

DEPARTAMENTO DE INTELIGENCIA ARTIFICIAL
ESCUELA TÉCNICA SUPERIOR DE INGENIEROS INFORMÁTICOS

MODELADO DE COMPORTAMIENTO DE CONDUCTORES CON TÉCNICAS DE INTELIGENCIA COMPUTACIONAL

TESIS DOCTORAL

Alberto Díaz Álvarez
Máster en Ciencias y Tecnologías de la Computación

DIRECCIÓN

Dr. Francisco Serradilla García
Doctor en Inteligencia Artificial
Dr. Felipe Jiménez Alonso
Doctor en Ingeniería Mecánica

hoja según tribunal

Alberto Díaz Álvarez

Modelado de comportamiento de conductores con técnicas de Inteligencia Computacional

Tesis doctoral, 9 de febrero de 2018

Revisores: Rev1, Rev2 y Rev3

Dirección: Dr. Francisco Serradilla García, Dr. Felipe Jiménez Alonso

Instituto Universitario de Investigación del Automóvil

Universidad Politécnica de Madrid

Campus Sur UPM, Carretera de Valencia (A-3), km7
28031 Madrid

This work is licensed under a Creative
Commons “Attribution-NonCommercial-
ShareAlike 3.0 Unported” license.



De momento nada de dedicatorias, mejor TODOs:

- *Cambiar el glosario para que esté en español.*

*Si pasamos la tesis a inglés, pasamos también
los términos del glosario.*

Resumen

Aquí el abstract en español

Abstract

Aquí el abstract en inglés

Índice general

Introducción 19

I Estado de la cuestión 25

Inteligencia Computacional 27

Simulación de tráfico 47

Modelos de comportamiento 61

II Modelado de conductores: definición y diseño 73

Definición del problema 75

Modelo de comportamiento de conductor 77

Ejecución del modelo en entornos de simulación 89

III Resultados y conclusiones 91

Resultados 93

Conclusiones 95

Sistemas desarrollados 97

Índice de figuras

1. Relación entre la emisión de CO_2 a la atmósfera y el aumento de la temperatura en el planeta los últimos 1000 años 21
2. Censo de conductores según género y edad 23
3. Diferentes objetivos perseguidos por la Inteligencia Artificial 30
4. Ilustración de una sección del neocórtex humano 32
5. Modelo de neurona artificial de McCulloch y Pitts 32
6. Three training epochs on a multilayer perceptron with a 0.5 dropout rate (i.e. 50 % probability for a neuron to be disabled) in its hidden layer. The grey neurons are the ones disabled each epoch. 34
7. *Modus ponendo ponens* vs. *modus tollendo tollens* 38
8. Diferencias entre *modus ponens* tradicional y generalizado 38
9. Diagrama general de un Sistema de Inferencia Difusa 39
10. Esquema de agente y sus propiedades 41
11. Arquitectura básica de un agente 43
12. Diferencias entre un agente sin y con modelo de entorno 43
13. Arquitecturas de agente según su comportamiento 44
14. Diferencias entre colaboración y competitividad de agentes 46
15. Aspectos medibles del tráfico 48
16. Clasificación de simuladores según granularidad 49
17. Alternativas a la clasificación por granularidad 50
18. Ejemplo de simulador basado en Autómata Celular 50
19. Ejemplo de modelo lineal en un espacio continuo 51
20. Ejemplo de efecto de ondas de choque en simulación de tipo Nagel-Schreckenberg 52

21. Simulación de comportamiento en intersección basada en un MAS 53
22. Captura de pantalla del simulador MovSim 54
23. Características obligatorias y deseables del simulador a elegir 55
24. Captura de pantalla del simulador SUMO 57
25. Ejemplo de forma de envío de mensajes a través de TraCI 58
26. Arquitectura de la plataforma TraaS 59
27. Los tres niveles jerárquicos de conducción según [Michon, 1985] 61
28. Diferentes modelos de aceleración 63
29. Operaciones involucradas en el proceso de cambio de carril 63
30. Nomenclaturas a usar en los modelos de conducción 64
31. Evolución de los tres tipos generales de modelo de *car-following* 65
32. Ejemplo de cambio de carril obligatorio frente a cambio de carril discrecional 66
33. Efecto de la distancia en el tipo de cambio de carril según el modelo de [Gipps, 1986] 67
34. Estructura del modelo de comportamiento propuesto por [Toledo et al., 2007]
35. Principales áreas de aplicación de la Inteligencia Computacional en los Sistema Inteligente de Transporte (ITS, Intelligent Transport System) 69
36. La inexactitud se tiene en cuenta en los modelos de la Inteligencia Computacional 70
37. Ejemplo de aproximación *neuro-fuzzy* al control de señales de tráfico 71
38. The instrumented Mitsubishi iMiEV. 79
39. The instrumented vehicle schema. 81
40. Comparación de indicadores entre los diferentes perfiles de conducción. 82
41. Las dos rutas del experimento, (a) Ruta R_1 para entrenamiento y (b) Ruta R_2 para validación. 82
42. Given the row of data in time t , we define the moment T_i as the row of data that occurred in time $t - \frac{i}{10}$ seconds. 83
43. Un ejemplo de mapa de profundidad capturado en una de las pruebas. La celda es más azul cuanto más cercano está el obstáculo detectado. 88

Índice de cuadros

1. Tabla comparativa de los simuladores seleccionados 57
2. Resúmen de información extraída del vehículo instrumentado 80
3. Indicadores de los datos extraídos de cada perfil 81
4. Resúmen de información extraída del vehículo instrumentado 84

Introducción

La **Inteligencia Artificial (AI, Artificial Intelligence)** como área de conocimiento ha experimentado un creciente interés en los últimos años. Esto no siempre ha sido así, ya que tras un nacimiento muy esperanzador, con mucho optimismo, le siguieron unas épocas de apenas avance. Sin embargo, en la actualidad es muy difícil encontrar un campo que no se beneficie directamente de sus técnicas.

Una de sus razones es su carácter multidisciplinar ya que, aunque pertenece al campo de la informática, es transversal a muchos otros de naturaleza en principio muy diferente, como pueden ser por ejemplo la biología, neurología o la psicología.

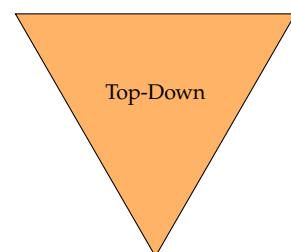
Dentro del área de la **AI** es común diferenciar dos tipos de aproximaciones a la hora de representar el conocimiento: el enfoque **clásico**, que postula que el conocimiento como tal se puede reducir a un conjunto de símbolos con operadores para su manipulación, y el enfoque de la **Inteligencia Computacional (CI, Computational Intelligence)**, que defiende que el conocimiento se alcanza a través del aprendizaje, y que basa sus esfuerzos en la simulación de elementos de bajo nivel esperando que el conocimiento “emerja” de la interacción de éstos.

El límite entre ambos conjuntos no está perfectamente definido, más aún si tenemos en cuenta las diferentes terminologías existentes, las sinergias entre distintas técnicas dentro del área y los diferentes puntos de vista sobre éstas por parte de los autores. Sin embargo, una de las principales diferencias de ambos paradigmas es el punto de vista a la hora de solucionar problemas, siendo la aproximación **top-down** la usada en problemas de **AI** clásica y la **bottom-up** la típica usada en la **CI**. Revisaremos las diferencias entre conceptos de diferentes autores en el capítulo **Inteligencia Computacional**.

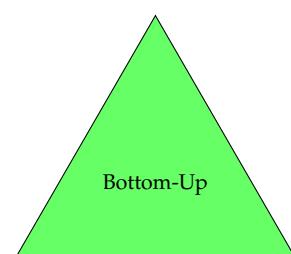
1

Uno de los campos de aplicación es el de los **ITSs**. Éstos se definen como un conjunto de aplicaciones orientadas a gestionar el transporte en todos sus aspectos (e.g. conducción eficiente, diseño de automóviles, gestión del tráfico o señalización en redes de carreteras) para hacerlos más eficientes y seguros. El interés es tal que en el año 2010 se publicó la directiva 2010/40/UE (ver [par, 2010]). En ella, los **ITSs** se definieron como *aplicaciones avanzadas que, sin incluir la inteligencia*

¹ **TODO!** Replantear esto (y quizás hacer una figura más interesante). También quizás estaría mejor mover esta figura a la siguiente página y poner en esta, o bien nada o bien alguna figura más llamativa.



Una aproximación *top-down* a los problemas funciona definiendo primero el algoritmos que resuelve el problema para posteriormente ejecutarlo y llegar así a la solución exacta.



Por otro lado, una aproximación *bottom-up* el algoritmo de resolución no se programa, sino que se aprende, llevando él sólo a soluciones no necesariamente exactas pero sí lo suficientemente buenas para ser aceptadas.

como tal, proporcionan servicios innovadores en relación con los diferentes modos de transporte y la gestión del tráfico y permiten a los distintos usuarios estar mejor informados y hacer un uso más seguro, más coordinado y «más inteligente» de las redes de transporte. Junto con dicha definición, se estableció el marco de implantación de los éstos en la Unión Europea.

La conducción es una tarea muy compleja que involucra la ejecución de muchas tareas cognitivas pertenecientes a diversos niveles de abstracción. El concepto del tráfico puede verse como un sistema complejo que emerge de las interacciones de agentes muy diversos, incluyendo a aquellos que realizan la tareas de conducir. El comportamiento durante la tarea de conducción es un objeto interesante de estudio: la evaluación de los conductores para conocer su manera de actuar en determinados escenarios nos permite, por ejemplo, detectar qué factores pueden afectar más o menos sobre determinados indicadores (e.g. el consumo estimado para una ruta en concreto). Sin embargo, la evaluación en algunos casos puede no ser posible debido a limitaciones como, por ejemplo, el tiempo, el dinero o la peligrosidad del escenario.

Los simuladores de tráfico son una solución para muchas de estas limitaciones, pero suelen basar su funcionamiento en conductores y vehículos (normalmente concebidos como una única entidad) basándose en modelos de conductor que responden a funciones más o menos complejas, además con pocas o ninguna opciones de personalización. Esto provoca que dichos modelos se adapten poco al modelo de un conductor en concreto.

Esta tesis pretende explotar la generación de modelos de conductor para simuladores que respondan al comportamiento de conductores reales usando, para ello, técnicas pertenecientes al campo de la [CI](#).

Concretamente pretende desarrollar un método para el análisis de la eficiencia de los conductores realizando, para ello, un modelo del perfil de conducción a partir de técnicas de la [CI](#) y aplicándolo a un entorno de simulación basado en [Sistemas Multiagente \(MASs, Multi-Agent Systems\)](#). Así, una vez configurado el entorno, se podrá estudiar aspectos generales como la evolución del tráfico con determinados perfiles o particulares como el estilo de conducción o el impacto de los sistemas de asistencia.

Motivación

Los conceptos introducidos al comienzo del capítulo obedecen a una *necesidad* de la sociedad en la que vivimos, y que afecta tanto a nuestra generación como a las venideras: la eficiencia en el transporte. Dado que es imprescindible saber que existe un problema para arreglarlo, nada mejor que puntualizar algunos hechos de sobra conocidos:

- En el año 2014, el número de vehículos a nivel mundial superó los 1,200 millones, con una tendencia creciente [OICA, 2015]. Reducir en un pequeño porcentaje el consumo evita la emisión de toneladas de gases considerados nocivos para el medio ambiente y el ser humano ².
- Aunque existen diferentes puntos de vista acerca de cuándo se agotarán las reservas de petróleo, los combustibles fósiles son recursos **finitos**. Lo más probable es que no se llegue a agotar debido a la ley de la oferta y la demanda, pero hay que recordar que el petróleo se usa como base para la producción de otros muchos tipos de productos, como por ejemplo la vaselina, el asfalto o los plásticos.
- La emisión de gases está correlacionada con el aumento de la temperatura del planeta, hecho que se ilustra en la figura 1. De seguir con el ritmo de consumo actual, se teme llegar a un punto de no retorno con consecuencias catastróficas para la vida en el planeta.
- La conducción eficiente afecta directamente a factores correlacionados con el número de accidentes de tráfico. Un factor de sobra conocido es el de la velocidad, factor relacionado no sólo con el número sino con la gravedad de los accidentes ([Imprialou et al., 2016]). Otros indicadores son las aceleraciones, deceleraciones y maniobras de cambio de dirección, cuyas frecuencias son directamente proporcionales a la agresividad, falta de seguridad y accidentes e inversamente proporcionales a la eficiencia ([Dingus et al., 2006, Lerner et al., 2010]).

Estos son sólo algunos hechos que ponen de manifiesto la necesidad de centrarse en el problema de cómo hacer de la conducción una actividad más eficiente y segura. Por ello, la **conducción eficiente** o *eco-driving* se define como la aplicación de una serie de reglas de conducción con el objetivo de reducir el consumo de combustible, independientemente del tipo (e.g. electricidad, gasolina, gas natural, ...).

Si es posible discriminar entre conductores eficientes y no eficientes se pueden identificar los hábitos recurrentes en estos últimos y adecuar la formación para eliminar dichos hábitos. Más aún teniendo en cuenta la relación existente entre la peligrosidad y algunas conductas agresivas. Un ejemplo donde la identificación de perfiles no eficientes pueden tener impacto claro económico y social es el de las empresas cuya actividad se basa en el transporte de mercancías o de personas.

Sin embargo, identificar la conducta de un conductor no es fácil, dado que su comportamiento se ve condicionado por numerosos factores como el estado de la ruta, el del tráfico o el estado físico o anímico. Además, la ambigüedad de las situaciones dificulta todavía

² Uno puede argumentar que el parque automovilístico se recicla con nuevos vehículos eléctricos categorizados “de consumo o”. La triste realidad es que estos vehículos consumen la electricidad generada actualmente de una mayoría de centrales de combustibles fósiles y nucleares. Además, mientras que en países desarrollados el crecimiento ha sido en torno al 4-7%, en países subdesarrollados, donde no existe aún infraestructura para la recarga de vehículos eléctricos, dicho crecimiento ha superado el 120%.

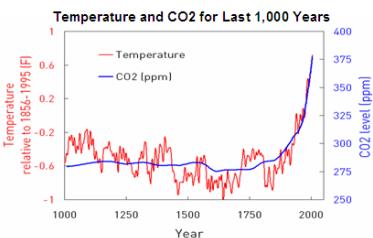


Figura 1: Desde el comienzo de la revolución industrial, el uso masivo de combustibles fósiles y el crecimiento de la población propició un aumento desproporcionado de CO₂ a la atmósfera, tendencia que sigue en aumento aún con la (lenta) adopción del vehículo eléctrico. La gráfica muestra cómo ambos valores parecen estar correlacionados. Fuente: Environmental Defense Fund (edf.org).

más la identificación. Por ejemplo, un conductor puede ser clasificado en un momento como agresivo o no eficiente en una situación, únicamente porque su comportamiento ha sido condicionado por las malas reacciones de otros conductores conductores.

