de realizar la acción. El color azul indica que el problema está etiquetado como "correcto", y el color rojo la otra etiqueta. Podemos ver que en general, las transiciones de arriba a abajo son ejercicios incorrectamente resueltos, y las transiciones de abajo a arriba son correctamente resueltas. Esto indica que hay un paralelismo entre la forma en que estos puntos se mueven en el espacio y el desempeño de los estudiantes, lo cual fue una de nuestras hipótesis iniciales.

La proyección t-sne, presentada en la figura 8.2, muestra patrones similares, aunque las transiciones parecen más caóticas y no son linealmente separables. Estas conclusiones son válidas para diferentes clasificadores, acciones y muestras de estudiantes.

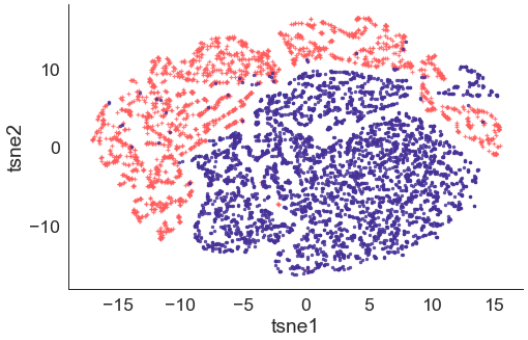
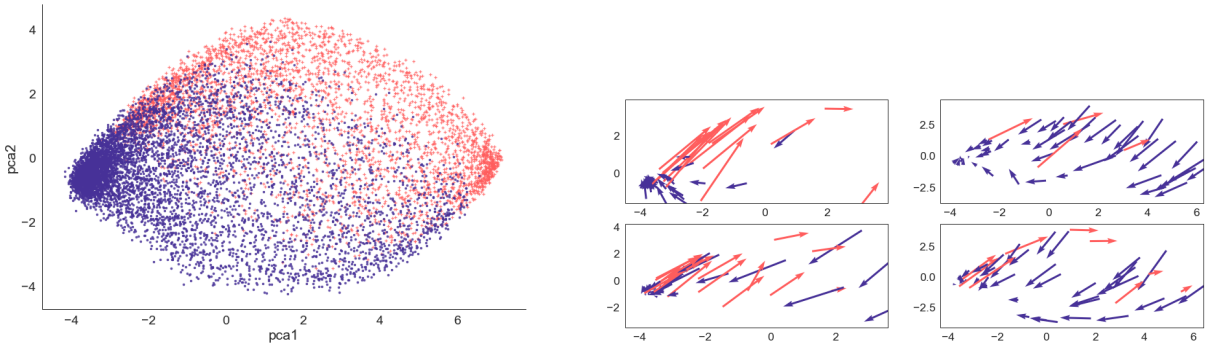



Figure 8.2: Visualización de los embeddings de los estudiantes producidas por una arquitectura E-LSTM con embeddings inicializadas al azar. Los vectores se proyectan usando tsne.

En la figura 8.3 vemos que las proyecciones de embeddings del estudiante (izquierda), y sus correspondientes cambios producidos por las acciones (derecha). A primera vista, parece como si el espacio conjunto no separara las dos clases así como el espacio desarticulado. Sin embargo, esto sólo ocurre porque hay más operaciones entre el embedding del estudiante y la capa de clasificación, es decir, la función de distancia. De hecho, como se muestra en la figura 8.4



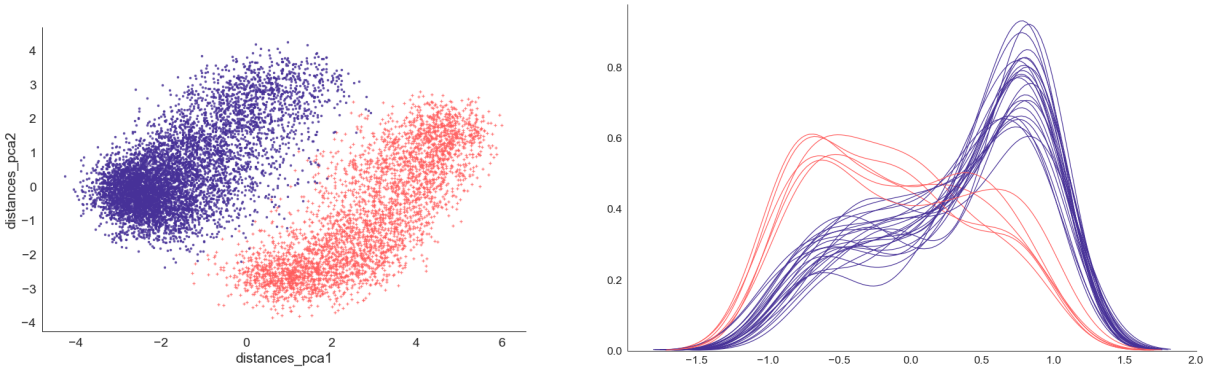

(a) Student embeddings colored by label

(b) Course element produced change

Figure 8.3: Visualización de los embeddings de estudiantes producidas por una arquitectura CoE-LSTM con activación tanh, embeddings pre-entrenadas y afinadas. Los vectores se proyectan usando la etiqueta PCA.

(izquierda), la distancia entre el embedding del elemento de curso y el embedding del estudiante, que es la entrada a la capa recurrente, están claramente agrupadas de acuerdo con su etiqueta.

En la figura 8.4 (derecha), podemos ver la distribución de 20 "vectores de distancia" seleccionados al azar, sin ninguna proyección. Cada vector de distancia es la entrada a la capa recurrente para un solo paso de tiempo, obtenido a través de la aplicación de la función $\delta$ a la diferencia entre el embedding del estudiante y el elemento del curso. El análisis de esta distribución nos ayudará a comprender si estas distancias son relevantes para la clasificación en su espacio original, y no sólo agrupadas según su etiqueta por el algoritmo PCA. De hecho, las distancias que corresponden a una interacción negativa, mostradas en rojo, tienen una forma de diferencia clara como interacciones positivas, mostradas en azul.




(a) Distancia entre el embedding del estudiante y el embedding del elemento del curso, coloreado por la etiqueta

(b) Distribución de los vectores de distancia entre el embedding del estudiante y del elemento del curso

Figure 8.4: Visualización de los embeddings de estudiantes producidas por una arquitectura CoE-LSTM con activación tanh, embeddings pre-entrenadas y afinadas. Los vectores se proyectan usando PCA.

La diferencia más importante entre el espacio conjunto y el espacio disjunto que observamos está relacionada con este fenómeno. Recordemos que los elementos del curso se representan

como un embedding de base, a la que añadimos un "embedding positivo" si la interacción se etiqueta como correcta. En el espacio disjunto, no observamos ninguna optimización realizada a el embedding positivo. En el espacio de unión, el embedding positivo está claramente separada del embedding base.

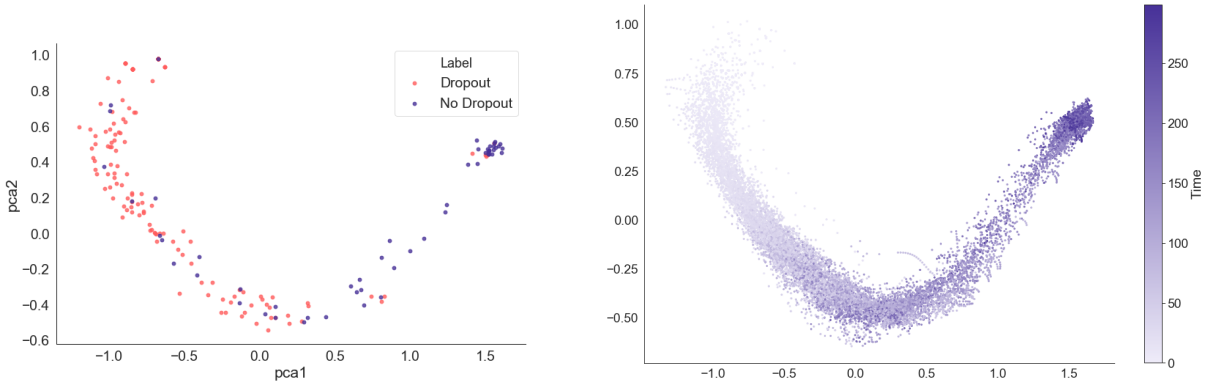
Queda por ver si un sistema de recomendación que utilice esta información supera los enfoques utilizados en los ITS actuales. Dado el embedding de un estudiante, sabemos la dirección deseada en la que queremos mover el embedding (en este caso, a la izquierda), y tenemos ejemplos de dónde ha movido cada lección a estudiantes que estaban cerca de regiones similares. Entonces, dado un punto de partida, debería ser posible seleccionar una lección que mueva al estudiante más cerca de la posición final deseada. Para probar este tipo de hipótesis, es necesario un proceso de prueba A/B completo.

Sin embargo, no hay evidencia sólida de que las regiones del espacio latente se correlacionen con el resultado de la siguiente interacción. En otras palabras, no hay evidencia de que si un estudiante es conducido a una región, el éxito de la siguiente interacción será más probable.