El análisis de todos los posibles casos es una tarea prácticamente imposible. Por ello, las simulaciones pueden dar una estimación de los posibles resultados de un estudio en el mundo real. Las simulaciones con **MASs** representan a los conductores como agentes independientes, permitiendo la evaluación del comportamiento tanto individual como general del sistema en base a sus individuos a través de iteraciones discretas de tiempo.

Si el comportamiento de dichos agentes es extraído a partir de los datos reales de conductores, su comportamiento dentro de la simulación podría ser considerado como fuente de datos para condiciones de tráfico y/o rutas no contempladas en el mundo real. De esta forma, se dispondría de un marco de trabajo para la comparación de diferentes conductores sin necesidad de exponerlos a todos y cada uno de los posibles eventos posibles. También sería factible evaluar sistemas de asistencia evitando los problemas de no comparabilidad de condiciones del entorno entre pruebas.

Demostrar que la evaluación de un modelo del conductor en entornos simulados es equivalente a la evaluación de conductores en entornos reales implica que se pueden comparar dos conductores usando un criterio objetivo, es decir, sin depender del estado del resto de factores a la hora de realizar la prueba de campo. Dicho de otro modo, implicaría que es posible comparar la eficiencia de dos conductores independientemente del estado del tráfico e, incluso, sobre rutas diferentes.

Objetivos

El objetivo de esta tesis doctoral es la de demostrar la hipótesis 1, quedando dicha demostración dentro de los límites impuestos por los supuestos y restricciones indicados más adelante.

Hipótesis 1 (H1): *La aplicación de técnicas pertenecientes al campo de la CI con datos extraídos de un entorno de microsimulación de espacio continuo y tiempo discreto basado en sistemas multiagentes permitirá modelar, de manera fiel a la realidad, el comportamiento de conductores reales.*

Por tanto, el objetivo de la tesis es el de simular el comportamiento de conductores en entornos de micro-simulación a partir de su comportamiento en entornos reales usando técnicas de **CI**. Para ello se consideran los siguientes objetivos específicos:

- Estudiar y aplicar técnicas de la **CI** sobre el área de la conducción.

Los **Estudio Naturalista de Conducción (NDS, Naturalistic Driving Study)** basan su funcionamiento en la captura masiva de datos de conducción, normalmente involucrando una gran cantidad de sensores, para analizar el comportamiento del conductor, las características del vehículo, la vía, etcétera. La cantidad de sensores y la velocidad de captura hacen que la tarea de analizar y extraer conclusiones sea una tarea prácticamente imposible para un humano, por lo que es necesario el uso de técnicas de análisis de datos que suelen recaer en los campos de la estadística y del aprendizaje automático.

- Realizar un Estudio Naturalista de Conducción sobre conductores reales para:
 1. Generar modelos personalizados de conductor a partir de los datos de conducción obtenidos.
 2. Aplicar modelos de conductores a entornos de simulación multiagente.
 3. Validar los modelos de conductor contra conductores reales.
- Estudiar la efectividad de sistemas de asistencia encaminados a mejorar la eficiencia y analizar el comportamiento de conductor.

Supuestos

- La circulación se supone por la derecha de la vía en el sentido de la circulación, siendo los carriles de lento a rápido de derecha a izquierda respectivamente.
- Los datos de los que extraer el comportamiento se corresponderán con lecturas realizadas durante el día, con buena visibilidad y sin lluvia.
- El tipo de vehículo sobre el que modelar el comportamiento será el *utilitario*.
- El conductor a modelar pertenecerá al grupo más representativo de conductores. Esto se corresponde con varón de 35 a 39 años (ver figura 2).

Restricciones

- Reduciremos el comportamiento del conductor a los de circulación en línea y en cambio de carril ³.
- El Sistema Multiagente hará uso de **Driver-Vehicle Units (DVUs)** como agentes, es decir, usando la tupla (conductor, vehículo) como un todo.
- La resolución máxima del modelo creado será de 1 Hz.
- En el caso de los modelos que hacen uso de Red Neuronal Artificial, no se pueden explicar las razones del comportamiento inferido.

Estructura de la tesis

La tesis está estructurada de la siguiente manera:

En los capítulos **Inteligencia Computacional, Simulación de tráfico** y **Modelos de comportamiento** se expone la revisión realizada del

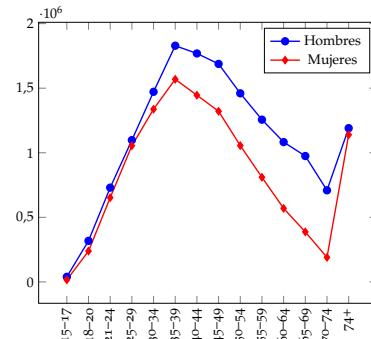


Figura 2: Último censo de conductores según género segmentado por edades. Fuente: Dirección General de Tráfico (dgt.es).

³ Son conocidos en la literatura como **modelos de aceleración** y **modelos de cambio de carril**. Entraremos en detalle sobre ambos conceptos en el capítulo **Modelos de comportamiento**

estado de la cuestión donde se explica en qué punto se encuentra la literatura de los temas en los que se apoya la presente tesis.

En los capítulos [Sistemas desarrollados](#) y [??](#) se explica el método seguido para la confirmación de la hipótesis describiendo además las instrumentaciones, los conjuntos de datos obtenidos, las técnicas utilizadas y las aplicaciones desarrolladas.

Por último en los capítulos [Resultados](#) y [Conclusiones](#) se exponen los resultados y las conclusiones respectivamente extraídos de la tesis. Además, tras las conclusiones se indican una serie de posibles líneas futuras de trabajo consideradas interesantes tras la realización de la tesis.

Estado de la cuestión

Inteligencia Computacional

Todo elemento dentro de un entorno se ve influenciado por una infinidad de variables. Identificar éstas están relacionadas es, en la mayoría de las ocasiones, una tarea que va de lo muy difícil a lo imposible, más aún si añadimos que éstas son muy numerosas y pueden llegar a ser imposibles de cuantificar o incluso de detectar.

La **Inteligencia Computacional (CI, Computational Intelligence)** engloba un conjunto de técnicas que facilitan enormemente estas tareas. El resto del capítulo ofrece una perspectiva de la literatura actual sobre las técnicas de la **CI** que son de interés para esta tesis. Introduciremos además las nociones de “agente”, de “aprendizaje” y de algunas de las técnicas clave de este área. Por último, desarrollaremos en detalle las dos técnicas principales sobre las que se apoya el trabajo teórico de esta tesis: Redes Neuronales Artificiales y Logica Difusa.

Inteligencia Artificial vs. Inteligencia Computacional

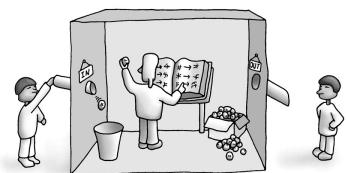
¿Qué es la **CI**? ¿Qué diferencias (si las hay) la separan de la **Inteligencia Artificial (AI, Artificial Intelligence)**? Para responder a esta pregunta tenemos que entender cómo ha evolucionado el concepto de la **AI** a lo largo de su historia.

El primer concepto a introducir es el de **conexionismo**. Se puede considerar a Santiago Ramón y Cajal como principal precursor de esta idea por sus trabajos acerca de la estructura de las neuronas y sus conexiones (e.g. [y Cajal, 1888] y [Ramón and Cajal, 1904]). Otros prefieren citar el trabajo “*A logical calculus of the ideas immanent in nervous activity*” ([McCulloch and Pitts, 1943]) sobre **Redes Neuronales Artificiales (ANNs, Artificial Neural Networks)** o “*The organization of behavior*” ([Hebb, 1968]) acerca de la teoría del aprendizaje como primeros trabajos en este tema. Independientemente de su origen, el conexionismo postula que tanto la *mente* como el *conocimiento* emergen de redes formadas por unidades sencillas interconectadas (i.e. neuronas).

Por otro lado, en 1950, Alan Turing publicó un artículo que comenzaba con la frase “*Can machines think? [Turing, 1950]*”⁴, introduciendo

El **Test de Turing** es un modelo que propuso Alan Turing para probar si una máquina es capaz de exhibir comportamiento inteligente similar al del ser humano. Hay tres participantes, dos humanos (*A* y *C*) y una máquina (*B*), separados entre sí pero pudiendo intercambiarse mensajes de texto. *C* envía preguntas a *A* y *B* sin saber quién es humano y quién es máquina y éstos le responden. Si *C* no es capaz de identificar qué participante es la máquina, se puede concluir que la máquina es inteligente.

⁴ El concepto de “pensar” es en sí un tema controvertido en el propio ser humano: ¿pensar es algo inherentemente biológico? ¿surge de la mente? Tanto si sí como si no, ¿de qué forma lo hace? Por ello existen detractores de la validez del Test de Turing como, por ejemplo, el experimento de la habitación china, propuesto por John Searle.



jolyon.co.uk

Se parte de un Test de Turing donde la máquina ha aprendido a hablar chino. Se reemplaza por una persona que no sabe nada del idioma pero que va equipada con un manual de correspondencias de ideogramas. Cuando una persona le manda mensajes en chino, esta otra responde. Evidentemente la persona no sabe hablar chino, y por ello no podemos afirmar que la máquina sabe hablarlo. Sin embargo, esto lleva a cuestiones quizás más intrigantes. Por ejemplo, si la máquina es capaz de realizar una acción sin entender lo que hace y por qué lo hace, ¿qué garantías tenemos de que el humano sí es capaz? Si los ordenadores operan sobre símbolos sin comprender el verdadero contenido de éstos, ¿hasta qué punto los humanos lo hacen de forma diferente?

el famoso Test de Turing para determinar si una máquina es o no inteligente. Se puede considerar este momento como el punto donde se estableció el objetivo a largo plazo del campo de la [AI](#), ya que en el artículo Turing propuso un método para determinar si una máquina era capaz de exhibir comportamiento inteligente. Sin embargo, no fue hasta 1956 en la Conferencia de Dartmouth [McCarthy et al., 1956] donde John McCarthy acuñó el término [AI](#) a la vez que presentó el tema de la conferencia como la pregunta realizada por Turing en dicho artículo.

A partir de este punto la investigación en [AI](#) recibió muchísima atención por parte de investigadores y gobiernos, lo que se tradujo en financiación. Los estudios estaban dominados por aquellos relacionados con las ideas del conexionismo hasta que en 1969, se publicó el libro *Perceptrons* [Minsky and Papert, 1969] de Marvin Minsky y Seymour Papert, donde se expusieron las limitaciones de los modelos de [ANNs](#) desarrollados hasta la fecha. El impacto fue tal que la investigación en [AI](#) se abandonó casi por completo. Concretamente el conexionismo dejó de estar presente en la literatura científica durante dos décadas. Es lo que se conoce como el primer *AI Winter*.

El AI Winter no sólo se produjo por el efecto gurú del libro *Perceptrons*, aunque éste fue la gota que colmó el vaso. A la emoción inicial por los avances le siguieron muchos años de promesas incumplidas, investigación sin resultados significativos, limitaciones de hardware, aumento de la complejidad del software (los comienzos de la crisis del software [Dijkstra, 1972]). Todo ello provocó un desinterés y una disminución de la financiación que se retroalimentaron la una a la otra.

El interés por el campo volvió de nuevo a principios de los 80 con la aparición en escena de los primeros Sistemas Expertos, los cuales se consideran como el primer caso de éxito en la [AI](#) ([Russell et al., 2003]). A finales de la década, sin embargo, empezaron a resurgir los enfoques conexionistas, en gran parte por la aparición de nuevas técnicas de entrenamiento en perceptrones multicapa y por el concepto de activación no lineal en neuronas [Rumelhart et al., 1985, Cybenko, 1989]). En este momento los sistemas expertos empezaron a perder interés frente al nuevo avance del conexionismo. Ésta década es conocida como segundo *AI Winter* dado que la investigación sobre Sistemas Expertos disminuye. Sin embargo no fue un abandono tan acusado como el del primero.

Mientras que el enfoque clásico de la [AI](#) postulaba que la mente opera de la misma manera que una máquina de Turing, es decir, mediante operaciones sobre un lenguaje de símbolos, el enfoque del conexionismo postula que la mente, el comportamiento inteligente, emerge de modelos a más bajo nivel. Esto provocó que algunas voces se alzaran contra lo que se consideraba el *enfoque incorrecto* de la [AI](#). Después de todo, los modelos desarrollados en los métodos clásicos son fáciles de interpretar mientras que los del enfoque conexionista no son del todo deducibles, más aún si estos problemas son de naturaleza estocástica.

Sin embargo, otras técnicas alineadas con el conexionismo como la Logica Difusa o los Algoritmos Genéticos ganaban popularidad y alimentaban el éxito del nuevo enfoque. Esto provocó una explosión de terminologías para diferenciar las investigaciones de la propia [AI](#) clásica. Por un lado se evitaba el conflicto, nombrando las áreas de trabajo con un término más acorde con el comportamiento o técnica

utilizada. Por otro, se separaba de las connotaciones negativas que fue cosechando la **AI** con el paso de los años (i.e. promesas, pero no resultados).

Lo verdaderamente interesante es ver la evolución de la literatura, y por tanto de los objetivos de la **AI** durante estos años. En el nacimiento del campo, se buscan literalmente máquinas que piensan como humanos, o al menos seres racionales, con mente. Con el paso de los años (y los continuos choques contra la realidad), la literatura va tendiendo hacia la búsqueda de conductas y comportamientos inteligentes cada vez más específicos. Este hecho se hace más patente en este momento, donde cada investigación se nombra de cualquier forma menos con el término **AI** (e.g **Aprendizaje Automático** (ML, Machine Learning), **Sistemas de Recomendación** (RSs, Recommender Systems), o **Procesamiento de Lenguaje Natural** (NLP, Natural Language Processing)). Es evidente que la **AI** se puede observar desde diferentes puntos de vista, todos perfectamente válidos. En [Russell et al., 2003], tras un análisis de las definiciones existentes en la literatura por parte de diferentes autores, se hace énfasis en este hecho mostrando los diferentes puntos de vista a la hora de hablar de lo que es la **AI**. El resumen se puede observar en la figura 3.

Volviendo al tema de la terminología, muchas de las técnicas se fueron agrupando dentro de diferentes áreas. Una de ellas es la conocida como Inteligencia Computacional. Dado que persigue el mismo objetivo a largo plazo y que surje de la propia **AI** parece lógico mantenerla como un subconjunto y no como un nuevo campo del conocimiento humano. Sin embargo, algunos autores abogan por que la **CI** es un campo diferenciado de la **AI**.

Podemos definir la **CI** como la “*rama de la AI que aporta soluciones a tareas específicas de forma inteligente a partir del aprendizaje mediante el uso de datos experimentales*”. A diferencia de la aproximación clásica de la **AI**, se buscan aproximaciones a las soluciones y no las soluciones exactas. Esto es debido a que muchos problemas son de naturaleza compleja, ya sea por la relación entre sus múltiples variables, a la falta de información o a la imposibilidad de traducirlos a lenguaje binario.