## 8.2 Predicción de deserción escolar

Llevaremos a cabo un análisis similar a los embeddings obtenidos con el conjunto de datos de KDDCup 2015. En la figura 8.5 (derecha) mostramos la posición de los embeddings de los estudiantes en sus últimas interacciones, coloreados por etiqueta y proyectados usando PCA. Podemos ver, como en el caso de KT, que hay una correlación entre la posición de los embeddings de los estudiantes y su clase, que parece estar relacionada con el tiempo de la interacción. Esto se ilustra en la figura 8.5 (izquierda), donde mostramos todos los embeddings de los estudiantes coloreadas por el tiempo de la interacción. Podemos ver claramente que los componentes principales están relacionados con el tiempo. Los estudiantes que tuvieron su última acción hacia la derecha del gráfico tuvieron secuencias más largas, y son menos propensos a abandonar.

En la figura 8.6, mostramos las visualizaciones equivalentes pero obtenidas con una arquitectura Co-ELSTM. En este caso, también se utiliza embeddings inicializados aleatoriamente, con una función *tanh* $\delta$. Los patrones observados son consistentes a través de las arquitecturas y cursos, cuando el clasificador obtiene buenos resultados. Por el contrario, en la figura 8.7 presentamos embeddings obtenidas con una arquitectura E-LSTM de bajo rendimiento. Podemos ver que los embeddings aparecen de forma más aleatoria, menos alineados según el tiempo de interacción. Para la arquitectura CoE-LSTM del mismo curso, por otro lado, se observaron más claramente 8.8. Esto sugiere que, para esta tarea, el espacio del estudiante necesita capturar la información temporal para producir un buen modelo.



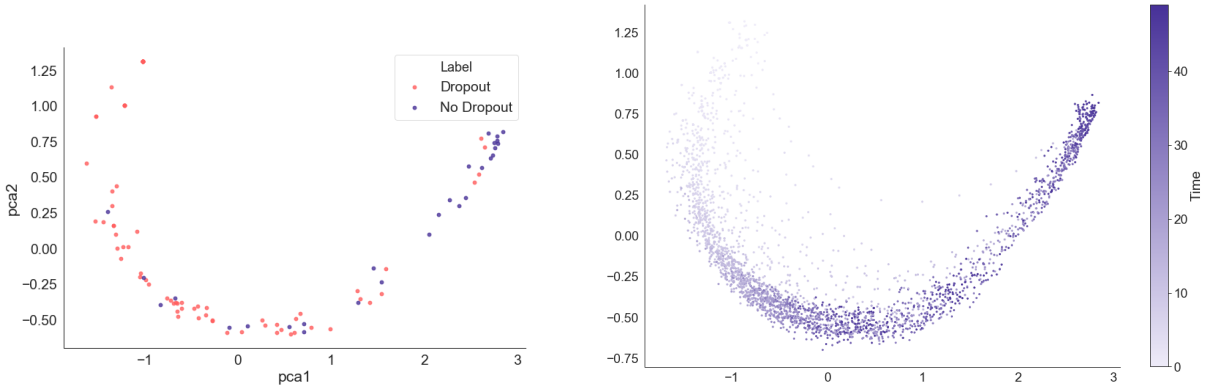

(a) Último embeddings del estudiante coloreado por etiqueta.

(b) Todos los embeddings de cada estudiante, coloreados por el tiempo de la interacción

Figure 8.5: Visualización de embeddings de estudiantes del curso 1 producidos por una arquitectura E-LSTM con embeddings inicializados al azar. Los vectores se proyectan usando PCA.

Representaciones de estudiantes proyectados con la técnica t-sne, como se muestra en la figura 8.9 (izquierda), no dividen visualmente los estados de los estudiantes según su etiqueta. En la subfigura derecha, sin embargo, siguen una dimensión temporal. Las estructuras observadas, con una forma parecida a la de los "gusanos", corresponden a diferentes estudiantes. Es importante señalar que el algoritmo de proyección tiende a mostrar este tipo de patrón. Por ejemplo, estructuras similares, aunque menos evidentes, se ven en la figura 8.2 y en el trabajo de Pardos and Horodyskyj [2017]. Contrariamente a la propuesta del trabajo citado, el algoritmo t-sne parece ser inadecuado para analizar las interacciones de los estudiantes para la tarea de predicción del abandono escolar. Sin embargo, un análisis de casos de secuencias individuales, usando más información de la que tenemos disponible, podría conducir a mejores interpretaciones.

También analizamos la distribución de las distancias entre el elemento del curso y los embeddings de los estudiantes. En la figura 8.10 presentamos la distribución de los valores de estos vectores de distancia, para las funciones *sigmoid* y *sq* $\delta$ (izquierda y derecha respectivamente). Los vectores visualizados corresponden a la última interacción de las secuencias seleccionadas al azar, cuando se conoce la etiqueta. Usamos el color rojo para representar la deserción y el color azul para representar la no deserción. Además de la diferencia esperada entre la forma de las distribuciones, emergen algunos patrones, aunque no son tan claros como en la tarea KT. Podemos ver que para la función *sigmoid*, la distribución de las distancias de abandono tiene una curtosis más alta, mientras que las distancias de no abandono exhiben una densidad ligeramente mayor en las colas. Para los estudiantes que abandonan, hay más dimensiones en las que el embedding del estudiante está cerca de el embedding de la lección, que es proyectada por la función *sigmoid* a un valor cercano a 0,5. Para la función *sq*, en la subfigura derecha, los dos tipos de distribuciones son muy similares, con algunos cambios en la forma de la cola.




(a) Último embedding de estudiantes coloreado por etiqueta.

(b) Todos los embeddings de cada estudiante, coloreados por el tiempo de la interacción

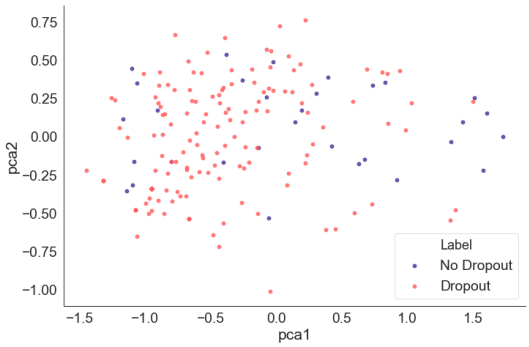
Figure 8.6: Visualización de embeddings de estudiantes del curso 1 producidos por una arquitectura CoE-LSTM con embeddings inicializadas al azar. Los vectores se proyectan usando PCA.

Las secuencias de deserción concentran más valores en el intervalo $(0,5;1)$, mientras que las distribuciones sin deserción concentran valores en el $(1,5;2)$. Esta evidencia sugiere que algunas dimensiones del espacio están capturando efectivamente las causas latentes relacionadas con la deserción.

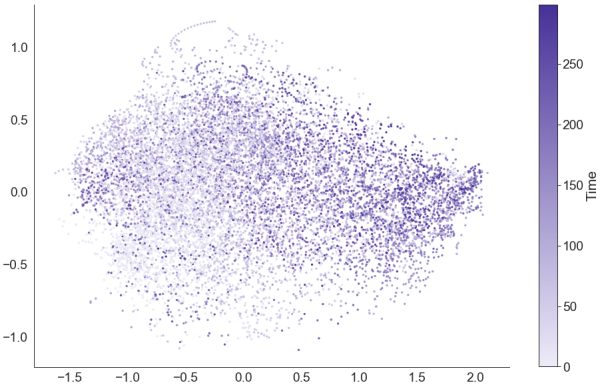
## 8.3 Discusión

Hemos visto que los embeddings representan adecuadamente las interacciones entre los estudiantes y los elementos del curso, distinguiendo claramente los resultados exitosos de los no exitosos. Hemos mostrado diferentes modos de visualización que tendrán que ser interpretados junto con un experto en la materia para explorar cómo pueden ser utilizados en la interpretación de los datos y el despliegue de soluciones.

En la tarea de predicción de la deserción escolar, la representación aprendida se basa principalmente en la dimensión temporal de los datos, que es un factor importante del problema. Las representaciones con embeddings disjuntos aplicadas en cursos más pequeños no encuentran este tipo de patrones, mientras que las representaciones conjuntas sí los encuentran. Esto parece indicar que los co-embeddings están encontrando una representación de los datos que capta las generalizaciones adecuadas, a partir de una menor cantidad de datos.

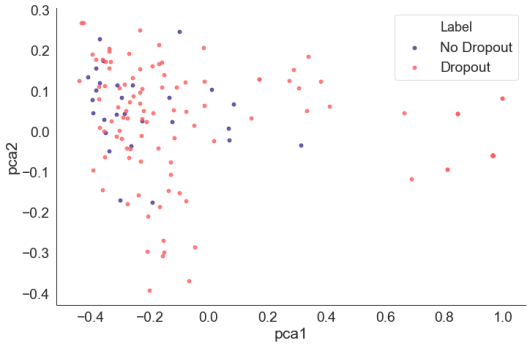



(a) Último embedding de estudiantes coloreado por etiqueta.