Se puede establecer el año 1994 como en el que la Inteligencias Computacionales nace formalmente como área, coincidiendo con el cambio de nombre del *IEEE Neural Networks Council* a *IEEE Computational Intelligence Society*⁵. Poco antes, en 1993, Bob Marks presentaba las que él consideraba diferencias fundamentales entre la Inteligencia Artificial clásica y la Inteligencia Computacional, resumiéndolas en la siguiente frase:

“*Neural networks, genetic algorithms, fuzzy systems, evolutionary programming, and artificial life are the building blocks of CI.*”

Durante estos años ganaba popularidad también el concepto del **Soft Computing (SC)** en contraposición con el **Hard Computing (HC)**.

⁵ <http://cis.ieee.org/>

Thinking Humanly "The exciting new effort to make computers think ... <i>machines with minds</i> , in the full and literal sense." (Haugeland, 1985) "[The automation of] activities that we associate with human thinking, activities such as decision-making, problem solving, learning .. ." (Hellman, 1978)	Thinking Rationally "The study of mental faculties through the use of computational models." (Charniak and McDermott, 1985) "The study of the computations that make it possible to perceive, reason, and act." (Winston, 1992)
Acting Humanly "The art of creating machines that perform functions that require intelligence when performed by people." (Kurzweil, 1990) "The study of how to make computers do things at which, at the moment, people are better." (Rich and Knight, 1991)	Acting Rationally "Computational Intelligence is the study of the design of intelligent agents." (Poole <i>et al.</i> , 1998) "AI ... is concerned with intelligent behavior in artifacts." (Nilsson, 1998)

Figura 3: Diferentes objetivos perseguidos por la Inteligencia Artificial. Las filas diferencian entre pensamiento o comportamiento mientras que las columnas separan entre inteligencia humana o el ideal de la inteligencia (racionalidad). Fuente: *Artificial Intelligence: A Modern Approach* (3rd Ed.), [Russell *et al.*, 2003].

HC y **SC** son la forma de referirse a la computación convencional frente al **SC**. El **HC** basa sus técnicas en aquellas basadas en modelos analíticos definidos de forma precisa y que en ocasiones requieren mucho tiempo de cómputo. Están basados en lógica binaria, análisis numérico, algoritmos y respuestas exactas. El **SC** por otro lado es tolerante a la imprecisión y al ruido y tiende a llegar a soluciones aproximadas de manera más rápida. Se basa en modelos aproximados, emergencia de algoritmos y modelos estocásticos.

El **SC** engloba las técnicas que buscan resolver problemas con información incompleta o con ruido. Debido a que el conjunto de técnicas definidas como constituyentes del **SC** son las mismas que se usan en la **CI** algunos autores consideran ambos términos equivalentes. Nosotros consideramos que el **SC** es un punto de vista de la computación y que la **CI** es un área de la Artificial Intelligence hace uso de métodos del **SC**.

Aprendizaje

TODO! Fusionar los dos párrafos siguientes y el de la margin note con el párrafo anterior y crear una nueva introducción para la parte del aprendizaje. Sobre todo hacer hincapié en que en el HC lo normal es desarrollar una solución para el problema en cuestión mientras que en el Soft Computing lo normal es tener un modelo que se tunea y/o que aprende a solucionar un problema determinado.

La resolución clásica a un problema suele ser la aplicación de una secuencia de instrucciones basadas en un conjunto de símbolos (e.g. una función escrita en el lenguaje de programación C). Esta forma de solucionar un problema no *aprende* a solucionarlo. Se puede interpretar como que la solución está grabada en su memoria.

En la aproximación de la **CI**, existen modelos y existen técnicas para hacer aprender esos modelos. La aplicación de dichas técnicas es lo que se conoce como **aprendizaje**. Las técnicas de aprendizaje en **CI** se suelen clasificar en 3 paradigmas:

- **Supervisado.** El entorno presentado al modelo consiste en un conjunto de la forma $D = (I_i, O_i) | \forall i \in \mathbb{N}$, donde cada O_i es la salida esperada del modelo a la entrada I_i . Los algoritmos tratarán de ajustar el modelo todo lo posible para que las salidas obtenidas sean lo más parecidas a las salidas originales. Este paradigma de

entrenamiento suele estar relacionado con problemas de *regresión*.

- **No supervisado.** Al modelo se le ofrece un conjunto de la forma $D = I_i | \forall i \in \mathbb{N}$, donde cada I_i es una entrada al problema, pero del que no conocemos la salida. Los algoritmos dentro de esta categoría harán uso de estos datos para ir reajustando el modelo tratando de encontrar las estructuras ocultas entre dichos datos (e.g. patrones, correlaciones o clases). Es un paradigma de entrenamiento íntimamente relacionado con problemas de *clasificación*.
- **Por refuerzo.** **TODO!**Explicar

Algunos autores hacen uso de técnicas pertenecientes a ambos paradigmas en forma de aproximación híbrida para suplir deficiencias u optimizar/acerlar el aprendizaje. Un claro ejemplo lo podemos ver en [Hinton, 2006], donde los autores hacen uso de *autoencoders* como técnica no supervisada para la inicialización de los pesos de una red neuronal y posteriormente realizan un entrenamiento supervisado para la optimización es éstos.

Técnicas en la Inteligencia Computacional

Bajo el paraguas de la **CI** se incluyen muchas técnicas diferentes, entre las cuales están las usadas en esta tesis. El resto de la sección describe el funcionamiento de cada una de estas técnicas. **TODO!**Reescribir esto porque parece que lo ha escrito un crío de la ESO

Redes Neuronales Artificiales

Son herramientas que tratan de replicar las funciones cerebrales de un ser vivo de una manera muy fundamental, esto es, desde sus componentes más básicos, las neuronas. Para ello se basan en estudios de neurobiología y de ciencia cognitiva moderna del cerebro humano⁶.

Una **ANN** es independiente del problema a solucionar. Se la puede considerar como una caja negra que aprende las relaciones que subyacen en los datos de un problema para abstraer el modelo a partir de éstos. Estas características de aprendizaje y abstracción son los factores determinantes por los que son usadas en prácticamente todas las áreas de la ciencia y de la ingeniería ([Du and Swamy, 2006]).

EL PRIMER TRABAJO en la disciplina se le atribuye a los investigadores McCulloch-Pitts por su modelo de neurona artificial ilustrado en la figura 5 ([McCulloch and Pitts, 1943]). Existen diferentes tipologías y formas de operar con redes, pero todas funcionan de la misma manera: unidades (e.g. neuronas) interconectados mediante enlaces

⁶ Aún apoyándose en la topología y funcionamiento del cerebro humano para realizar el símil, lo cierto es que dichos modelos distan aún de considerarse *cerebros artificiales*. La red neuronal más compleja hasta la fecha es la propuesta en [Trask ANDREWTRASK et al.,], con alrededor de 160,000 parámetros a ser ajustados (podemos abstraernos y pensar en ellos como conexiones entre neuronas). Si comparamos esta cifra sólo con las del neocórtex (figura 4) hace que, tecnológicamente hablando, nos quedemos con la sensación de estar aún a años luz de aproximarnos a la complejidad de un cerebro humano.

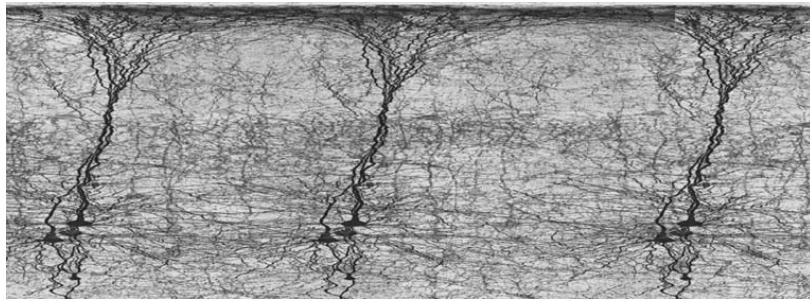


Figura 4: Sección del neocórtex humano, región asociada a las capacidades cognitivas y que supone alrededor de un 76% del volumen total del cerebro humano. Está distribuido en 6 capas y miles de columnas que las atraviesan, cada una con alrededor de 10,000 neuronas y un diámetro de 0,5mm. Como dato anecdótico, se estima que sólo en el neocórtex humano existen alrededor de 20,000 millones de neuronas, cada una de las cuales conectada a entre 100 y 100,000 neuronas vecinas ([Pakkenberg and Gundersen, 1997]).

Esto supone entre $2 \cdot 10^{12}$ y $2 \cdot 10^{15}$ conexiones. Fuente: [Blue Brain Project](https://bluebrain-project.eu/)

Figura 5: Variación de la representación del modelo de neurona artificial propuesto por McCulloch y Pitts. En éste, cada una de las entradas x_i es incrementada o inhibida aplicando el producto con su peso asociado w_i . La activación vendrá determinada por la aplicación de una función (denominada "de activación") a la suma de los valores. Esta variación en concreto incluye una entrada x_0 y un peso w_0 como bias de la neurona para la variación dinámica del umbral de activación.

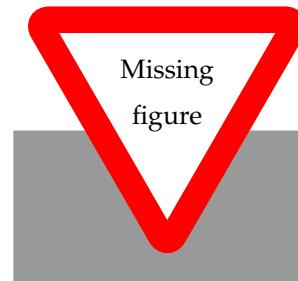


Figura del modelo artificial de McCullochs-Pitts

Este primer modelo de neurona proponía una función escalón para determinar si la neurona se activaba o no (analogía al funcionamiento de la neurona artificial). **TODO!** Parrafito sobre la limitación de la neurona singular para dar hilo a las topologías.

Posteriormente aparecieron nuevos modelos de neuronas con diferentes funciones de activación. De éstas, las más comunes son las de tipo sigmoide **7.TODO!** Actualizar esto porque creo que empieza a er ás común la ReLU, que se usa hasta en la sopa. Una o dos ilustraciones mostrando los diferentes tipos (grafiquitas) de las funciones de activación en redes neuronales. Para que sea de un poco más de 1990 y no esta mierad. Poner escalón, sigmoidal y tanh (como clásicas) y ReLU y adjustid ReLU (o como coño se llamen) como modernas.

EXISTEN DIFERENTES TOPOLOGÍAS DE REDES NEURONALES o arquitecturas dependiendo de qué forma toma el grafo que modela las neuronas y sus conexiones. En este caso, las redes neuronales pueden ser de dos tipos:

- **Feed-Forward.** Sus grafos no contienen ningúñ ciclo (figura XXX) | **TODO!** Esa pedaaaaaoso de figura. A lo mejor hacer una que incluya tres ejemplos.. Es la topología más usada en aplicaciones prácticas debido a su sencillez y su efectividad. En ellas el flujo de

⁷ Concretamente la función logística de pendiente 1 definida como:

$$\frac{1}{1 + e^{-x}} \quad (1)$$

información sigue un camino desde las entradas hasta las salidas, sin ninguna retroalimentación. No es requisito que las neuronas se agrupen en capas, aunque suele ser la estructura común. A las redes de más de dos capas ocultas (i.e. las capas que se encuentran entre la capa de neuronas de entrada y la capa de neuronas de salida) se las denomina “profundas” o *deep*. Algunos tipos pertenecientes a esta categoría pueden ser el Perceptrón [Rosemblat, 1957], el Perceptrón multicapa [Rumelhart et al., 1986], el algoritmo LVQ y su sucesor los Mapas Auto-Organizados [Kohonen, 1998].

- **Recurrentes.** Sus grafos contienen uno o más ciclos, de tal manera que el flujo de información de salida de una neurona puede llegar a afectar a su propio estado. Estas topologías representan de una forma más fiel las bases biológicas de las ANN, pero son más complejas a la hora de operar y entrenar. Algunos casos particulares de este tipo de arquitectura son las Redes de Hopfield [Hopfield, 1982] o las memorias LSTM (del inglés Long-Short Term Memory) [Hochreiter & Schmidhuber, 1997].

TODO!Más cosas a añadir del paper de modelling behabior of lane change execution

ANNs are CI techniques used as a general solution in prediction and classification problems. They are composed by single processing units called neurons and are usually classified depending on how are arranged and connected those neurons. Depending on its architecture, the ANN can be trained on a problem by means of a supervised training scheme (i.e. by showing tuples of input → output) or non-supervised training scheme (i.e. by showing a whole dataset and letting them to find patterns and solutions by themselves). They have been used successfully in several fields like speech recognition [20], series and sequences prediction [21] or terrain classification [22].

In this paper, for mimicking the lane-change acceptance of a particular user, two different families of ANN are proposed: MLPs and CNNs. Figure 2. Example of two feed-forward architectures: (a) A Multilayer Perceptron architecture; (b) A Convolutional Neural Network architecture. Figure 2. Example of two feed-forward architectures: (a) A Multilayer Perceptron architecture; (b) A Convolutional Neural Network architecture. Both families rely on the same model of artificial neuron, the McCulloch-Pitts model [23], and belong to the category feed-forward, i.e. a Directed Acyclic Graph (DAG) where the inputs are presented by one end of the DAG (called input layer) and the outputs are collected at the other end (called output layer).

In these two models, the connections are represented with a real value called weight imitating the behaviour of biological neurons with the strength of connection between axon and dendrites. Typically, the training process belongs to the supervised scheme and is based on the modification of these weights to obtain the desired response, following the principles of the Hebbian theory.

One of the most difficult tasks on these models is their hyper-parameter tuning, and it is difficult because ANNs involve a lot of them. Apart from the family, and the arrangement of their neurons that fortunately, in the most of cases, comes imposed by the family, there are number of neurons, number of layers, training parameters (depending on the learning algorithm), and they follow the non-free lunch theorem [24]. The problems derived from that are two: the high bias or under-fitting and the high variance or over fitting.

To avoid under-fitting, the classical technique is to increase the size of the network, both deepness and/or number of neurons. Over-fitting, on the other hand, is a little bit harder because it can be motivated for the lack of data, the number of layers, the number of neurons, etc. In these cases, the solution may be the augmentation of training set, either naturally or artificially, and/or regularization techniques. We will talk later about the data augmentation techniques used in this work.

The regularization technique used in this experiment is the one called Dropout [25]. It's a simple technique where in each training epoch some random neurons are, indeed, dropped. This technique slows down the convergence process a bit, also avoiding the neurons to rely only in few of their inputs, thus reducing the over-fitting. Figure 3 shows an example of this techniques through three different epochs.

TODO! Quizá esto quede bien al lado como ejemplo de dropout.

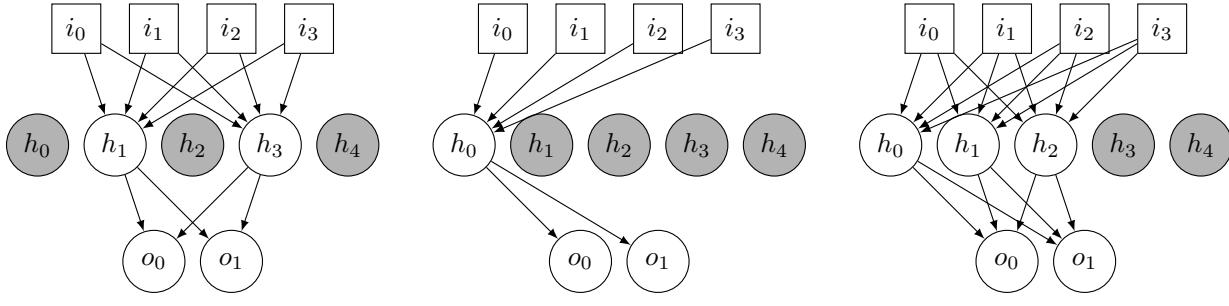


Figura 6: Three training epochs on a multilayer perceptron with a 0.5 dropout rate (i.e. 50% probability for a neuron to be disabled) in its hidden layer. The grey neurons are the ones disabled each epoch.

Multilayer Perceptrons In a MLP, the neurons are arranged in layers so that all the neuron outputs on one layer are the inputs for all the neurons in the next layer. The first and last layers are called input layer and output layer respectively. The inner layers are called hidden layers. As shown in Figure 2 (b) (a two-layered MLP), in this architecture the inputs are usually presented as vectors. As they have been used extensively in several areas with great success, we use them here to make a comparison of the improvement of the use of CNNs over MLPs.

Convolutional Neural Networks As its name suggests, a CNN uses convolutions, i.e. mathematical operations that filter regions of a struc-

ture (generally in the form of a matrix or a cube) for pattern identification and/or structure transformations. A CNN has also an arrangement of layers, but they work differently. Firstly, this architecture is separated into two groups of layers or phases that can be called pattern identification phase and prediction phase. The pattern recognition phase works with layers dealing with pattern recognitions and input sub-sampling while the prediction phase works with an MLP using the output of the pattern recognition phase as input. Secondly, the inputs are arranged as n -dimensional matrices as opposed to the MLP that are always presented as vectors; this fact allows us working more easily with properties and patterns arranged in an n -dimensional space like images (2-dimension matrix) or videos (3-dimension matrix). The layer types in that phase are as follows:

1. Convolutional Layer. Given a n -dimensional input, a convolution or filter is an n -dimensional pattern (composed by neurons) that travels along the n -dimensional space to generate a new n -dimensional space. A convolutional layer is a layer composed by m filter and generate a set of m new n -dimensional spaces which will be the new input for another layer.
2. Local Response Normalization (LRN) layer [26]. This layer is intended to enhance the difference between the existent values in a space. They are commonly placed after a convolution layer to increase the contrast of the values in the space. In this paper, every convolution layer is followed by an LRN layer.
3. Pooling layer. Pooling is grouping together elements. A pooling layer is intended to subsample a space to ease its management. Given an n -dimensional space, a pooling filter of dimension travels the space in steps of size generating one value per step. When placed after convolutions, it is expected to subsample the space without losing the recognized pattern while reducing computation costs and avoiding overfitting. The most common operation used in pooling is the max operation.

TODO!Desarrollar un poquito el tema de los grafos computacionales indicando que si bien no se tratan de una técnica dentro de la inteligencia computacional, sí que se usan para representar operaciones.

Grafos computacionales

En la actualidad el concepto de grafo computacional se relaciona directamente con las redes neuronales, y por ello se introduce el concepto en este apartado. Sin embargo, no se trata de un concepto exclusivo de esta técnica. De hecho, ni siquiera es un concepto perteneciente al área de la inteligencia computacional, sino que es simplemente una forma de representar operaciones sobre datos.

Formalmente, un **grafo computacional** es un grafo dirigido donde los vértices representan operaciones sobre datos mientras que las aristas representan el flujo de dichos datos. La figura XXX ilustra un ejemplo de un grafo computacional.

Una ventaja al representar un modelo como grafo computacional es que ayuda a abstraerse de las formas de las entradas y las salidas, facilitando el trabajo de operaciones en batch. Otra ventaja, todavía mayor, es que, al organizar de entrada a salida (en la figura XXX de izquierda a derecha) las operaciones que se necesitan para obtener una salida a partir de una entrada, en los casos donde el objetivo es optimizar la salida permiten fácilmente representar el gradiente al organizarlo del modo contrario (es decir, de las salidas a la entrada o, en el caso de la figura XXX de derecha a izquierda).

TODO!Poner aquí un ejemplo de grafo computacional, por ejemplo una recta o algo así. Algo que tenga al menos dos pasos y explicar cómo se calcula la derivada y por qué esto viene genial. Luego, en el siguiente capítulo se puede explicar el backpropagation con esto (ver <http://colah.github.io/posts/2015-08-Backprop/>)

Aprendizaje en Redes Neuronales Artificiales

TODO!Falta un **huevazo**. Por lo menos hay que desarrollar las arquitecturas de MLP y CNN (que son las que uso), explicar el descenso del gradiente y ADAM (y cómo se llega a él explicando los conceptos de momento y su puta madre), explicar también cuales son los problemas de High bias y High variance o overfitting y underfitting y ilustraciones bonitas del to. También hay que explicar qué es un grafo computacional, aunque quizás esto mejor después de la lógica difusa y antes de agentes inteligentes. También hay que hablar de qué es el aprendizaje profundo, deep learning vs shallow learning.

La lógica nace en el siglo IV a.C. dentro de la física Aristotélica, que permaneció inalterada hasta la revolución científica (alrededor del siglo XVI. d.C.), momento en que se separó y permaneció como disciplina paralela perteneciente más al campo de la filosofía que de la física y la matemática. Empezó a relacionarse de nuevo con la matemática a principios del siglo XIX y a principios del siglo XX la lógica y la teoría de conjuntos pasaron a convertirse en partes indispensables la una de la otra. Por ello suelen ir de la mano cada vez que se habla de la una y de la otra. La evolución de la teoría de conjuntos (Cantor, finales del siglo XIX, buscar referencia) y su unión con la lógica es una época bastante convulsa dentro de la historia de la matemática.

¿Qué es el aprendizaje profundo?

Logica Difusa

La lógica matemática (y por extensión la teoría de conjuntos) tiene como misión servir de fundamento del razonamiento matemático. Se basa en la definición precisa y con rigor de un razonamiento evitando cualquier tipo de ambigüedad y de contradicción. Es por ello que la lógica tradicional no suele servir como fundamento de razonamientos del mundo real.

Los conceptos que se manejan en el mundo real suelen ser vagos, llenos de imprecisiones. Además tienden a ser nombrados cualitativamente, no quantitativamente, y cuando existe una correspondencia, ésta suele estar marcada por la subjetividad de los términos.

Explicar lógica difusa y control difuso. Indicar los controladores difusos de segundo, tercer y sucesivos niveles.

Teoría de conjuntos difusos

A diferencia de los conjuntos tradicionales, los conjuntos difusos expresan el grado de pertenencia de un elemento a la categoría representada por el conjunto. La definición podría escribirse de la siguiente manera:

TODO! Creo que habría que definir antes qué es un dominio

Definición 1: Sea X una colección de elementos. Se define al **conjunto difuso** F como un conjunto ordenado de pares de la forma $F = (x, \mu_F(x)) | x \in X$, siendo $\mu_F(x) \in [0, 1] \forall x \in X$.

La función de la definición 1 se denomina **función de pertenencia**, y caracteriza unívocamente a un conjunto difuso del dominio de X .

TODO! Quizá aquí habría que decir qué es una partición de un dominio

Operaciones entre conjuntos

La unión, intersección y el complemento son operaciones básicas en la teoría de conjuntos. **TODO!** hablar aquí de tnorm, tconorm y complemento, pero someramente. No hay que enrollarse demasiado.

Razonamiento

Al igual que en la lógica tradicional, en la **Logica Difusa (FL, Fuzzy Logic)** el razonamiento o inferencia es la manera de extraer conclusiones a partir de premisas en función de un conjunto de reglas.

8

Estas reglas se expresan como implicaciones, definidas típicamente en lógica difusa como $A \rightarrow B \equiv A \wedge B$.

Las dos formas de extraer conclusiones a partir de premisas en **FL** son el *modus ponens* generalizado (del que hablaremos) y el *modus tollens* generalizado, modificaciones sobre los procesos de inferencia *modus ponens* y *modus tollens*⁹, dos formas similares de razonamiento (figura 7). Nosotros centraremos nuestro discurso en la primera.

EL **MODUS PONENS GENERALIZADO** es una generalización del *modus ponens* de la lógica tradicional donde, en lugar de expresas las reglas de forma absoluta, se expresan de forma aproximada. En la figura 8 se ilustra las diferencias fundamentales entre ambos modos.

Para determinar qué grado le asignamos a un consecuente a partir de las premisas parciales y las reglas que dirigen el razonamiento se utiliza un método denominado **regla composicional de inferencia**.

⁸ La implicación en lógica se representa como $A \rightarrow B$, donde A es cualquier operación de premisas y B la conclusión que arrojan.

En lógica tradicional, el valor de verdad de una implicación es equivalente al de la expresión $\neg A \vee B$. Sin embargo, en lógicas multivaluadas (y por tanto en lógica difusa) esta equivalencia da lugar a razonamientos que se pueden considerar contraintuitivos.

En el caso concreto de la lógica difusa se han propuesto muchas cantidad de equivalencias. Sólo en los trabajos [Kiszka et al., 1985] se analizan 72 alternativas al operador $\neg A \vee B$.

El operador más usado no obstante es el definido como $A \wedge B$ debido a su rendimiento (en la implicación de Mamdani la *T*-norma se implementa como el operador mínimo).

⁹ En realidad se llaman *modus ponendo ponens* ("la forma que al afirmar, afirma") y *modus tollendo tollens* ("la forma que al negar, niega").

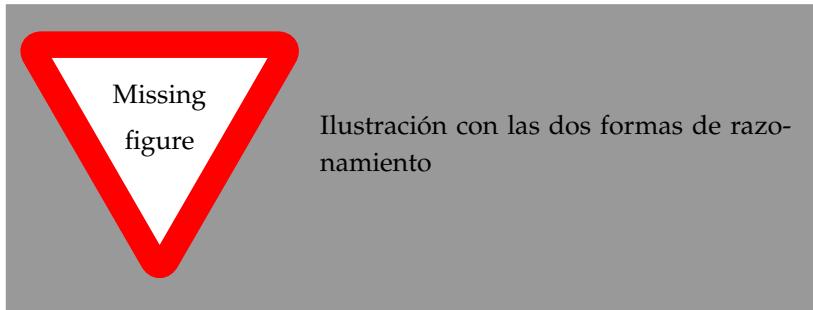


Figura 7: Formas de razonamiento en lógica tradicional: *modus ponendo ponens* y *modus tollendo tollens*.

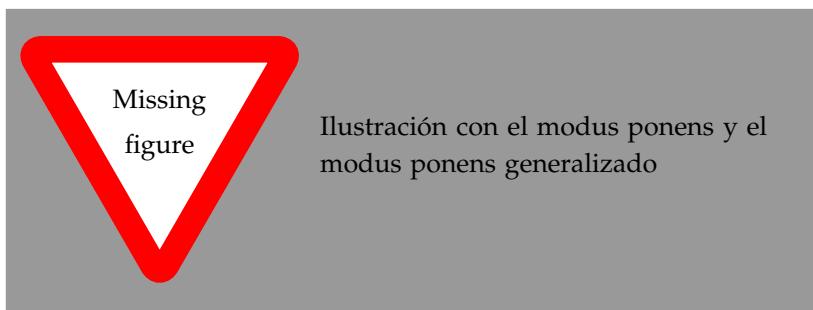


Figura 8: Proceso de razonamiento según el *modus ponens* tradicional frente al *modus ponens* generalizado. En el primero, si la premisa A es cierta, entonces la conclusión B será cierta. En el segundo, dado que la premisa A no es del todo cierta (es A'), entonces la conclusión B será cierta sólo en parte (B').

Una regla $A' \rightarrow B'$ se puede representar como una implicación caracterizada por una función $I(\mu_A(x), \mu_B(y))$ ([Ful,]).

TODO! Explicar mejor porque es terrorífico.

...

Tengo que hablar también del softmax, el por qué usarlo. También de la entropía cruzada y de por qué usamos este loss sobre el resto.

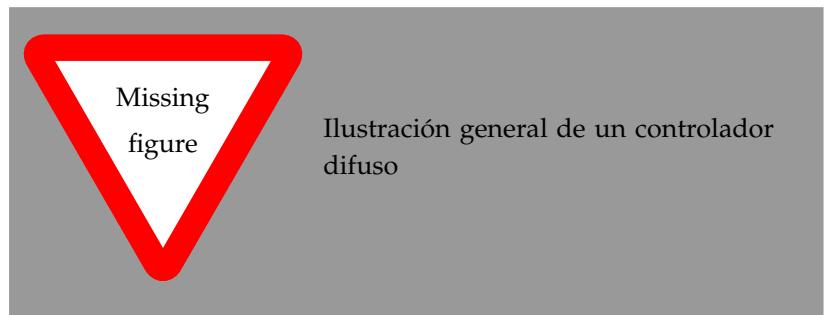
He visto una cosa curiosa que es el "negative sampling". Podría hablar de ello y usarlo porque haría el entrenamiento mucho más rápido. Por lo que me ha parecido ver, se usa en lugar de la capa softmax y de lo que va es transformar la capa softmax (que es coñazo de calcular) en una capa de $k+1$ clasificadores binarios, donde hay 1 correcto y k incorrectos. Se usa en clasificación.

En [Ma, 2004] hay un capítulo de razonamiento que parece que está guay. Revisarlo un poco a fondo a ver si merece la pena tirar or ahí.

Fuzzy Inference System

Los Sistemas de Inferencia Difusa (FISs, *Fuzzy Inference systems*) (o Sistemas de Control Difuso (FCSs, *Fuzzy Control systems*)) son el caso de éxito de la lógica difusa que más resultados ha cosechado tanto a nivel académico como a nivel industrial. Se trata sistemas que utilizan el razonamiento difuso para inferir una respuesta a partir de un conjunto de entradas.

Figura 9: Diagrama del esquema general de un Sistema de Inferencia Difusa.



Habitualmente son descritos como un componente dividido en tres bloques conceptuales:

- **Fuzzificación.** Traducir los valores de entrada en crudo del dominio sobre el que está definida cada variable lingüística a sus respectivos grados de pertenencia a conjuntos difusos a través de sus funciones de pertenencia. **TODO!** Ojo, algunos controladores toman como valores de entrada conjuntos difusos según [Ma, 2004]. Habrá que buscar sobre ello.
- **Inferencia.** Realiza todo el proceso de razonamiento difuso a partir del conjunto de reglas que dan significado a este controlador difuso.
- **Defuzzificación.** Traduce los conjunto difuso resultado del proceso de inferencia a valores del los dominios sobre los que están definidos dichos conjuntos difusos. **TODO!** En un sugeno, la salida es una función directamente así que se podría especificar que en un tipo Sugeno, se puede ver como que la salida son sólo singletones, manteniendo la generalización del proceso de funcionamiento de un **FIS**.

Esta división se ilustra en la figura 9.

HAY VARIOS TIPOS DIFERENTES DE **FIS**, aunque tienden a seguir el esquema básico de un controlador difuso típico (figura 9).

Sistemas de tipo Mandamni Son la primera aproximación de **FIS** propuestos

Sistemas de tipo Takagi-Sugeno ...