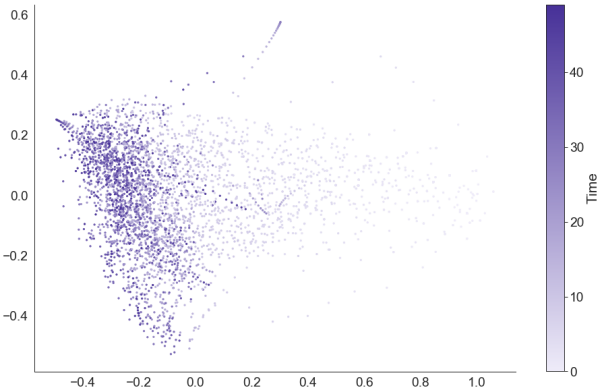



(b) Todos los embeddings de cada estudiante, coloreados por el tiempo de la interacción

Figure 8.7: Visualización de embeddings de estudiantes del curso 21 (pequeño) producidos por una arquitectura E-LSTM con embeddings inicializadas al azar. Los vectores se proyectan usando PCA.

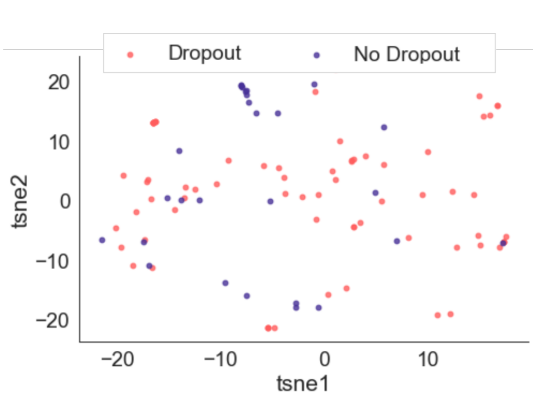



(a) Último embedding de estudiantes coloreado por etiqueta.

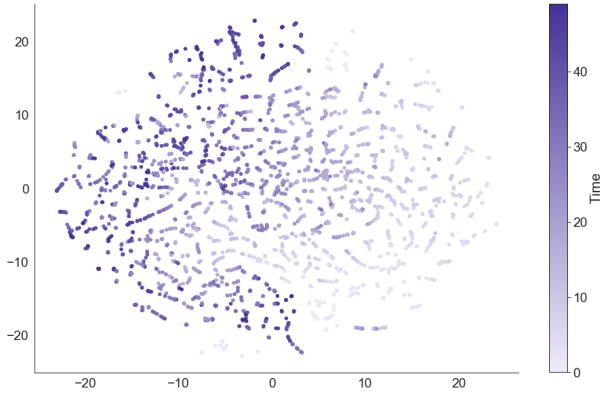



(b) Todos los embeddings de cada estudiante, coloreados por el tiempo de la interacción

Figure 8.8: Visualización de embeddings de estudiantes del curso 21 (pequeño) producidos por una arquitectura CoE-LSTM con embeddings inicializadas al azar. Los vectores se proyectan usando PCA.

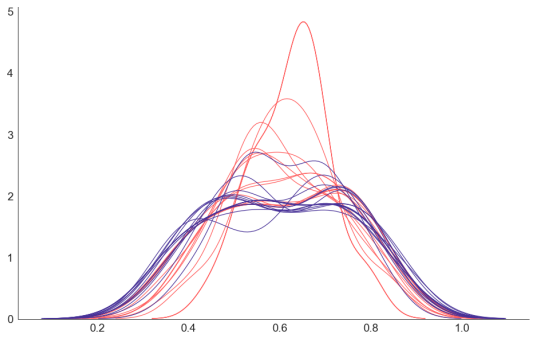



(a) Último embedding de estudiantes coloreado por etiqueta.

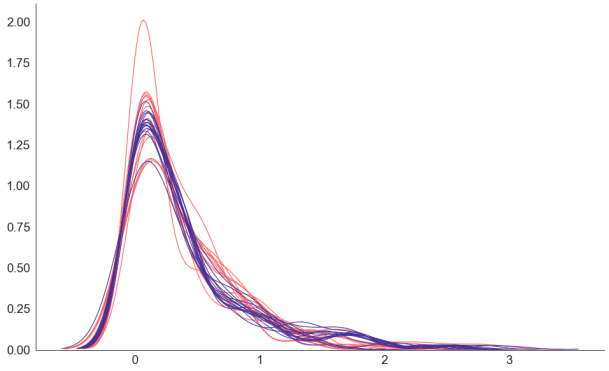



(b) Todos los embeddings de cada estudiante, coloreados por el tiempo de la interacción

Figure 8.9: Visualización de embeddings de estudiantes del curso 1 producidos por una arquitectura CoE-LSTM con embeddings inicializadas al azar. Los vectores se proyectan usando t-sne.




(a) Arquitectura Co-ELSTM con función de *sigmoid delta*




(b) Arquitectura Co-ELSTM con función de *sq delta*

Figure 8.10: Distribución de la distancia entre el elemento del curso y los embeddings de los estudiantes, en la última interacción de las secuencias.

# Capítulo 9

# Conclusiones

Hemos presentado una exploración de las técnicas de aprendizaje de la representación aplicadas a la minería de datos educativos. Más concretamente, el objetivo de este trabajo era evaluar el impacto de los embeddings neuronales en dos tareas concretas: la predicción de la deserción escolar y el seguimiento de los conocimientos (KT). Propusimos un enfoque para modelar el comportamiento de los estudiantes en plataformas de aprendizaje con embeddings conjuntos de estudiantes y de elementos del curso.

En la sección 2.6 propusimos varias preguntas de investigación que dieron forma a nuestro trabajo y el contenido de esta tesis. Resumimos en esta sección las conclusiones que obtenemos tras evaluar la arquitectura en las dos tareas propuestas.

En cuanto a la relación de esta propuesta con los modelos anteriores (pregunta 1), hemos encontrado que, para la tarea KT, la mayoría de los enfoques de modelado de estudiantes se basan en representar directamente una simplificación del proceso de aprendizaje. Se analizan las posibles causas latentes del fenómeno y se construyen modelos teóricos basados en las conclusiones. En este sentido, nuestro trabajo está alineado con investigaciones previas en el campo. Basamos la propuesta principalmente en los trabajos de Piech et al. [2015], y Reddy et al. [2016] y su uso de embeddings para representar en el mismo espacio estudiantes y elementos del curso.

Las propuestas para la tarea de predicción del abandono escolar son más variadas y menos teóricas, posiblemente debido a la falta de consenso sobre la definición de abandono escolar, sus causas latentes y conjuntos de datos de referencia para comparar diferentes enfoques. En general, la atención se centra en resolver la tarea en lugar de modelar al estudiante. Como resultado, es difícil encontrar puntos de conexión con nuestra propuesta, aunque varias investigación previas también utilizan redes neuronales.

En nuestro trabajo, los embeddings conjuntos (co-embeddings) se obtienen con una arquitectura neural recurrente modificada para modelar explícitamente la relación entre embeddings de

estudiantes y elementos del curso, incorporando conocimiento de dominio sobre la tarea a resolver directamente en la arquitectura neuronal. De esta manera, esperamos superar algunas limitaciones de los métodos anteriores, que utilizan representaciones obtenidas manualmente, embeddings disjuntos, o embeddings conjuntos no neuronales.

Por un lado, los embeddings de elementos de curso eliminan la necesidad de etiquetar los elementos de curso con información de habilidades o cualquier otro metadato, lo cual es común en las plataformas ITS pero muy costoso en los MOOC. Y por otro lado, los embeddings neuronales crean representaciones más robustas al no depender del modelado de dominio experto de las causas de ciertos fenómenos, logrando así una mayor cobertura. Por último, los embeddings conjuntos reintroducen el conocimiento del dominio en el modelo, conservando la flexibilidad de las representaciones proporcionadas por las redes neuronales. A continuación resumimos cómo los resultados experimentales proporcionan pruebas para afirmar si estas hipótesis iniciales eran correctas.

Para responder a la pregunta de investigación 2 y mostrar que el modelo es factible de construir y entrenar, propusimos modificaciones a una arquitectura recurrente de referencia que proyecta ambos embeddings en el mismo espacio. Aunque la contribución de la arquitectura en sí misma sólo es relevante para las tareas dadas, sigue principios generales. Las redes neuronales son modelos muy poderosos que pueden aprender cualquier fenómeno, pero sólo provistos de la arquitectura, los datos y el algoritmo de optimización correctos. Ante la falta de datos o la falta de recursos para realizar búsquedas extensas en el espacio de los hiperparámetros, es posible ayudar al proceso de aprendizaje añadiendo información de dominio. En este caso, se trata de una simple conexión entre dos capas, pero no hay límite para las posibles arquitecturas.

Los resultados obtenidos en el análisis experimental de los modelos nos ayudan a determinar el impacto de los embeddings, y a responder a la pregunta 3. Ambas tareas de aplicación tienen características diferentes que las hacen difíciles de resolver. KT es una tarea central para el desarrollo de Sistemas de Tutoría Inteligente, y ha sido ampliamente abordada en trabajos anteriores. Se modela a través de una tarea de pretexto de predecir qué ejercicios podrá resolver el estudiante en la siguiente iteración. Para esta tarea, los modelos que incluyen elementos del curso y embeddings del estudiante, sin utilizar la información de habilidades disponible, se desempeñan de manera similar a los métodos del estado del arte, que sí explotan la información de habilidades agregada manualmente. El uso exclusivo del identificador del problema y la incorporación del estudiante proporciona suficiente información para que estos modelos puedan predecir la etiqueta. Este tipo de modelo debería recomendarse en los casos en que no haya información adicional sobre el contenido del curso.

El uso de embeddings preentrenadas tuvo un impacto positivo en el rendimiento del clasificador, aunque fue necesario ajuste junto con el clasificador. Esto sugiere un paralelismo entre el KT y el lenguaje natural, el dominio original donde se aplicó el algoritmo word2vec utilizado para

generar los vectores de inicialización. Las embeddings preentrenadas representan cada elemento de la secuencia (elementos del curso, o palabras) que está condicionada por la representación de los elementos circundantes. Es razonable esperar que los problemas que el estudiante resolvió justo antes de una interacción sean altamente indicativos del éxito o fracaso del problema actual.

Las representaciones conjuntas no mostraron una mejora con respecto a las desarticuladas. Para esta tarea, el uso de las co-embeddings no es necesario, aunque tampoco perjudica el rendimiento. Nuestra hipótesis, también apoyada por los resultados obtenidos en la tarea de predicción de abandono, es que hay suficientes datos para modelar el problema. Cabe señalar que las funciones más exitosas $\delta$ son más fáciles de entrenar, porque se ven menos afectadas por la elección de los hiperparámetros.

La tarea de predicción de abandono escolar se basa más en modelos de dependencia y relaciones a largo plazo que capturan aspectos de los estudiantes más difíciles de evaluar automáticamente, como el compromiso cognitivo y emocional. Además, es una tarea de clasificación de secuencias, donde una única señal de error está presente al final de la serie de interacciones. Esto dificulta el entrenamiento de un modelo recurrente, porque hay poca información para propagar a través de la red.

Una regresión logística con una representación hecha a mano basada en el recuento del número y el tipo de interacciones se desempeña de forma similar a los modelos neuronales, lo que plantea interrogantes sobre cuán adecuados son los modelos recurrentes. Sin embargo, el rendimiento sólo es comparable para cursos con numerosas interacciones. Los co-embeddings funcionan mejor en cursos con menos estudiantes, secuencias con información parcial y cursos con mayor tasa de abandono. Esta mejora en el rendimiento puede observarse no sólo con respecto a la regresión logística, sino también en comparación con las representaciones disjuntas y sin embeddings. Esto parece indicar que las representaciones conjuntas son capaces de captar mejor las causas latentes que intervienen en la predicción del abandono escolar.

Hay conclusiones interesantes que se pueden inferir del comportamiento de los clasificadores. En este caso, el uso de sólo la identificación del problema es menos adecuado para representar el problema, como lo demuestra el rendimiento de las representaciones obtenidas manualmente. Sin embargo, en los casos en que ésta es la única información disponible, seguiría siendo recomendable utilizar representaciones conjuntas. Las representaciones conjuntas son, de hecho, más estables que las representaciones disjuntas, produciendo una distribución de resultados más reducida.

Hemos visto una diferencia significativa en el impacto de la función $\delta$ utilizada en las representaciones conjuntas, y los efectos están alineados con las intuiciones que teníamos sobre cada tarea de pretexto. En particular, para la tarea KT, las funciones de distancia *square*, *abs* y *tanh* dan los mejores resultados, siendo *tanh* la más estable. Las funciones *norm* y *sigmoid*,

que apagan o agrupan las señales provenientes de las interacciones en las que el estudiante está demasiado cerca o demasiado lejos del ejercicio, muestran signos de falta de adaptación y resultados muy bajos.

En la tarea de abandono, se observa lo contrario, señalando la importancia de la función *delta* cuando se modela un problema. La función *sigmoid* fue la que mejor funcionó, seguida de la función *norm*. Nuestra explicación más fuerte es el hecho de que ambas funciones proyectan diferencias nulas en valores no nulos, lo que indica que el clasificador se basa en la información de situaciones en las que los elementos del estudiante y del curso están cerca para generar mejores predicciones.

Cuando visualizamos los embeddings obtenidos, emergen algunos patrones. La interpretabilidad de estos patrones está limitada sin la intervención de un experto de dominio, caracterizaciones completas de los elementos del curso o plataformas de prueba A/B. Sin embargo, hemos observado que la posición de los embeddings de los estudiantes predice la etiqueta de la interacción, y los estudiantes siguen caminos similares en el espacio latente. Hay zonas que pueden identificarse con resultados deseables. Esto apunta a representaciones que están modelando los fenómenos adecuadamente, y podrían ser usadas para diferentes tareas con causas latentes similares.

## 9.1 Trabajo Futuro

Después de concluir este trabajo, vemos nuevas y prometedoras direcciones de investigación. El primer paso a seguir es evaluar el rendimiento de las representaciones conjuntas en diferentes conjuntos de datos, evaluando especialmente su impacto en los cursos en el momento en que se están dictando. También existe un rico campo de estudio sobre la comprensión de las redes neuronales y un enfoque sistemático de las posibles modificaciones de la arquitectura que ayuda al modelo a encontrar representaciones más adecuadas. El consenso general es que este tipo de modelos no son interpretables ni explicables, pero hay posibilidades de evaluar los espacios que crean para encontrar perspectivas sobre las causas latentes en las que se basan.

En particular, el conjunto de datos de ASSISTmets 2012 contiene anotaciones semiautomáticas de señales de afectividad en los estudiantes. Vale la pena explorar si hay correlaciones entre estas anotaciones y los embeddings optimizados para una tarea de rastreo de conocimiento. Encontrar tales correlaciones puede ayudar a entender mejor el proceso de adquisición de conocimiento.

Las representaciones obtenidas por los embeddings neuronales también pueden ser evaluadas a través de diferentes tareas de pretexto relacionadas. Una línea de trabajo futura relacionada es la evaluación del impacto de los embeddings conjuntos y desarticuladas generadas utilizando

RNN como insumo de otras tareas. Entre ellas, podemos nombrar la recomendación de contenido, la personalización de la trayectoria de aprendizaje y la estimación del compromiso.

Para la predicción de la deserción en particular, una posible mejora de la tarea es incluir en la representación otra información de libre acceso, como el tiempo de acceso, información de la sesión, dispositivos de acceso, etc. Como hemos visto, el uso exclusivo del identificador del elemento del curso no da buenos resultados. El objetivo de este trabajo no se centró en el rendimiento, pero es una tarea importante para resolver futuros desarrollos puede centrarse en la comprensión de qué tipo de representación resuelve mejor el problema.

En particular para el enfoque de detección temprana aplicado a la predicción del abandono escolar, proponemos utilizar un método de aprendizaje de refuerzo para generar predicciones precisas tras el menor número de interacciones posible. En el apéndice A describimos nuestro trabajo inicial en esta área. Aún es necesario probar empíricamente el clasificador propuesto. Se trata de una nueva vía de exploración con infinitas posibilidades, desde la optimización del algoritmo a2c hasta el uso de modelos más nuevos y sofisticados como el Aprendizaje de Refuerzo Jerárquico.

También consideramos que explorar una línea de investigación a través de técnicas de visualización y recomendación de contenido (personalización) puede dar buenos resultados. Este trabajo debe realizarse en colaboración con instituciones de enseñanza, y en estrecho contacto con expertos en la materia y estudiantes.

# Bibliography

F Amador, Ana Nobre, and Daniela Melaré Vieira Barros. *Towards a model of a didactic of eLearning: an application to education for sustainable development in Handbook of Research on Engaging Digital Natives in Higher Education Settings.*, pages 400–419. Hershey, PA: IGI Global, 04 2016. doi: 10.4018/978-1-5225-0039-1.

Kailash Arulkumaran, Marc Peter Deisenroth, Miles Brundage, and Anil Anthony Bharath. Deep reinforcement learning: A brief survey. *IEEE Signal Processing Magazine*, 34:26–38, 2017.

David P. Ausubel. *The psychology of meaningful verbal learning.* Grune & Stratton, Oxford, England, 1963.

Antoni Ballester Vallori. Meaningful learning in practice. *Journal of Education and Human Development*, 3, 01 2014. doi: 10.15640/jehd.v3n4a18.

Marc G. Bellemare, Will Dabney, and Rémi Munos. A distributional perspective on reinforcement learning. In *ICML*, 2017.

Y. Bengio, P. Frasconi, and P. Simard. The problem of learning long-term dependencies in recurrent networks. In *IEEE International Conference on Neural Networks*, pages 1183–1188 vol.3, March 1993. doi: 10.1109/ICNN.1993.298725.

B.B. Bernstein. *Pedagogy, Symbolic Control, and Identity: Theory, Research, Critique.* Class, codes, and control. Rowman & Littlefield, 2000. ISBN 9780847695768. URL `https://books.google.com.ar/books?id=_V0L-6eTYUAC`.

Peter Bhlmann and Sara van de Geer. *Statistics for High-Dimensional Data: Methods, Theory and Applications.* Springer Publishing Company, Incorporated, 1st edition, 2011. ISBN 3642201911, 9783642201912.