Hablar someramente de los tres tipos clásicos que se usan, e indicar que al final los más usados son el Mandamni y el Sugeno. Añadir también quizás una tabla comparativa entre los tres o al menos entre los dos principales:

El consecuente de un **FIS** de tipo Mandamni siempre es un conjunto difuso. Por tanto, el proceso de sacar un valor crisp es costoso.

Lo bueno, se mantiene significado semántico de las salidas. El consecuente en un Sugeno es un valor, y se puede decir que no necesita proceso de defuzzificación. Si embargo, la respuesta pierde significado semántico si la suma de la fuerza de salida no es 1 (no entiendo qué quiero decir con esto).

Agentes inteligentes

Si echamos un poco la vista atrás, en la figura 3 se mostraban los cuatro objetivos perseguidos por la AI. En uno de ellos en particular se la entiende como el estudio del conseguir que las entidades (e.g. sistemas, software, ...) actúen de la manera más inteligente posible. A dichas entidades se las conoce como *agentes*, concretamente en este contexto como *agentes inteligentes*¹⁰. Sin embargo, si es difícil encontrar un consenso en la definición de agente más lo es a la hora de definir cuándo la conducta de éstos es inteligente.

Lo que sí existe es una serie de características comunes que se repiten a lo largo de la literatura (figura 10):

- Operan siempre en un **entorno**, ya sea éste físico (e.g. una red de carreteras para un vehículo autónomo) o virtual (e.g. un cliente de correo electrónico para un clasificador de spam).
- Tienen la capacidad de **percibir** el entorno por medio de *sensores* y de **actuar** sobre él por medio de *actuadores*.
- Son **autónomos** en el sentido de que pueden actuar sin intervención externa (e.g. humana u otros agentes) teniendo control sobre su estado interno y su comportamiento. Algunos autores les presuponen una autonomía absoluta mientras que otros hablan de que sólo es necesaria cierta autonomía parcial.
- Tienen **objetivos** a cumplir, actuando para ello sobre el entorno de la manera que les indique su comportamiento.
- Pueden ser **sociales**, es decir, tienen la capacidad de comunicarse con otras entidades (e.g. otros agentes) para llevar a cabo sus objetivos.

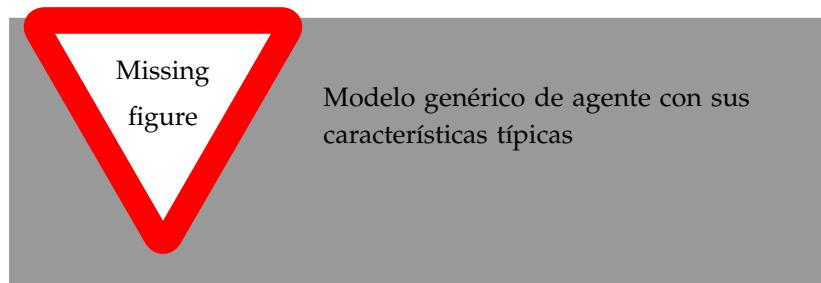
Por tanto nosotros usaremos la siguiente definición: Un agente es una entidad física o virtual que realiza una acción¹¹ de manera total o parcialmente autónoma dada una secuencia de percepciones del entorno en el que se ubica.

Pero, ¿qué hace a un agente inteligente? Según algunos autores, el hecho de que posea unos objetivos y autonomía suficiente para cumplirlos ya denota inteligencia (TODO!encontrar el trabajo y citar). Según otros, es necesario que el comportamiento sea flexible, esto es, que sea reactivo (reacciona ante el entorno que percibe),

¹⁰ En realidad los autores prefieren denominarlo *agente racional*, dado que captura la esencia de lo que es un comportamiento inteligente. Sin embargo, según esta definición, hasta un elemento tan rudimentario como un termostato puede de ser considerado como elemento inteligente, ya que realiza siempre la mejor acción para cumplir sus objetivos, por simples que puedan parecer. Dónde está el límite entre qué es y qué no es un agente inteligente cae dentro de los dominios de la filosofía.

¹¹ En [Russell et al., 2003] se define como "... just something that acts" alegando que la palabra *agent* proviene del latín *agere*. Para clarificar esto, *agere* es la forma verbal para *hacer*, pero impriime un significado de movimiento/actividad diferente que no tiene mucho que ver con *hacer* como forma verbal para *crear* o *dar forma* (de lo que se ocupa el verbo *facere*). Por ello, el verbo *actuar* es un verbo que se relaciona con *agere* y de ahí la definición.

Figura 10: Esquema de un agente y sus propiedades. Aunque no existe una definición comúnmente aceptada de agente, sí que existe una serie de propiedades que los que los identifican. Es autónomo, opera realizando acciones sobre un entorno dependiendo de las percepciones que le llegan de éste y tiene la capacidad de comunicarse con el resto de elementos, incluidos otros agentes.



proactivo (iniciativa para tratar de cumplir sus objetivos) y social (capaz de interactuar con otros agentes para cumplir sus objetivos) [Wooldridge et al., 1995]. Y otros directamente exigen, además, un comportamiento racional a la hora de cumplir los objetivos para calificarlo de inteligente (**TODO!**encontrar el trabajo y citar).

Por tanto, asumiremos la definición ofrecida por [Russell et al., 2003] donde, se indica que un agente es considerado **agente inteligente** cuando éste realiza la mejor acción posible (según un criterio de medida). En este contexto, “la mejor acción posible” se refiere en términos de objetivos y comprensión del entorno, que puede ser o no correcta ¹².

Las nociones de agentes inteligentes y la de **CI** van de la mano. Esto es debido a que su definición funciona a la perfección para las técnicas de la **CI**, esto es, agentes autónomos que perciben el entorno (problema) y actúan de la mejor manera posible sobre él (resuelven) de acuerdo a su conocimiento del medio y su estado interno (en base a algoritmos como **ANN**, **FL**, ...). Por ello desde mediados de los años 1990 el concepto de agente inteligente ha ganado tanta popularidad ¹³.

Tipos de entorno

La tupla (*entorno, agente*) es esencialmente una metáfora para referirse a la tupla (*problema, solución*) por lo que existen casi tantos entornos diferentes como problemas.

Afortunadamente es posible caracterizar los entornos de acuerdo a un conjunto de propiedades o dimensiones. Este conjunto es usado por la totalidad de la literatura a la hora de caracterizar entornos:

- **Observable.** Un entorno es **totalmente observable** cuando el agente es capaz de captar toda la información relevante para la toma de una decisión y no necesita mantener ningún modelo interno del entorno, **parcialmente observable** cuando la información obtenida es incompleta o tiene ruido y **no observable** cuando el agente no posee sensores.
- **Multiagente o. monoagente.** Un entorno es **multiagente** cuando

¹² Que la comprensión del entorno no sea total es un factor clave que diferencia la racionalidad de la omnisciencia. La omnisciencia significa conocer el resultado de toda acción antes de realizarla y por tanto implica el conocimiento de absolutamente todos los detalles del entorno. La racionalidad existe dentro de un contexto de conocimiento limitado.

¹³ Tanto es así que en algunos trabajos se define el objetivo de la **AI** como la implementación de la función agente, esto es, la función que realiza la correspondencia de una percepción a una acción, para un problema dado.

requiere de múltiples agentes interactuando para llegar a una solución mientras que es **monoagente** cuando sólo requiere de uno para ello.

- **Determinista o. no determinista.** Si el estado del entorno actual depende totalmente del estado anterior, se dice que el entorno es **determinista**. Si no es así, se considera **no determinista o estocástico**¹⁴.
- **Episódico o. secuencial.** Un entorno en el que las acciones se dividen atómicamente donde cada una de ellas conlleva un ciclo de (percepción, decisión, acción) y sin relación una con otra se denomina episódico. Si en lugar de ello la acción del agente puede afectar a las decisiones futuras se dice que el entorno es **no episódico o secuencial**.
- **Estático o. dinámico.** Si durante la toma de decisión en entorno no cambia, se dice que el entorno es **estático**. En caso contrario, se dice que es **dinámico**.
- **Discreto o. continuo.** Esta dimensión en realidad se divide en cuatro, estado del entorno, tiempo en el entorno, percepciones y acciones. La dimensión es **discreta** cuando ésta se divide en una partición discretizada, y **continua** cuando no. Por ejemplo, en el Juego de la Vida de Conway, si se modela en un sistema multiagente, tanto el estado (i.e. tablero) como el tiempo (i.e. turnos) como las percepciones y acciones están discretizadas. Sin embargo, en un entorno de conducción automática se puede determinar que las cuatro dimensiones son continuas.
- **Conocido o. desconocido.** Un entorno es **conocido** cuando es posible determinar cuál va a ser el resultado de una acción. Si por el contrario no es posible, entonces se dice que es **desconocido**.

Arquitecturas

Existe una serie de arquitecturas básicas o tipos de agentes que dependen principalmente de cómo perciben el entorno y de qué forma se comportan aunque, dependiendo de los autores, las nomenclaturas, tipologías y esquemas pueden variar. Por ello, hemos decidido ofrecer una abstracción donde poner de manifiesto las partes comunes y no comunes entre arquitecturas.

La figura 11 muestra el esquema de las partes principales de un agente. En general, todo arquitectura de agente inteligente está cortada por el mismo patrón y obedece al siguiente funcionamiento:

1. El agente, a través de sus **sensores**, percibe el entorno en el que éste se mueve.
2. De acuerdo a cómo recordamos el entorno (llamémoslo **modelo del entorno**), el agente genera una **interpretación del entorno** tal

Figura 11: Arquitectura básica de un agente. Aunque existen múltiples arquitecturas diferentes, todas se basan en la misma estructura. El agente percibe el entorno, lo interpreta y toma la decisión de cómo actuar sobre él.



y como supone el agente que es. Esto es, percibe el entorno y, de acuerdo a sus sensaciones, lo entiende de una determinada forma.

3. Esta interpretación del entorno es pasada a un proceso de **inferencia** el cual, en función la implementación para la consecución de sus objetivos, generará una serie de acciones a realizar sobre el entorno.
4. Estas acciones serán ejecutadas sobre el entorno a través de una serie de **actuadores**, provocando probablemente una modificación en éste que será percibida de nuevo en momentos sucesivos.

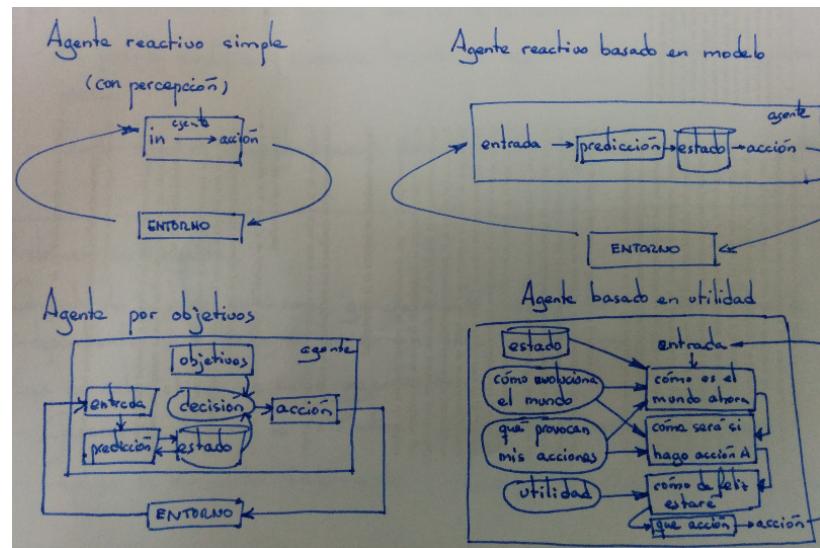
La primera diferencia clave surge en la manera que se ofrece al bloque de inferencia la interpretación del entorno y genera la primera clasificación (figura 12):



Figura 12: Ilustración de la diferencia entre un agente sin modelo de entorno y uno con modelo de entorno. Cada acción realizada por el agente con modelo de entorno tiene en cuenta el estado del entorno en momentos pasados. El agente sin modelo de entorno actúa tal y como interpreta el entorno en cada momento, como si sufriese de amnesia.

- **Sin modelo de entorno.** Si el agente ofrece su interpretación del entorno directamente, sin hacer uso de información histórica sobre el entorno que se ha movido. Otras formas de denominar a estos agentes es como *agentes reactivos* o *simple-reflex agents* ([Russell et al., 2003]). Sin embargo, los términos *reactivo* o *reflex* para algunos autores se refieren a la forma de inducción de acciones a partir de percepciones, y por ello preferimos la denominación *sin modelo de entorno*.
- **Con modelo de entorno.** El agente genera su interpretación más detallada del entorno a partir de las percepciones que llegan desde los sensores y de el histórico del entorno que mantiene. Otras formas de llamarlo es *agentes con estado* o *Model-based*, pero lo hemos denominado de esta manera para diferenciar que el modelo que se mantiene en este punto pertenece únicamente al entorno.

Figura 13: Distintas arquitecturas de agentes en función del comportamiento. Dependiendo de las acciones a realizar, se identifican tres tipos, los reactivos que aplican una acción sin proceso deductivo y los basados en modelo y utilidad (en algunos contextos denominados deliberativos) que basan su comportamiento en alguna forma de deducción.



La siguiente clasificación viene motivada por la forma de deducir el conjunto de acciones a ser aplicadas por parte de los sensores. En este sentido podemos identificar tres tipos distintos de agentes (figura 13):

- **Reactivos.** Son aquellos donde el uso de un proceso de razonamiento explícito es demasiado costoso para producir una conducta en un tiempo aceptable. Se suelen implementar como correspondencias (percepción → acción) sin ningún razonamiento adicional.
- **Basados en objetivos.** Plantean una deducción de forma que determinan cuál sería el estado del entorno tres aplicar varias o todas las acciones que puede realizar. En base a los resultados, selecciona la acción que se corresponde con sus propios objetivos.
- **Basados en utilidad.** Éstos plantean una deducción similar a los basados en objetivos con la diferencia de que, mientras los primeros sólo diferencian entre entorno objetivo o no objetivo, éstos asignan un valor (i.e. *utilidad*) a cada uno de los escenarios de entorno posibles para seleccionar el mejor (e.g. el que mayor utilidad tiene).

En la literatura se describen muchos tipos de agente, como por ejemplo los agentes BDI (Believe-Desire-Intention) o los agentes lógicos (i.e. el entorno se representa con reglas lógicas y se infiere mediante métodos como por ejemplo deducción lógica o prueba de teoremas). Sin embargo, éstos pueden definirse en los términos aquí expuestos (figuras 11, 12 y 13).

Sistema Multiagente

Son aquellos sistemas compuestos de dos o más agentes que interactúan de alguna manera para llegar a una solución.

Cuando los agentes son inteligentes y el problema cae dentro del dominio de la [AI](#), el ámbito de estudio es el de la [Inteligencia Artificial Distribuida \(DAI, Distributed Artificial Intelligence\)](#), la rama dedicada a la resolución de problemas mediante procesamiento descentralizado.