Weiyu Chen, Andrew S. Lan, Da Cao, Christopher G. Brinton, and Mung Chiang. Behavioral analysis at scale: Learning course prerequisite structures from learner clickstreams. In *Proceedings of the 11th International Conference on Educational Data Mining, EDM 2018, Buffalo, NY, USA, July 15-18, 2018*, 2018.

Konstantina Chrysafiadi and Maria Virvou. Review: Student modeling approaches: A literature review for the last decade. *Expert Syst. Appl.*, 40(11):4715–4729, September 2013. ISSN 0957-4174. doi: 10.1016/j.eswa.2013.02.007. URL http://dx.doi.org/10.1016/j.eswa.2013.02.007.

Wu chun Feng, Jie Tang, and Tracy Xiao Liu. Understanding dropouts in moocs. In *AAAI 2019*, 2019.

Jack Clark and Dario Amodei. Faulty reward functions in the wild, Dec 2016. URL https://openai.com/blog/faulty-reward-functions/.

Albert T. Corbett and John R. Anderson. Knowledge tracing: Modeling the acquisition of procedural knowledge. *User Modeling and User-Adapted Interaction*, 4(4):253–278, 1994. ISSN 1573-1391. doi: 10.1007/BF01099821. URL http://dx.doi.org/10.1007/BF01099821.

Dumitru Erhan, Yoshua Bengio, Aaron Courville, Pierre-Antoine Manzagol, Pascal Vincent, and Samy Bengio. Why does unsupervised pre-training help deep learning? *J. Mach. Learn. Res.*, 11:625–660, March 2010. ISSN 1532-4435. URL http://dl.acm.org/citation.cfm?id=1756006.1756025.

Mi Fei and Dit-Yan Yeung. Temporal models for predicting student dropout in massive open online courses. In *Data Mining Workshop (ICDMW), 2015 IEEE International Conference on Data Mining*, pages 256–263. IEEE, 2015.

H. Gardner. *Frames of Mind: The Theory of Multiple Intelligences*. Fontana Press, 1993. ISBN 9780006862901. URL https://books.google.com.ar/books?id=ZvJKPgAACAAJ.

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

S Halawa, D Greene, and J Mitchell. Dropout prediction in moocs using learner activity features. *Proceedings of the Second European MOOC Stakeholder Summit*, 37:7–16, 01 2014.

Linda Harasim. *Learning Theory and Online Technologies*. Routledge, New York, NY, 10001, 1st edition, 2011. ISBN 0415999766, 9780415999762.

Robert J. Harvey and Allen L. Hammer. Item response theory. *The Counseling Psychologist*, 27(3):353–383, 1999. doi: 10.1177/0011000099273004. URL http://dx.doi.org/10.1177/0011000099273004.

Peter Henderson, Riashat Islam, Philip Bachman, Joelle Pineau, Doina Precup, and David Meger. Deep reinforcement learning that matters. *Thirty-Second AAAI Conference on Artificial Intelligence (AAAI18)*, 2018.

Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural Comput.*, 9(8): 1735–1780, November 1997. ISSN 0899-7667. doi: 10.1162/neco.1997.9.8.1735. URL `http://dx.doi.org/10.1162/neco.1997.9.8.1735`.

Mohammad Khajah, Robert V. Lindsey, and Michael C. Mozer. How deep is knowledge tracing? *CoRR*, abs/1604.02416, 2016.

René F Kizilcec, Chris Piech, and Emily Schneider. Deconstructing disengagement: analyzing learner subpopulations in massive open online courses. In *Proceedings of the third international conference on learning analytics and knowledge*, pages 170–179. ACM, 2013.

Marius Kloft, Felix Stiehler, Zhilin Zheng, and Niels Pinkwart. Predicting mooc dropout over weeks using machine learning methods. In *EMNLP 2014*, 2014.

David Kolb. *Experiential Learning: Experience As The Source Of Learning And Development*, volume 1. 01 1984. ISBN 0132952610.

Andrew S. Lan, Andrew E. Waters, Christoph Studer, and Richard G. Baraniuk. Sparse factor analysis for learning and content analytics. *J. Mach. Learn. Res.*, 15(1):1959–2008, January 2014. ISSN 1532-4435. URL `http://dl.acm.org/citation.cfm?id=2627435.2670314`.

Youngju Lee and Jaeho Choi. A review of online course dropout research: Implications for practice and future research. *Educational Technology Research and Development*, 59:593–618, 10 2011. doi: 10.1007/s11423-010-9177-y.

C M. Bishop. Regularization and complexity control in feed-forward networks. *EC2 & Cie*, 1, 01 1995.

Hongzi Mao, Mohammad Alizadeh, Ishai Menache, and Srikanth Kandula. Resource management with deep reinforcement learning. In *Proceedings of the 15th ACM Workshop on Hot Topics in Networks*, HotNets '16, pages 50–56, New York, NY, USA, 2016. ACM. ISBN 978-1-4503-4661-0. doi: 10.1145/3005745.3005750. URL `http://doi.acm.org/10.1145/3005745.3005750`.

Scott W. McQuiggan, Sunyoung Lee, and James C. Lester. Early prediction of student frustration. In Ana C. R. Paiva, Rui Prada, and Rosalind W. Picard, editors, *Affective Computing and Intelligent Interaction*, pages 698–709, Berlin, Heidelberg, 2007. Springer Berlin Heidelberg. ISBN 978-3-540-74889-2.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg Corrado, and Jeffrey Dean. Distributed representations of words and phrases and their compositionality. In *Proceedings of the 26th International Conference on Neural Information Processing Systems - Volume 2*, NIPS'13, pages 3111–3119, USA, 2013. Curran Associates Inc. URL `http://dl.acm.org/citation.cfm?id=2999792.2999959`.

Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A. Rusu, Joel Veness, Marc G. Bellemare, Alex Graves, Martin Riedmiller, Andreas K. Fidjeland, Georg Ostrovski, Stig Petersen, Charles Beattie, Amir Sadik, Ioannis Antonoglou, Helen King, Dharshan Kumaran, Daan Wierstra, Shane Legg, and Demis Hassabis. Human-level control through deep reinforcement learning. *Nature*, 518(7540):529–533, February 2015. ISSN 00280836. URL http://dx.doi.org/10.1038/nature14236.

Volodymyr Mnih, Adrià Puigdomènech Badia, Mehdi Mirza, Alex Graves, Tim Harley, Timothy P. Lillicrap, David Silver, and Koray Kavukcuoglu. Asynchronous methods for deep reinforcement learning. In *Proceedings of the 33rd International Conference on International Conference on Machine Learning - Volume 48*, ICML'16, pages 1928–1937. JMLR.org, 2016. URL http://dl.acm.org/citation.cfm?id=3045390.3045594.

Usue Mori, Alexander Mendiburu, Eamonn J. Keogh, and José Antonio Lozano. Reliable early classification of time series based on discriminating the classes over time. *Data Mining and Knowledge Discovery*, 31:233–263, 2016.

Zachary A. Pardos and Lev Horodyskyj. Analysis of student behaviour in habitable worlds using continuous representation visualization. *CoRR*, abs/1710.06654, 2017. URL http://arxiv.org/abs/1710.06654.

Philip I. Pavlik, Hao Cen, and Kenneth R. Koedinger. Performance factors analysis –a new alternative to knowledge tracing. In *Proceedings of the 2009 Conference on Artificial Intelligence in Education: Building Learning Systems That Care: From Knowledge Representation to Affective Modelling*, pages 531–538, Amsterdam, The Netherlands, The Netherlands, 2009. IOS Press. ISBN 978-1-60750-028-5. URL http://dl.acm.org/citation.cfm?id=1659450.1659529.

Karl Pearson. On lines and planes of closest fit to systems of points in space. *Philosophical Magazine*, 2:559–572, 1901.

Jeffrey Pennington, Richard Socher, and Christopher D. Manning. Glove: Global vectors for word representation. In *Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543, 2014. URL http://www.aclweb.org/anthology/D14-1162.

Chris Piech, Jonathan Bassen, Jonathan Huang, Surya Ganguli, Mehran Sahami, Leonidas Guibas, and Jascha Sohl-Dickstein. Deep knowledge tracing. In *Proceedings of the 28th International Conference on Neural Information Processing Systems*, NIPS'15, pages 505–513, Cambridge, MA, USA, 2015. MIT Press. URL http://dl.acm.org/citation.cfm?id=2969239.2969296.

Siddharth Reddy, Igor Labutov, and Thorsten Joachims. Learning student and content embeddings for personalized lesson sequence recommendation. In *Proceedings of the Third*

*(2016) ACM Conference on Learning @ Scale*, L@S '16, pages 93–96, New York, NY, USA, 2016. ACM. ISBN 978-1-4503-3726-7. doi: 10.1145/2876034.2893375. URL http://doi.acm.org/10.1145/2876034.2893375.

Nils Reimers and Iryna Gurevych. Reporting Score Distributions Makes a Difference: Performance Study of LSTM-networks for Sequence Tagging. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 338–348, Copenhagen, Denmark, 09 2017. URL http://aclweb.org/anthology/D17-1035.

David E. Rumelhart, Geoffrey E. Hinton, and Ronald J. Williams. Neurocomputing: Foundations of research. chapter Learning Representations by Back-propagating Errors, pages 696–699. MIT Press, Cambridge, MA, USA, 1988. ISBN 0-262-01097-6. URL http://dl.acm.org/citation.cfm?id=65669.104451.

Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: A simple way to prevent neural networks from overfitting. *J. Mach. Learn. Res.*, 15(1):1929–1958, January 2014. ISSN 1532-4435. URL http://dl.acm.org/citation.cfm?id=2627435.2670313.

Richard S. Sutton and Andrew G. Barto. *Introduction to Reinforcement Learning*. MIT Press, Cambridge, MA, USA, 1st edition, 1998. ISBN 0262193981.

Steven Tang, Joshua C. Peterson, and Zachary A. Pardos. Deep neural networks and how they apply to sequential education data. In *Proceedings of the Third (2016) ACM Conference on Learning @ Scale*, L@S '16, pages 321–324, New York, NY, USA, 2016. ACM. ISBN 978-1-4503-3726-7. doi: 10.1145/2876034.2893444. URL http://doi.acm.org/10.1145/2876034.2893444.

Laurens van der Maaten and Geoffrey Hinton. Visualizing data using t-SNE. *Journal of Machine Learning Research*, 9:2579–2605, 2008. URL http://www.jmlr.org/papers/v9/vandermaaten08a.html.

Wei Wang, Han Yu, and Chunyan Miao. Deep model for dropout prediction in moocs. In *Proceedings of the 2Nd International Conference on Crowd Science and Engineering*, ICCSE'17, pages 26–32, New York, NY, USA, 2017. ACM. ISBN 978-1-4503-5375-5. doi: 10.1145/3126973.3126990. URL http://doi.acm.org/10.1145/3126973.3126990.

Jacob Whitehill, Kiran Mohan, Daniel Seaton, Yigal Rosen, and Dustin Tingley. Mooc dropout prediction: How to measure accuracy? In *Proceedings of the Fourth (2017) ACM Conference on Learning @ Scale*, L@S '17, pages 161–164, New York, NY, USA, 2017. ACM. ISBN 978-1-4503-4450-0. doi: 10.1145/3051457.3053974. URL http://doi.acm.org/10.1145/3051457.3053974.

Ronald J. Williams and Jing Peng. An efficient gradient-based algorithm for on-line training of recurrent network trajectories. *Neural Computation*, 2:490–501, 1990.

Kevin H. Wilson, Yan Karklin, Bojian Han, and Chaitanya Ekanadham. Back to the basics: Bayesian extensions of IRT outperform neural networks for proficiency estimation. *CoRR*, abs/1604.02336, 2016. URL `http://arxiv.org/abs/1604.02336`.

Wanli Xing, Xin Chen, Jared Stein, and Michael Marcinkowski. Temporal predication of dropouts in moocs: Reaching the low hanging fruit through stacking generalization. *Computers in Human Behavior*, 58:119–129, 2016.

Xiaolu Xiong, Siyuan Zhao, Eric G Van Inwegen, and Joseph E Beck. Going deeper with deep knowledge tracing. In *Proceedings of the 9th International Conference on Educational Data Mining (EDM 2016)*, pages 545–550, 2016.

Diyi Yang, Tanmay Sinha, David Adamson, and Carolyn Penstein Rose. "turn on, tune in, drop out": Anticipating student dropouts in massive open online courses. In *Proceedings of the 2013 NIPS Data-driven education workshop*, 12 2013.

Adams Wei Yu, Hongrae Lee, and Quoc Le. Learning to skim text. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1880–1890. Association for Computational Linguistics, 2017. doi: 10.18653/v1/P17-1172. URL `http://www.aclweb.org/anthology/P17-1172`.

Zhenpeng Zhou, Xiaocheng Li, and Richard N. Zare. Optimizing chemical reactions with deep reinforcement learning. In *ACS central science*, 2017.

# Apéndice A

# Aprendizaje por refuerzo para predicción temprana

En el capítulo 7 propusimos y evaluamos varias arquitecturas recurrentes para las tareas de predicción de abandono en dos escenarios diferentes: utilizando la secuencia completa de interacciones, que sólo es posible después de que el curso haya terminado, y prediciendo con información parcial. La principal motivación para este segundo escenario es simular un curso en desarrollo. Sin embargo, el entrenamiento con las interacciones de una semana para predecir si el estudiante volverá al curso la semana siguiente no es completamente equivalente a un escenario de predicción temprana, en el que lo ideal es que la detección del abandono se realice tan pronto como se disponga de información suficiente. Para acercarnos a la predicción temprana real, presentamos en este apéndice una propuesta basada en aprendizaje por refuerzo (RL).

El Aprendizaje por Refuerzo utiliza una función de pérdida diferente para optimizar modelos, explorando un entorno con un agente en lugar de utilizar un conjunto de datos predefinidos de ejemplos de entrenamiento. Compararemos este método con los modelos que hemos utilizado en los experimentos realizados, y presentaremos las principales razones por las que esto podría captar el proceso de aprendizaje del estudiante de una manera más adecuada. Definimos los diferentes conceptos que intervienen en el LR para el caso de la predicción de la deserción escolar, incluyendo la forma en que el estudiante y el elemento del curso pueden ser modelados bajo tal enfoque. Por último, también evaluamos los posibles inconvenientes o dificultades al tratar de entrenar a dicho agente para la predicción de la deserción en general y al utilizar el conjunto de datos dado en particular.

## A.1 Diferentes tipos de el aprendizaje automático

Los enfoques para el aprendizaje automático puede dividirse en varios grandes grupos de algoritmos:

El *Aprendizaje supervisado*, que incluye todos los enfoques tratados en los capítulos anteriores, es el aprendizaje de un conjunto a partir de ejemplos etiquetados proporcionados por una fuente externa. Cada instancia de entrenamiento está compuesta por un ejemplo del fenómeno que nos interesa, junto con una o más etiquetas asociados al ejemplo. El objetivo de este tipo de aprendizaje es que el sistema aprenda a predecir las etiquetas de las instancias de entrenamiento y posteriormente extrapolar, o generalizar, sus predicciones a ejemplos no presentes en el conjunto de entrenamiento. Esto se logra a través de la minimización estocástica de una función de pérdida o coste, que representa el error de clasificación.

Por el contrario, en es escenario de *Aprendizaje no supervisado* no hay etiquetas que se puedan predecir. En su lugar, el objetivo es encontrar la estructura subyacente en las colecciones de ejemplos no etiquetados, por ejemplo para agrupar las instancias de entrenamiento en grupos. El *aprendizaje semisupervisado* intenta combinar, usando diferentes técnicas, ejemplos etiquetados y no etiquetados para el modelado de una tarea supervisada.

El *Aprendizaje de refuerzo* (RL) es otra categoría que difiere de todas las anteriores en el sentido de que no depende de los ejemplos preestablecidos (etiquetados o no). En cambio, un agente (lo que antes llamábamos modelo) interactúa con un entorno definido y con el objetivo de encontrar el conjunto óptimo de acciones (denominado política $\pi$) que el agente debe tomar para maximizar una función dde recompensa dada.

RL tiene como objetivo resolver problemas de toma de decisiones que pueden ser simplificados y modelados como Procesos de Decisión de Markov (MDP), o en otras palabras, procesos estocásticos discretos. Las señales de recompensa pueden ser inmediatas o a largo plazo y el agente tiene que hacer un balance entre las dos para obtener la máxima recompensa posible. A diferencia del aprendizaje automático supervisado y no supervisado, no existe un conjunto de ejemplos construido previamente. Los datos se recogen permitiendo nuevas interacciones del agente con el entorno, y se utilizan posteriormente para optimizar la política actual en función de las recompensas obtenidas.

Los algoritmos de RL tienen su origen en los años 50, y encontramos una revisión de la historia temprana de estos métodos en el trabajo de Sutton and Barto [1998]. Más allá de la aplicación inmediata de juegos como el ajedrez o el tic-tac, donde el agente tiene el objetivo de ganar la partida, hay muchas aplicaciones de RL en el mundo real. Por ejemplo, la planificación o la asignación de recursos Mao et al. [2016], la optimización de las reacciones químicas Zhou et al. [2017], el control operacional de la maquinaria en las industrias, la planificación de rutas

o tareas. En los últimos años, RL ha tomado una renovada importancia después de haber demostrado ser excepcionalmente bueno para jugar a los videojuegos Atari Mnih et al. [2015], y se espera que sean la base en el desarrollo de vehículos autónomos.

## A.2 ¿Por qué usar aprendizaje por refuerzo?

En el escenario de predicción de abandono, tenemos acceso a un conjunto de datos supervisados con el verdadero abandono de un estudiante, que sólo se determina al final del curso. Sin embargo, no poseemos información sobre cuándo el estudiante muestra un comportamiento indicativo de abandono, aparte de no volver a interactuar con la plataforma. En ese sentido, no tenemos la información necesaria para decidir cuándo es el momento óptimo para tomar medidas preventivas que puedan evitar el futuro evento de abandono. Esta tarea, introducida anteriormente como detección temprana, consiste en detectar el abandono escolar lo antes posible, con la mínima pérdida de precisión en la predicción.