Desde el punto de vista de la ingeniería de sistemas, y a pesar del aumento de complejidad, los [MAS](#), al ser sistemas inherentemente descentralizados, ofrecen múltiples ventajas frente a los sistemas centralizados tradicionales:

- Los sistemas son más robustos y fiables frente a fallos, ya que los agentes son autónomos e independientes del resto.
- La modificación del sistema se puede realizar sobre la marcha, agente a agente sin necesidad de parar el sistema al completo.
- Su diseño fuerza a desacoplar las dependencias entre agentes.
- Son inherentemente paralelizables y por tanto pueden llegar a ser más eficientes que sus homólogos centralizados. Este punto es quizá el más controvertido, ya que esta ganancia en eficiencia se puede perder rápidamente en función de la cantidad de comunicación existente entre agentes.
- Debido al nivel de complejidad alcanzado en los sistemas existentes en la actualidad, la computación se distribuye a través de múltiples sistemas, normalmente heterogéneos. La tendencia además es a la alza. La definición de los [MAS](#) hace natural su implementación en este tipo de arquitecturas.

Desde el punto de vista de la [AI](#) podemos añadirles la ventaja de que permiten el estudio de conductas complejas de poblaciones a partir del comportamiento de sus elementos básicos, facilitando el estudio de modelos y teorías sobre éstos.

LA COMUNICACIÓN ENTRE AGENTES, se trata de una característica clave en un [MAS](#), ya que para denominarse de esta manera dos o más agentes deben interactuar (i.e. comunicarse) entre sí. Esta interacción puede implementarse de diversas maneras ¹⁵ y siempre toman una o las dos formas siguientes (figura 14):

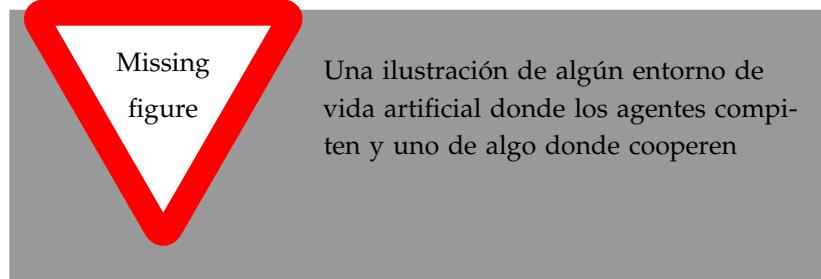
- **Cooperación.** Los agentes intercambian información entre sí para llegar a una solución. Esta solución puede ser fragmentada (i.e. cada agente posee parte de la solución y se comunican para ir avanzando de forma común hacia la solución global) o poseerla uno o

¹⁵ Las formas clásicas de comunicación son el de paso de mensajes, los sistemas de pizarra y la estigmergia. Para los dos primeros existen dos propuestas para estándar de lenguaje de comunicación, [Knowledge Query and Manipulation Language \(KQML\)](#) ([Finin et al., 1994]) y [Agent Communications Language \(ACL\)](#) ([Poslad, 2007]). La tercera forma de comunicación suele ser muy dependiente del problema y no se apoya en lenguajes estándares. Se trata de una forma de comunicación basada en la modificación del entorno, como la efectuada por las hormigas en la búsqueda de alimento, donde éstas dejan rastros de feromonas modificando el entorno para modificar el comportamiento del resto de la colonia.

varios agentes que hacen uso de más agentes para ir avanzando la solución.

- **Competición.** Los agentes compiten dentro de un entorno, generalmente mediante la adquisición de recursos limitados. Un ejemplo de este tipo de sistemas multiagente puede ser aquellos sistemas de vida artificial.

Figura 14: La comunicación entre agentes puede ser de dos tipos: *colaborativa*, donde los agentes tratan de llegar a una solución intercambiándose información y *competitiva*, donde los agentes compiten unos contra otros en un entorno.



Simulación de tráfico

El tráfico es un sistema de comportamiento tan caótico que extraer modelos de su funcionamiento es una tarea prácticamente imposible. Por un lado, la cantidad de variables existentes es innumerable y en muchos casos con relaciones no detectables a primera vista. Por otro, es un sistema que funciona en el mundo real, es decir, donde las mediciones en unos casos afectan a los resultados y en otros, directamente no se pueden realizar, ya sea por regulaciones vigentes o por imposibilidad física.

Los simuladores de tráfico son herramientas de software que, usando diferentes modelos para representar sus componentes, describen el tráfico como sistema, permitiendo, entre otros:

- Extracción de resultados y conclusiones en escenarios de tráfico determinados.
- Implementación de técnicas determinadas en tráfico simulado para su evaluación sin necesidad de alterar el tráfico real.
- Introducción de modificaciones en puntos determinados (e.g. espaciales o temporales) de un escenario conocido para estudiar la divergencia en la evolución del tráfico.

El objetivo principal de un simulador de tráfico es el de hacer que sus modelos se parezcan lo máximo posible a la realidad. En este capítulo vamos a ver cuál es la realidad actual de este tipo de simuladores, cuáles son sus diferentes tipologías y formas de modelar los diferentes aspectos del tráfico y, posteriormente, qué simulador de los disponibles en el mercado es el idóneo para nuestro trabajo.

Limitaremos nuestro estudio a los simuladores de DVUs, obviando otros tipos de simulación de tráfico que nada tienen que ver con esta temática, como por ejemplo los orientados a la evaluación de sistemas de señalización inteligentes (e.g. [Jin et al., 2016]), a la estimación de emisiones (e.g. [Quaassdorff et al., 2016]) o los de carreras (e.g. [Wymann et al., 2013]).

A lo largo del capítulo, se utilizarán indistintamente los términos DVU, conductor y vehículo para referirse al mismo concepto. En caso de no ser así, se indicará de manera explícita.

El entorno de simulación [TORCS](#), usado en multitud de concursos e investigaciones, se trata de un juego. Los juegos son un *sandbox* perfecto ya que presentan una abstracción de complejidad acomodada sobre el dominio que trabajar.



Algunos trabajos interesantes que usan como base [TORCS](#) para emular comportamientos de conductores reales (conduciendo en el simulador) son [Muñoz et al., 2010], donde se usan perceptrones multicapa entrenados con técnicas de *back propagation* y [Van Hoorn et al., 2009], donde también se usan perceptrones, pero esta vez entrenados mediante Algoritmo Genético multiobjetivo. Sin embargo, este tipo de modelos se encuentran más cercanos al nivel de control que al nivel táctico (ver figura 27).

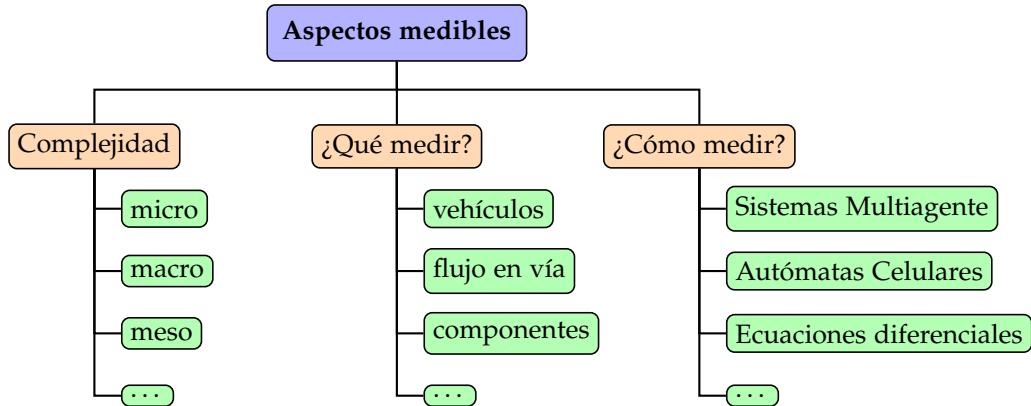


Figura 15: Los aspectos medibles del problema del tráfico son muy diversos, y dependen del nivel de granularidad (i.e. complejidad) al que se quiere llegar, de qué queremos medir y de cómo lo queremos hacer.

Clasificación de simuladores de tráfico

Los aspectos simulables y medibles del problema del tráfico son muy diversos, dependiendo sobre todo:

- Del nivel de **complejidad** del tráfico (e.g. modelar una vía por la que circula un centenar de coches no es lo mismo que modelar una ciudad por la que circulan millones).
- De **qué** queremos medir (e.g. evaluar a un conductor en una situación determinada o evaluar la evolución del flujo de tráfico en un cuello de botella causado por un accidente).
- De **cómo** (e.g. un Autómata Celular se modela de forma diferente a un modelo lineal de vías o carriles).

El resto de la sección ofrece una visión de las principales categorías existentes para clasificar a los simuladores de tráfico.

Tipos de simulador en función de la complejidad

La complejidad en una simulación se refiere al nivel de detalle que queremos alcanzar durante la ejecución de la misma y/o en sus resultados. Es evidente que según aumentamos el detalle en la simulación aumenta la cantidad de cálculo. Por ejemplo, si queremos modelar el comportamiento de 10 billones de canicas cayendo por un tubo es considerablemente más eficiente modelarlas como un fluido con una serie de parámetros que como una colección de elementos individuales, cada uno con sus propiedades (e.g. masa, aceleración, ...) e interaccionando entre sí.

El caso de los simuladores de tráfico es similar. En éstos existe un amplio intervalo de granularidades, desde por ejemplo el flujo de entrada en una autovía hasta el consumo de carburante de un vehículo en ciudad. Lo más común es clasificar los simuladores dentro de dos grandes grupos, los cuales se ilustran en la figura 16:

- **Microsimulación** o simulación de tipo **micro**. Su objetivo es estudiar, desde un punto de vista de granularidad fina (e.g. vehículos o peatones), las micropropiedades del flujo de tráfico como, por ejemplo, los cambios de carril, las aproximaciones a vehículos de lanteros o los adelantamientos, para evaluar su comportamiento. Sus dos principales ventajas son la posibilidad de estudiar el tráfico como un todo a partir de sus elementos más simples (ofreciendo una representación más fiel de éste) y la posibilidad de estudiar cada elemento por separado. Sin embargo, su principal desventaja es que cada elemento de la simulación requiere de cómputo independiente y por tanto simulaciones con alto contenido de elementos pueden llegar a ser inviables ¹⁶.
- **Macrosimulación** o simulación de tipo **macro**. Este tipo de modelos centran su esfuerzo en estudiar el flujo de tráfico como un todo (generalmente como fluido), explorando sus macropropiedades (e.g. evolución del tráfico, efectos onda, velocidad media o flujo en vías). Su ventaja principal es que a nivel macroscópico permiten estudiar propiedades que a nivel microscópico requeriría una cantidad ingente de recursos. Sin embargo, con este modelo es imposible obtener información precisa de un elemento en particular del tráfico.



Aunque ésta es la categorización típica de modelos, en la literatura aparecen otros tipos de modelo con granularidades que pueden considerarse no pertenecientes a ninguno de estos dos conjuntos. Éste es el caso de los simuladores de tipo **sub-micro** y de tipo **meso**, de los cuales se muestra un ejemplo en la figura 17.

Los **sub-micromodelos** especifican granularidades por debajo del nivel de “vehículo” o “peatón”. Por ejemplo, en ([Minderhoud, 1999]) trabaja a nivel de funcionamiento del control de crucero inteligente de un vehículo en función del entorno del vehículo.

Por otro lado los **mesosimuladores** (e.g. [Munoz et al., 2001] o [Casas et al., 2011]) nacen para amortiguar los problemas inherentes a la complejidad en los micromodelos y a la falta de resolución en los macromodelos.

Dado que el objetivo de la tesis la evaluación de modelos de comportamiento de conductores concretos, nos ceñiremos al uso de simuladores que modelen un nivel de granularidad **micro**.

¹⁶ Existen técnicas de computación distribuida que superan ampliamente los límites impuestos por la computación en un único nodo. Un ejemplo relativamente reciente es el simulador de IBM *Megaffic*. Éste implementa un modelo de granularidad micro donde cada elemento es un agente independiente (i.e. Sistemas Multiagente) usando para ello entornos con cientos de núcleos de proceso que proveen de capacidad suficiente para modelar ciudades enteras como Tokio (ver [Osogami et al., 2012] y [Suzumura and Kanezashi, 2012]).

Figura 16: Clasificación clásica de simuladores en función de la granularidad (complejidad) de la simulación. En la imagen de la izquierda se muestra un ejemplo clásico de macrosimulador donde el tráfico se modela como un flujo a través de las vías. En la de la derecha, se ilustra un modelo clásico de microsimulación donde cada elemento (en este caso vehículos) circula por un carril de la vía.

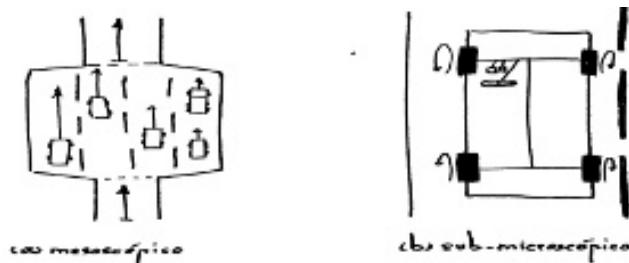
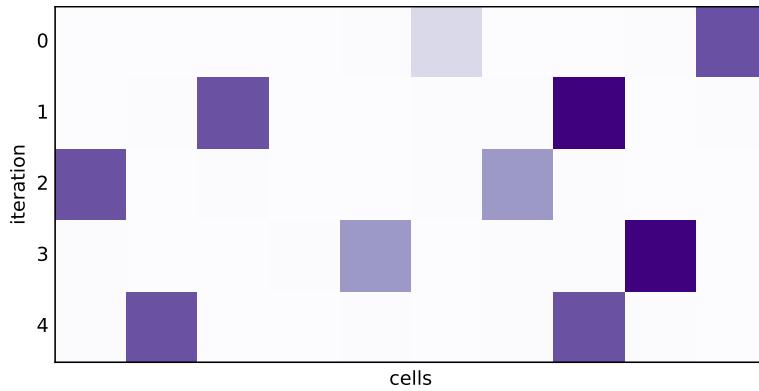


Figura 17: Otras aproximaciones alternativas de modelos en función de la complejidad. Ejemplo de mesosimulación como ventana de microsimulación dentro de un flujo de tráfico en un macrosimulador (e.g. [Munoz et al., 2001]) y ejemplo de submicrosimulación donde se modelan componentes internos de un vehículo.

Tipos de simulador en función del espacio y el tiempo

Existen otras dos formas de clasificar los simuladores en función de cómo evolucionan en la simulación las dimensiones **espacio** y **tiempo**. Sin embargo, aunque *complejidad, espacio y tiempo* son dimensiones diferentes a la hora de clasificar simuladores, el tipo de simulador según una de ellas tiende a determinar en gran medida los tipos en las demás.

Figura 18: Simulador de tráfico basado en CA. El espacio se divide en celdas que pueden estar vacías u ocupadas por un vehículo a una velocidad (más oscuro implica más lento). Concretamente muestra la evolución a lo largo del tiempo del movimiento de 2 vehículos donde en eje x representa la posición en la vía y el eje y el momento temporal (iteración) de la vía. Fuente: simulador nagel-scherckenberg-demo (Ver capítulo Sistemas desarrollados).