Proponemos utilizar un algoritmo RL para estimar el punto óptimo de intervención para evitar la deserción. El aprendizaje por refuerzo es una técnica ideal para explorar escenarios en los que no se conoce toda la información, pero aún así hay una señal de éxito o fracaso durante el entrenamiento. En nuestro caso, esta señal es el regreso del estudiante a la plataforma de aprendizaje.

En este enfoque, no tenemos una plataforma real que simule el entorno con el que el agente interactuaría. Sin embargo, podemos utilizar el conjunto de datos recogidos para simular un conjunto de "episodios" que el modelo necesita para "jugar", mostrando sólo una interacción a la vez. En la siguiente sección, describiremos en detalle un posible modelado formal del problema como una tarea RL.

### A.2.1 Modelación formal

Para modelar un problema con un enfoque de RL, es necesario entenderlo como un MDP finito. En otras palabras, es necesario definir los siguientes términos:

Un **agente** que aprende a tomar acciones. Para el escenario de predicción de abandono, proponemos usar una red neural recurrente para elegir entre tomar una acción preventiva o no. El estado del agente es, como en los modelos anteriores, el estado oculto de la RNN, que representa el estado del estudiante. Este enfoque entra en la categoría de los modelos basados en políticas, donde el agente contiene la política sobre sí mismo, y donde el objetivo es optimizar directamente el conjunto de pesos de la red.

Un **ambiente** donde el agente puede ganar nueva experiencia y recibir las señales de recompensa. En nuestro caso, el entorno simula el funcionamiento en línea de un MOOC. Hay un episodio por cada secuencia de entrenamiento de las interacciones. Cada paso en una simulación es una única interacción de un estudiante, y el episodio se completa cuando no hay más interacciones o el agente decide hacer una intervención para prevenir el abandono. Bajo estas definiciones, las **observaciones** son las acciones que el alumno realiza. Pueden ser vistas como las respuestas del ambiente a la acción de nuestro agente.

Una vez que al agente se le presenta el siguiente paso o interacción, selecciona una **acción** para realizar. En este caso, las dos únicas acciones posibles son lanzar una advertencia de abandono, o continuar sin intervención. Después de elegir la acción, el entorno devuelve una señal de recompensa al agente. Si la acción era intervenir, el episodio se detiene. Por el contrario, si la acción es no intervenir, el entorno devuelve al agente la siguiente interacción o el indicador de fin de secuencia si no hay más interacciones.

La **señal de recompensa** indica el éxito de la acción del agente en el paso dado. En nuestro entorno, proponemos que el agente obtenga una alta recompensa si interviene cuando hay una verdadera deserción, y una mayor penalización cuando el episodio termina en una deserción y el agente no interviene. También hay una alta recompensa negativa en los casos en que hay una intervención pero no hay abandono. Esto evitará que el sistema emita la advertencia a cada estudiante, y lo obligará a esperar hasta que el abandono sea seguro. Como nos acercamos a la tarea con un enfoque en la detección temprana y la simulación de un escenario realista, el agente no debe saber si ocurrirá un abandono en el futuro. En los casos en los que no hay intervención y hay una acción posterior del estudiante, la recompensa es una pequeña cantidad fija para animar al agente a continuar el episodio durante muchos pasos. Por ejemplo, las recompensas que se muestran en la tabla A.1 se ajustan a los criterios descriptos:

Table A.1: Ejemplo de recompensa para el problema de predicción de deserción temprana

| | Intervenir | No intervenir |
|---|---|---|
| Episodio termina (abandono) | +50 | -50 |
| Episodio continúa | -25 | +5 |

Habiendo descrito los conceptos a un alto nivel, podemos continuar describiendo nuestra propuesta a un nivel más detallado. El objetivo del algoritmo RL es encontrar una política óptima $\pi^*$ que maximice el **rendimiento total descontado** definido como:

$$R = \sum_{t=0}^{T-1} \gamma^t r_t(s_t, a_t) \tag{A.1}$$

donde el índice $t$ indica el paso temporal, $T-1$ es la duración del episodio, $r_t$, $a_t$ y $s_t$ son la recompensa, la acción y el estado en el tiempo $t$, respectivamente. Es importante notar que, aunque todos los episodios tienen un final en el tiempo $T$, podría tener cualquier número arbitrario de pasos y el agente no sabe esta cantidad de antemano. Por lo tanto, esencialmente estamos en un escenario de horizonte infinito.

Para fomentar la predicción temprana, es decir, que el agente actúe con la menor cantidad de datos posible, usaremos el **factor de descuento** *gamma*. Este factor es un número entre 0 y 1, y controla la importancia relativa de las recompensas obtenidas a corto plazo con respecto a las recompensas a largo plazo. Si está más cerca de 1, todas las recompensas tienen el mismo peso, y si está más cerca de 0, se ignoran las recompensas a largo plazo. Independientemente de nuestro modelado del problema, se recomienda utilizar un factor de descuento para reducir la varianza.

Para fomentar la predicción temprana, es necesario dar prioridad a la recompensa a largo plazo. De lo contrario, el clasificador sólo emitirá una señal de abandono cuando esté cerca del final de la secuencia. Esto significa que el factor de descuento $\gamma$ debe ser cercano a uno, aunque el valor exacto debe definirse empíricamente, en función de los demás valores que intervienen en la función de recompensa.

## A.3 Algoritmo Actor-Critic

El concepto *actor-critic* es mejor pensado como un marco de trabajo o una clase de algoritmos que satisfagan los criterios de que existen actores parametrizados y críticos. El actor es la política $\pi_\theta(a|s)$ con parámetros $\theta$ que lleva a cabo acciones en un entorno. El crítico calcula las funciones de valor para ayudar al actor a aprender. Estas son generalmente el valor del estado, valor de acción de estado, o valor de ventaja, denotado como $V_\pi(s)$, $Q_\pi(s,a)$, y $A_\pi(s,a)$, respectivamente. Por el contrario, otros algoritmos como Q-learning no se consideran críticos actor-critic, ya que el crítico (Q-network) es suficiente para determinar la política. Simplemente aplicando una función softmax a la función de valor de estado-acción y eligiendo la acción que maximiza la recompensa esperada.

Como se describe en Sutton and Barto [1998], la función de valor de un estado $V_{pi}(s)$ en virtud de una política *pi* estima el valor total de la recompensa que un agente puede esperar acumular en el futuro, a partir de estado dado. Mientras que las recompensas determinan la conveniencia inmediata e intrínseca de los estados del ambiente, los valores indican la conveniencia a largo plazo de los estados después de tener en cuenta los estados que probablemente seguirán, y las recompensas disponibles en esos estados.

El valor de estado-acción de un estado $Q_\pi(s,a)$, también conocido comúnmente como función

Q, es en esencia similar a la función de valor. Estima la recompensa esperada del agente del estado $s$ si la acción $a$ es tomada, bajo la política $\pi$.

El valor de ventaja $A_\pi(s, a)$ es una estimación de la ganacia de tomar la acción $a$ del estado $s$ con respecto a la política actual o a una línea base determinada. Esta cantidad se utiliza para ayudar al modelo a encontrar una política óptima, en lugar de sólo una buena política. Desarrollaremos este concepto con más detalle a lo largo de la sección.

Los algoritmos actor-critic se consideran basados en políticas. La principal diferencia entre los métodos basados en políticas y los basados en valores es que los primeros tienen características de convergencia más fuertes. Los segundos suelen divergir cuando se utiliza la aproximación de funciones, ya que optimizan el espacio de valores y un ligero cambio en la estimación de los mismos puede dar lugar a un gran cambio en el espacio de políticas.

El algoritmo Asynchronous Advantage Actor Critic (A3C), desarrollado por Mnih et al. [2016], y su versión síncrona A2C, se basan en el cálculo del valor de ventaja:

$$A_\pi(s_t, a_t, \theta, \theta_v) = Q_\pi(s_t, a_t, \theta) - V_\pi(s_t, \theta_v) \tag{A.2}$$

donde $\theta$ y $\theta_v$ son los parámetros de la función Q y V, respectivamente.

La función de ventaja es fundamental para reducir la varianza durante el entrenamiento del clasificador. Estamos calculando cuánto mejor es nuestro modelo actual ($Q_\pi(s_t, a_t, \theta)$) con respecto al valor esperado obtenido en el estado dado $V_\pi(s_t, \theta_v)$. Sin esta función, todas las secuencias reproducidas durante las simulaciones que tengan una recompensa positiva, aunque sean pequeñas, tendrán sus probabilidades aumentadas en el modelo. Esto hace que el modelo se incline hacia secuencias no óptimas. Con la función de ventaja, el proceso de optimización sólo aumenta la probabilidad de las secuencias que introducen alguna mejora en el rendimiento.

### Uso de embeddings en el agente

El agente que proponemos para este problema es una red neuronal recurrente, donde las entradas son las observaciones, y las salidas son las dos acciones posibles. El **estado** del agente es el estado oculto de la capa recurrente. Este estado cumple con la propiedad Markoviana necesaria, que establece que toda la información que describe el entorno debe ser codificada en el valor del estado y no depender de valores externos. El estado, junto con la observación, es utilizado por la política para generar una nueva acción.