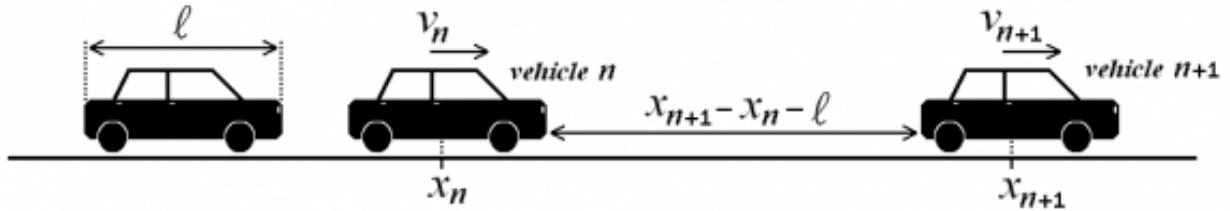


En el caso de la dimensión **espacio**, la clasificación diferencia las simulaciones que se mueven por un espacio discreto o por uno continuo:

- **Espacio discreto.** Simulación donde el espacio está dividido en celdas que (normalmente) sólo pueden estar ocupadas por un elemento en un momento determinado. Este es el caso, por ejemplo, de los simuladores basados en **Autómatas Celulares (CAs, Cellular Automata)** (figura 18).
- **Espacio continuo.** Simulación que transcurre en una secuencia infinita de puntos en el espacio. Es el caso por ejemplo de los simuladores basados en modelos lineales (figura 19).

En el caso de la dimensión **tiempo**, la división se realiza en los mismos términos que en los del espacio:

- **Tiempo discreto.** También denominada *simulación de eventos discretos*, divide el tiempo en intervalos discretos, generalmente (aunque existen excepciones) de longitud fija durante toda la simulación.



Los simuladores basados en **CA**s son también simuladores típicos discretos, ya que cada posición en el espacio se va calculando para cada intervalo discreto de tiempo (figuras 18 y 20).

- **Tiempo continuo.** En estos simuladores el tiempo es un factor más para un modelo de ecuaciones diferenciales. La figura 19 ilustra un modelo de *car-following* que puede implementarse en una simulación de tiempo continuo si la aceleración viene determinada por un modelo que entre otros factores incluye el tiempo.

En nuestro caso queremos conocer la situación exacta del vehículo y no una situación aproximada en una separación discreta del espacio. Esto nos dirige hacia simuladores de **espacio continuo**. Por otro lado, realizamos la recolección de datos en intervalos cuantificables de tiempo, los cuales serán usados para modelar los comportamientos de los conductores y para contrastar los resultados; por tanto, la elección en la dimensión tiempo ha de ser de **eventos discretos**.

Modelos de microsimulación

Los simuladores que se basan en un modelo de granularidad micro están en su mayoría implementados en dos tipos de paradigma: Autómatas Celulares y Sistemas Multiagente.

Existe un tercer punto de vista a la hora de implementar este tipo de modelos, que es el de los sistemas de partículas. Sin embargo, su ámbito de aplicación es el mismo que el del punto de vista macroscópico, esto es, usar sistemas de partículas para el análisis del tráfico como fluido. Por tanto, el resto de la sección describirá los dos tipos principales sin tener en cuenta éste último.

Microsimulación basada en Autómatas Celulares

Un **CA** es una colección ordenada de celdas (*células*) ordenadas en un espacio n -dimensional que parcelan el universo de estudio. Cada una de ellas se encuentra en un estado (e.g. contiene un valor numérico), y el estado de toda la malla se actualiza de manera síncrona¹⁷ (i.e., todas a la vez) en intervalos regulares de tiempo denominados *ciclos*. El cambio de estado de cada célula depende de los valores

Figura 19: Ejemplo de un modelo lineal en un espacio continuo. La posición del vehículo es un valor $x \in \mathbb{R}$. Este ejemplo muestra un modelo de *car-following* donde el comportamiento de la aceleración del vehículo es determinado por la distancia al coche siguiente. Fuente: [Tordeux et al., 2011].

¹⁷ Existen arquitecturas diseñadas para operar de esta manera, esto es, arquitecturas basadas en **CA** (e.g. [Margolus, 1993]). En ellas, cada ciclo de reloj actualiza todas las celdas de memoria del autómata. Éstas arquitecturas se suelen usar para la implementación de modelos físicos superando en varios órdenes de magnitud la capacidad computacional de las arquitecturas tradicionales.

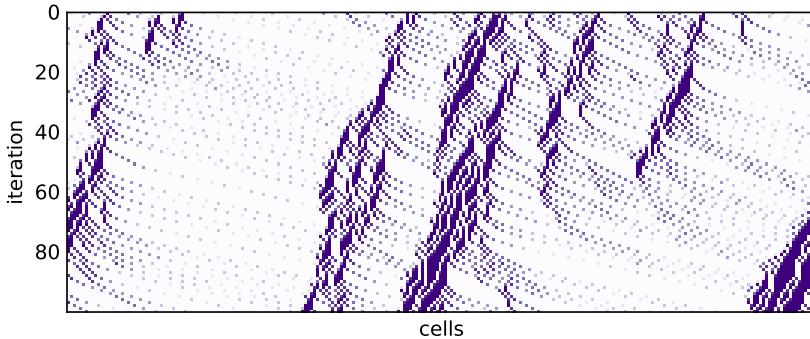


Figura 20: Aparición de retenciones en una autopista de 250 celdas usando el modelo Nagel-Scherckenberg. La densidad de ocupación es de 50 coches en la vía, la velocidad máxima es de $5c/\Delta t$ y la probabilidad de frenada es de $p = 0,5$. Se puede observar en la figura cómo se desplazan las olas del atasco a lo largo de las 100 iteraciones. Fuente: simulador `nagel-scherckenberg-demo` (Ver capítulo [Sistemas desarrollados](#)).

El modelo Nagel-Scherckenberg es un Autómata Celular que basa su funcionamiento en los siguientes aspectos:

- La vía está dividida en celdas de longitud $7,5m$. La razón de este valor es que ésta es la distancia media entre los parachoques traseros de dos coches consecutivos en un atasco.
- La celda puede tener dos estados, vacía o con un vehículo a velocidad $v = \{0, \dots, v_{max}\} \in \mathbb{N}$. La unidad de medida es $c/\Delta t$ (celdas por unidad de tiempo).
- Δt queda establecido en $1s$, considerado el tiempo medio de reacción de un conductor ante una eventualidad. Esto hace, por ejemplo, que una velocidad de $6c/\Delta t$ sea $45m/s$ ($162km/h$).
- En cada ciclo y para cada vehículo, se realizan tres acciones de manera consecutiva: (i) acelerar una unidad si no está a la máxima velocidad o frenar si se ve obligado, (ii) freno aleatorio (la velocidad se reduce en una unidad hasta un mínimo de $v = 1c/\Delta t$ con una probabilidad de $p = 0,5$) y (iii) reposicionamiento.

de las células vecinas y del mismo algoritmo de modificación al que responden todas y cada una de las células.

Estos modelos de microsimulación, debido a la propia naturaleza de los [CA](#), se encuentran clasificados como simuladores de tiempo y espacio discreto, y se usan debido a su facilidad de implementación y a su eficiencia, ya que son fácilmente paralelizable.

El modelo clásico de esta aproximación es el propuesto por Nagel-Scherckenberg en su artículo *A cellular automaton model for freeway traffic* [Nagel and Schreckenberg, 1992], un modelo teórico creado para la simulación de tráfico en autopistas. La figura 20 muestra la evolución del tráfico en una autopista a lo largo del tiempo en una implementación basada en este paradigma.

En general los modelos de la literatura suelen ser una variación del de Nagel-Scherckenberg con modificaciones para estudiar aspectos concretos de modelos de tráfico o para dotarle de un mayor realismo. Algunos ejemplos de estas variaciones son la modificación del paso de *aleatorización* (e.g. [Barlovic et al., 1998]), reglas para determinar niveles de molestia a vehículos vecinos ([Wagner et al., 1997]), celdas más pequeñas (e.g. [Krauss et al., 1997]) para comprobar la metaestabilidad del flujo de tráfico, o modelos y reglas para cambio de carril en vías de dos carriles ([Brilon and Wu, 1999, Nagel et al., 1998]).

Microsimulación basada en sistemas multiagentes

Los modelos basados en Autómatas Celulares, aunque interesantes, no son suficientemente realistas desde un punto de vista microscópico. Por poner un ejemplo, en una situación típica de un modelo Nagel-Scherckenberg, los vehículos realizan aleatoriamente aceleraciones y deceleraciones de $27km/h$. Es más, en una situación favorable, cualquier vehículo puede realizar una aceleración de 0 a $162km/h$ en tan sólo 6 segundos. Por tanto, no ofrecen una visión demasiado realista ni fiable en caso de querer realizar estudios muy detallados de tráfico a nivel micro.

Por otro lado, en un Multi-Agent System cada uno de los agentes

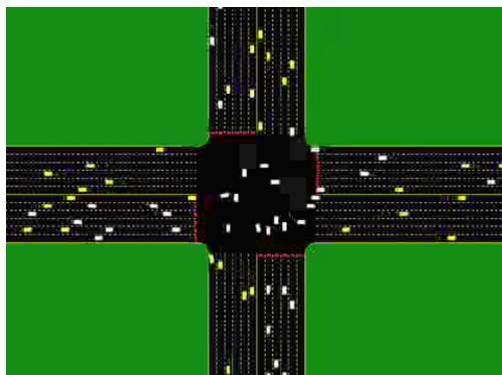


Figura 21: Simulación de comportamiento en intersección basada en un MAS. En ésta, cada uno de los vehículos representa a un vehículo real que posee un controlador para hacerlo autónomo. Modelar este caso de estudio con una arquitectura basada en MASs permite centrarse en el diseño del agente en concreto (i.e. el controlador de conducción del vehículo) y estudiar el comportamiento emergente surgido de la interacción de todos los agentes. Fuente: Proyecto AIM (<http://www.cs.utexas.edu/~aim/>).

tiene su propia entidad dentro del sistema. Esto es, perciben tanto el entorno como al resto de agentes y actúan de acuerdo a lo percibido y a su comportamiento. Basarse no sólo en las magnitudes físicas del resto de vehículos (e.g. distancia, aceleración, ...) sino también en un comportamiento de conducción ofrece un interesante campo de estudio a nivel cognitivo. Se entra habla más en detalle sobre los MASs en el capítulo [Inteligencia Computacional](#) y sobre los comportamientos concretos de agentes de interés para esta tesis en el capítulo [I](#). Por ello, este apartado únicamente hará una pequeña introducción a estudios existentes y aplicaciones de simuladores basados en este modelo.

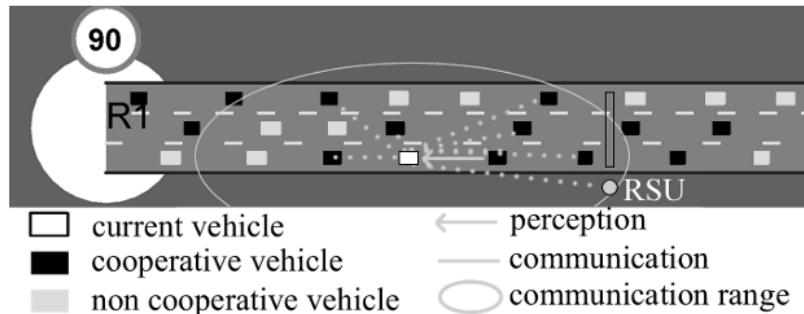
A diferencia de los [CAs](#), los [MASs](#) pueden emplazarse en un entorno virtual que represente un espacio continuo y no discreto. Esto permite modelar con mayor fidelidad magnitudes físicas asociadas a cada agente (e.g. posición y velocidades actuales, dimensiones del vehículo, masa, velocidad máxima permitida, ...). Sin embargo, aun así no es una propiedad inherente de éstos. No existe ninguna limitación en cuanto a la representación del espacio y es perfectamente posible representar un modelo basado en Autómatas Celulares usando para ello Sistema Multiagente.

Cada uno de los agentes es independiente del resto, y una consecuencia directa es que el comportamiento de cada individuo permite evaluar comportamientos grupales complejos, como el descrito en la figura 21. Esta independencia da la posibilidad de tener todos los agentes diferentes entre sí, ofreciendo la ventaja de permitir experimentar con diferentes perfiles de conducción (e.g. un perfil agresivo en un flujo de tráfico dominado por conductores tranquilos). Esto es debido a que en un [MAS](#) cada agente es una parte del sistema y las decisiones de cómo se ha de comportar las toma él mismo. Desde el punto de vista de un [CA](#), el comportamiento existe en cada celda, sin dar control al contenido o estado de cada celda.

En general los estudios basados en este modelo suelen seguir el patrón 1 [DVU](#) \equiv 1 agente, dando así una enorme cantidad de posibilidades a experimentar. Por ejemplo en [Das et al., 1999] se hace uso de sistemas difusos para decidir cómo comportarse en la vía mientras que en [Ehlert and Rothkrantz, 2001] se hace uso de un patrón

reactivo. Otros, como [Dia, 2002] o [Balmer et al., 2004] hacen uso de encuestas o censos para establecer las propiedades y calibrar los parámetros de diferentes tipos de agentes.

Figura 22: Captura de pantalla del simulador [MovSim](#). Este simulador implementa un modelo multiagente donde los vehículos incorporan sistemas de comunicación vehicular. El estudio se centra en el uso de la comunicación entre vehículos para el acoplamiento dinámico de vehículos en sus respectivos carreteras. Fuente: [Gu et al., 2015].



Los estudios en materia de simuladores de tráfico con Sistema Multiagente no se limitan a vehículos, sino que se usan también en otras áreas como el control de luces de tráfico o agentes para peatones entre otros. Por ejemplo el estudio presentado en [Clymer, 2002], los agentes del sistema son las señales de tráfico luminosas y no los vehículos, y el objetivo es adaptar la señalización en una red de carreteras para minimizar al máximo el tiempo de espera por parte de los vehículos en las intersecciones gestionadas por las señales. Otro ejemplo es el propuesto por en [Galis and Rao, 2000], donde los agentes, en lugar de ser los vehículos son los tramos de las carreteras; en él, los vehículos poseen comportamiento, pero lo reciben del agente que les guía de acuerdo a la zona en la que se encuentran. Esto tiene la ventaja de que el paso de información a vehículos dentro de la misma zona se realiza mucho más rápido en un entorno distribuido.

En los últimos años, otro concepto que está en auge es el de las redes intervehiculares e intravehiculares, [Vehicle-to-Vehicle \(V2V\)](#) y [Vehicle-to-Infrastructure \(V2I\)](#) respectivamente. El modelo de [MAS](#) permite la implementación rápida de diferentes políticas y protocolos de comunicación via sensores y actuadores para estudiar estos tipos de redes de comunicación (figura 22). Estudios como por ejemplo [Shiose et al., 2001] o [Galis and Rao, 2000] hacen uso de un [MAS](#) para implementar diferentes formas de [V2V](#) con el objetivo de aliviar congestiones de tráfico (en el primer caso) y por el propio estudio de las comunicaciones en si (en el segundo caso). En el caso de redes [V2I](#), un buen ejemplo es [Dresner and Stone, 2004], donde se representan como agentes tanto los vehículos como las intersecciones de la vía. Éstas gestionan un sistema de reservas de tokens que los vehículos solicitan cuando van a entrar en la intersección y devuelven cuando salen, gestionando comunicando en todo momento mediante eventos los cambios en dicho sistema. El estudio concluye que una comunicación de este tipo es más eficiente que una intersección clásica basada en señales de tráfico luminosas.