Existe un claro paralelismo entre esta descripción y el modelo propuesto para la predicción general de la deserción escolar. Podemos interpretar el estado del agente como el estado actual

del estudiante, e inicialmente proyectamos la observación en una densa incrustación para cada elemento del curso. También está claro que los embeddings conjuntas pueden ser replicadas en este nuevo escenario.

El uso de RL no cambia la forma de la red, sino el proceso de optimización. Responde a la necesidad de una función de pérdida más adecuada, dado que la señal de abandono presente en el conjunto de datos no puede ser utilizada con éxito para entrenar un modelo de buen rendimiento.

### Procesos de exploración y optimización

El modelo A2C utiliza dos modelos paralelos: un modelo de muestreo y un modelo de entrenamiento. Ambos se construyen a partir de la política, y sólo divergen en el tamaño del lote, lo que significa que comparten todos los parámetros. El modelo de muestreo siempre tiene un tamaño de lote igual al número de entornos ejecutados en paralelo. Este modelo se utiliza para muestrear la siguiente acción tomada por la política, el valor y el estado en cada paso. La nueva acción se envía al ambiente y se obtiene una nueva observación y recompensa. Basándose en la recompensa real $r$, la recompensa esperada se calcula usando el factor de descuento gamma como se ve en la ecuación A.1. Sin embargo, si el episodio no ha terminado en el tiempo $T$, la recompensa del último paso se aproxima usando la función de valor. Este ciclo se repite para los pasos dados, o hasta que se alcanza un estado final.

Una vez que el modelo de muestreo se ha utilizado para calcular las recompensas descontadas $R$ y las funciones de valor, el algoritmo A2C estima la ventaja utilizando la ecuación A.2. Todos estos valores se alimentan al modelo de tren entrenamiento y se utilizan para calcular y aplicar los gradientes de la red de políticas y de la red de valores. Por esta razón, el modelo de entrenamiento toma un lote de tamaño igual al número de entornos por el número máximo de pasos. La función de pérdida para el modelo de política, para un lote de ejemplos es:

$$loss_\pi = \frac{1}{BS}\sum_{i=0}^{BS}\sum_{t=0}^{T_i} log\pi(a_t|s_t;\theta')A_\pi(s_t,a_t;\theta,\theta_v) \tag{A.3}$$

$$= \frac{1}{BS}\sum_{i=0}^{BS}\sum_{t=0}^{T_i} log\pi(a_t|s_t;\theta')(R(s_t,a_t;\theta') - V_\pi(s_t;\theta_v)) \tag{A.4}$$

$$= \frac{1}{BS}\sum_{i=0}^{BS}\sum_{t=0}^{T_i} log\pi(a_t|s_t;\theta')((\sum_{j=0}^{T_i-t-1}\gamma^t r_{t+j}) + \gamma^{T_i}V_\pi(s_{T_i};\theta_v) - V_\pi(s_t;\theta_v))$$

donde $BS$ es el tamaño del lote, $T_i$ es el número de pasos dados hasta que el episodio termina o el número máximo de pasos alcanzados. $theta'$ son los parámetros del modelo de pasos, y

*theta'* son los parámetros del modelo de entrenamiento. Sin embargo, en un modelo asíncrono esta diferencia puede ser ignorada, ya que ambos modelos se actualizan al mismo tiempo.

De la ecuación A.3 podemos ver que los gradientes de los diferentes pasos de tiempo se están acumulando, lo que según los autores originales tiene el mismo impacto que el entrenamiento en minibatches. Además, los cálculos necesarios para calcular la nueva observación y la recompensa del paso son muy sencillos. Por estas razones, utilizamos un solo entorno, ya que paralelizar varios lotes no mejorará el rendimiento.

La función de pérdida $loss_V$ para la aproximación de la función de valor es sólo el error cuadrado medio entre el valor real de la recompensa descontada y el valor predicho:

$$loss_V = \frac{1}{BS}\sum_{i=0}^{BS}(R(s_t, a_t; \theta') - V_\pi(s_t; \theta_v))^2 \tag{A.5}$$

Finalmente, la función de pérdida también incluye la entropía del modelo ponderada por un coeficiente $\beta$. La entropía se utiliza como término de regularización, ya que se ha comprobado que mejora la exploración al desalentar la convergencia prematura hacia políticas deterministas. La pérdida completa del modelo es:

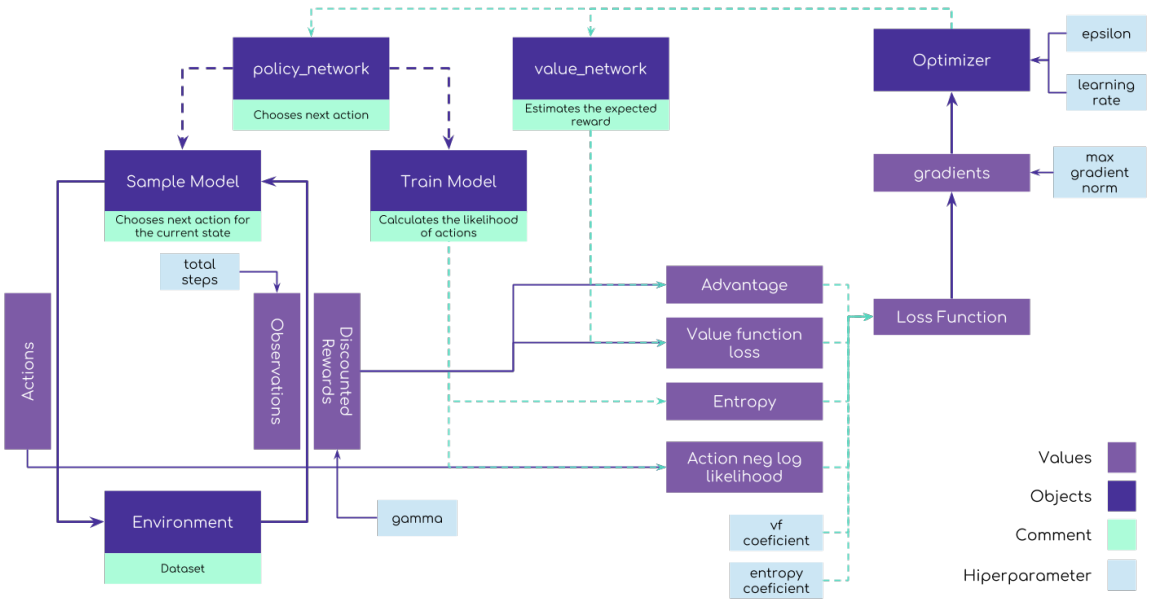
$$loss = loss_\pi + loss_V + \beta\frac{1}{BS}\sum_{i=0}^{BS}\sum_{t=0}^{T_i} H(\pi(s_t); \theta') \tag{A.6}$$

En la siguiente figura mostramos la organización general del algoritmo y cómo se relacionan las partes:

## A.4 Desventajas de usar DRL

Entrenar un algoritmo DRL no es tan simple como entrenar un modelo supervisado. Arulkumaran et al. [2017] menciona algunas de las dificultades que se han presentado durante el proceso de diseño y optimización. La única señal disponible para el agente es la recompensa, y la exploración está fuertemente correlacionada por las acciones tomadas por el agente. Este intercambio suele dar lugar a una sobreexplotación de los mínimos locales, situaciones en las que el agente obtiene alguna recompensa pero que claramente no son lo que se esperaba. No existe una receta óptima para equilibrar la relación exploración/explotación a fin de asegurar la cobertura de las partes pertinentes del espacio, debe aprenderse empíricamente.

Figure A.1: Esquema del algoritmo de optimización de a2c.




No sólo la selección de los episodios es empírica, sino que el éxito del modelo depende de la adecuación de la función de recompensa al problema, ya que debe fomentar exactamente el comportamiento deseado. La literatura actual se centra en soluciones notables para la aplicación de la LR, pero se habla poco de las que son defectuosas. Un claro ejemplo es el blog de Clark and Amodei [2016]. Los autores muestran, en detalle, las dificultades que enfrentaron al tratar de optimizar un algoritmo de funcionamiento de un barco, originado por un mal diseño de la función de recompensa.

La otra posible preocupación de la aplicación de RL es el tamaño del conjunto de datos. No tenemos una estimación del número de interacciones necesarias para obtener un modelo de buen funcionamiento para este problema. Otras aplicaciones, como aprender a jugar videojuegos, requieren un número muy grande de interacciones para ser optimizadas, del orden de miles de millones de [Bellemare et al., 2017]. La búsqueda de hiperparámetros es también más delicada que en los escenarios de aprendizaje supervisado, en particular los factores de escala aplicados a la función de recompensa Henderson et al. [2018]. Adicionalmente, si se optimizara un modelo para este conjunto de datos en particular, no hay garantías de que se generalizara a otros escenarios sin un reentrenamiento completo.

## A.5 Discusión

Para finalizar este trabajo, en este apéndice hemos presentado una propuesta teórica de cómo se podría abordar el problema de la predicción de la deserción con el aprendizaje por refuerzo, en particular utilizando el algoritmo A2C. Con esta arquitectura, el modelado de embeddings

descripto en el capítulo 4 sigue siendo aplicable, y también podrían evaluarse los beneficios de una representación conjunta. Sin embargo, debido a las limitaciones de tiempo, no hemos implementado y aplicado el algoritmo al problema.