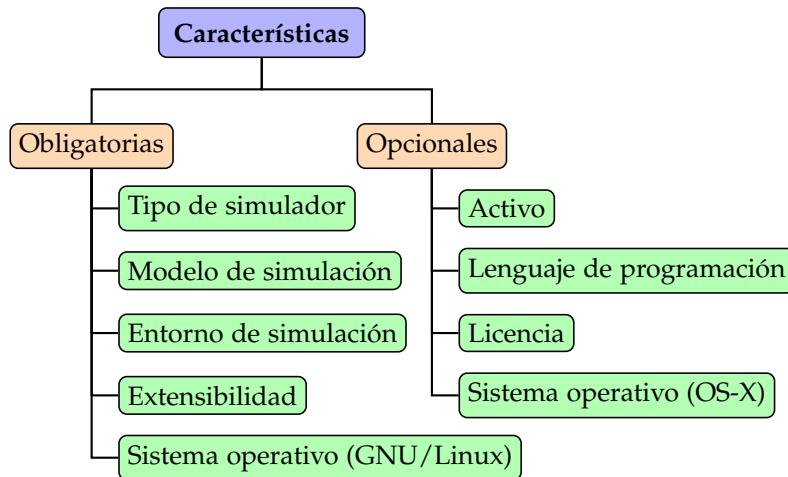


Figura 23: Características obligatorias y deseables del simulador donde implementar nuestros modelos personalizados de conductor.

Software de simulación

Para la realización de esta tesis es necesario contar con un paquete de simulación que permita modelar un Sistema Multiagente en el que poder ejecutar los modelos de comportamiento desarrollados.

Aunque en un principio se ha valorado el desarrollo de una solución propia, la oferta de simuladores en el mercado es muy amplia, cada uno de ellos implementando uno o varios modelos diferentes bajo distintas licencias. Por ello se ha optado por la elección de un paquete de simulación ya desarrollado.

Para elegir el mejor simulador que se adapte a nuestras necesidades se ha realizado un listado de características obligatorias y, de este modo, realizar una primera criba eliminando simuladores no aptos (resumidos en la figura 23):

1. **Tipo de simulador.** Para nuestras necesidades es necesario un simulador que implemente **microsimulación**, ya que es el único tipo de granularidad que permite evaluar el comportamiento de un conductor independientemente del resto de la simulación. Además, debido a la forma en la que se recolectan los datos, es necesario que represente un **espacio continuo** y una dimensión de **tiempo discreto** con una resolución de al menos 1 segundo.
2. **Modelo de simulación.** Debe ofrecer un entorno basado en un **MAS** donde cada **DVU** se comporte como agente individual.
3. **Entorno de simulación.** Debe ofrecer un entorno de **simulación de tráfico general**, permitiendo la creación de escenarios. Quedan excluidos los simuladores de propósito específico o de casos particulares como simuladores de autopistas, congestiones o colisiones.
4. **Extensibilidad.** El simulador debe permitir extender de alguna la ejecución de los modelos desarrollados en los agentes (**DVUs**). Aunque se puede considerar que si es simulador **Software Abierto**

(OSS, Open Source Software), se puede modificar su comportamiento para adecuarlo a los modelos desarrollados, es mejor que el propio software ofrezca los mecanismos necesarios para la integración sin necesidad de tocar los fuentes del sistema.

5. **Sistema operativo.** Es imprescindible que el software se ejecute sobre sistemas operativos GNU/Linux por la configuración de los sistemas sobre los que se trabaja.

Posteriormente se ha desarrollado un listado de características deseables. No son determinantes para descartar simuladores pero sí favorecen la elección de unos sobre otros.

1. **Activo.** Es preferible que el sistema esté activamente desarrollado porque eso favorece la aparición de parches y mejoras sobre el software. En caso contrario, se trata de un proyecto con poca actividad por parte de sus autores.
2. **Lenguaje de programación.** Es favorable la implementación de los modelos en código Python.
3. **Licencia.** Es preferible una licencia de tipo OSS ya que, en caso de error o falta de funcionalidad, es posible acceder a los fuentes para modificarlos.
4. **Sistema operativo.** Es favorable que el sistema se ejecute en entornos tipo OS-X.

Entornos de simulación a estudiar

El primer listado de características deja atrás la mayoría de simuladores (una gran cantidad de ellos son o bien de propósito específico, están desarrollados para sistemas operativos Windows o no permiten extender su modelo). Tras la selección, nos quedamos con los siguientes simuladores: [AORTA](#), [MatSIM](#), [MitSIM](#), [MovSim](#) y [SUMO](#).

Dichos entornos están prácticamente igualados en las características presentadas, tal y como se puede observar en el cuadro 1. Sin embargo, en materia de extensibilidad, [SUMO](#) es el único que permite el desarrollo de [DVUs](#) de manera externa. El resto requiere la modificación del código fuente del simulador para variar los comportamientos de los conductores. [AORTA](#) además no es un proyecto que se mantenga activo en la actualidad (las últimas modificaciones del repositorio datan de principios del año 2014).

No obstante se ha tratado de modificar los comportamientos de los conductores en los cuatro simuladores para validar este hecho y ha quedado patente que es mucho más eficaz usar [SUMO](#) como simulador para nuestro estudio.

	AORTA	MatSIM	MitSIM	MovSim	SUMO
Activo	✗	✓	✗	✓	✓
Lenguajes de programación	Scala	Java	C++	Java	C++ y Python
Licencia					
Propietaria	✗	✗	✗	✗	✗
OSS	✓	✓	✓	✓	✓
GPL	✓	✓	✗	✓	✓
Extensibilidad					
Código fuente	✓	✓	✓	✓	✓
API	✗	✗	✗	✗	✓
Sistemas Operativos					
GNU/Linux	✓	✓	✓	✓	✓
OS X	✓	✓	✗	✓	✓
Windows	✓	✓	✗	✓	✓

Cuadro 1: Tabla comparativa donde se contrastan las características de los si-

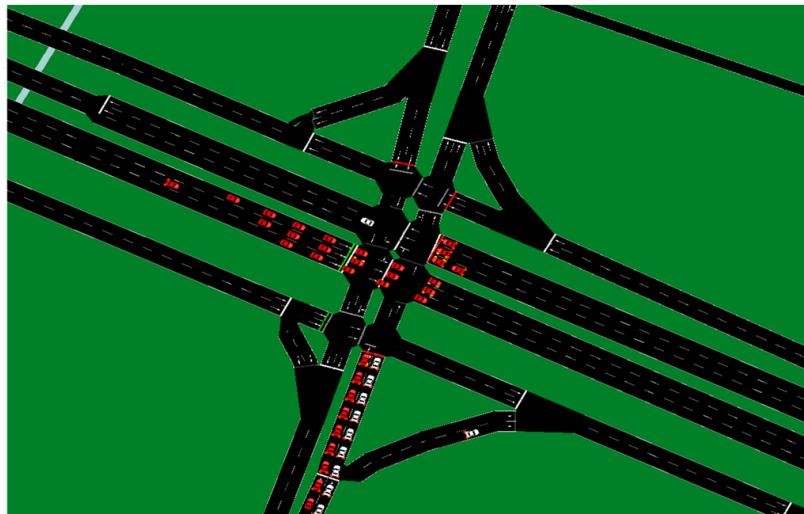


Figura 24: Captura de pantalla del simulador **SUMO**. Además de software de simulación propiamente dicho, **SUMO** provee de una interfaz gráfica que permite una visualización general, de zonas y de elementos en concreto a la vez que permite la variación de configuración de la simulación durante el desarrollo de la misma. **TODO!** Meter una imagen de nuestras simulaciones cuando estén, a poder ser sin color. Hacer el alto más pequeño

Entorno seleccionado: **SUMO**

En definitiva, el simulador que más se adapta a nuestras necesidades y el que se usará como simulador base en el desarrollo de esta tesis será **SUMO** [Krajzewicz et al., 2002, Behrisch et al., 2011, Krajzewicz et al., 2012].

SUMO es un entorno de microsimulación licenciado bajo la **GPL** versión 3,0 y desarrollado por el instituto de sistemas de transporte del Centro Aeroespacial Alemán. Implementa un modelo discreto en el tiempo y continuo en el espacio.

Además de simulación clásica, incorpora una interfaz gráfica (se puede ver una captura de la vista gráfica en la figura 24) donde se puede ver el comportamiento de cada vehículo durante la simulación.

Es interesante para obtener de un vistazo información acerca del funcionamiento del modelo en concreto a controlar. Otras de las características que el simulador ofrece son las siguientes:

- Granularidad micro y meso.
- Multimodalidad permitiendo modelar no sólo tráfico de vehículos sino de peatones, bicicletas, trenes e incluso de barcos.
- Simulación con y sin colisiones de vehículos.
- Diferentes tipologías de vehículos y de carreteras, cada una con diferentes carriles y éstas con diferentes subdivisiones de subcarriles (diseño conceptual para permitir modelar comportamientos en vehículos como motocicletas y similares).

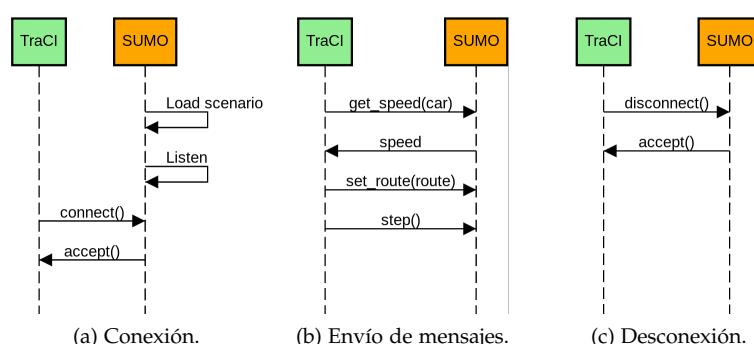
SUMO usa como modelo por defecto de *car-following* el modelo de Stefan Krauß [Jin et al., 2016], debido a su simplicidad y su velocidad de ejecución y como modelo de cambio de carril el modelo de Gipps [Krajzewicz et al., 2002]. No obstante, se encuentran paraseleccionar otros modelos como el **Intelligent Driver Model (IDM)** *Intelligent Driver Model*, el modelo de tres fases de Kerner [Kerner et al., 2008] y el modelo de Wiedemann [Wiedemann, 1974].

Al estar licenciado bajo la licencia **GPL**, su distribución implica a su vez la distribución de su código fuente. Esto permite la modificación de su comportamiento y el desarrollo de nuevos modelos integrados dentro del simulador. Sin embargo nosotros no haremos uso de esta característica, sino que usaremos **SUMO** como aplicación servidor y el módulo **TraCI** como aplicación cliente desde donde gestionar todos los aspectos de la simulación.

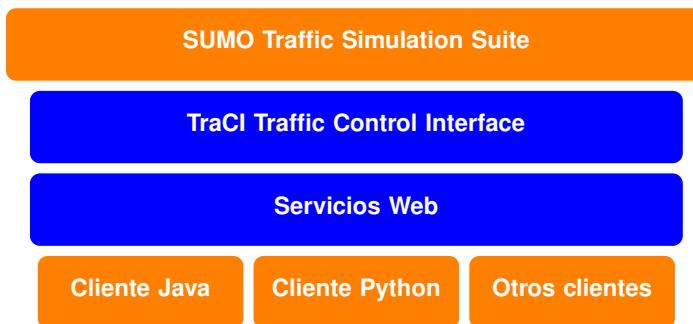
La interfaz **TraCI**

TraCI [Wegener et al., 2008] es tanto el nombre del protocolo de comunicación expuesto por **SUMO** en su versión servidor como el nombre de la librería escrita en Python para interactuar con el mismo.

Figura 25: **SUMO** ofrece la posibilidad de interactuar con la simulación desde cualquier aplicación a través del uso del protocolo **TraCI**. En la figura podemos ver, de izquierda a derecha, ejemplos de comunicación a través de la interfaz como el *handshake* o inicialización, mensajes de obtención de información y modificación de la misma más una solicitud de avance de paso en la simulación y una señal de finalización de simulación y desconexión.



Como protocolo, la interacción a través de cliente/servidor comienza especificando a **SUMO** que se desea trabajar de este modo. En ese



momento, **SUMO** se inicializa en modo servidor dejando abierto un puerto TCP para la conexión del cliente (figura 25 (a)).

Una vez el servidor se encuentra en ese estado, el cliente se conecta enviando una señal de conexión indicando que él se encargará de controlar la simulación. Desde ese momento y hasta que el cliente no envíe una señal de desconexión (figura 25 (c)), el cliente podrá enviar y recibir todos los mensajes que desee para capturar información y modificar los detalles de la simulación, incluido el mensaje *step*, que es el encargado de avanzar un paso en la simulación (figura 25 (b)).

Como librería, **TraCI** es un módulo desarrollado en Python 2.7. Aunque es posible trabajar directamente con el protocolo de comunicación a través de sockets, una librería abstrae todos los detalles dando una interfaz de trabajo más clara y sencilla. Por ello, aunque no se usarán en la tesis, existen otras dos implementaciones que merece la pena mencionar:

- **TraCI4J**¹⁸. El homólogo de la librería de abstracción de Python pero para el lenguaje Java. Está desarrollada por un tercero.
- **TrasS**¹⁹. Una plataforma ofrecida como SaaS que proporciona una interfaz de servicios web bajo protocolo SOAP para abstraer el protocolo en mensajes HTTP (figura 26).

Figura 26: Concepto arquitectural de la plataforma **TraaS**. La plataforma se conecta como cliente a **SUMO** y ofrece un API basado en SOAP de mensajes que traduce en mensajes del protocolo **TraCI**, lo que independiza completamente la elección de lenguaje de programación a la vez que abstrae los detalles del protocolo de comunicación.

¹⁸ <https://github.com/egueli/TraCI4J>.

¹⁹ <http://traas.sourceforge.net/cms/>.

Modelos de comportamiento

El objetivo que persigue la simulación de tráfico es hacer cada vez más realistas los modelos generados. Cuando el simulador está basado en Sistemas Multiagente, el realismo aumenta cuanto más se parece el comportamiento de los agentes al de los conductores reales.

Conducir implica la ejecución de múltiples tareas en paralelo, cada una de ellas pertenecientes a un nivel cognitivo. Además, las acciones no están limitadas a la interacción con el vehículo; el conductor ha de tener en cuenta otros factores como pueden ser señales, peatones o **Sistemas Avanzados de ayuda a la Conducción (ADASs, Advanced Driver Assistance Systems)**.

El modelo de [Michon, 1985] define tres niveles de abstracción para las tareas que requieren procesos cognitivo, entre ellas la tarea de conducir: el nivel de **control**, donde se encontrarían las tareas de más bajo nivel como el mantenimiento de la velocidad o los cambios de marcha, el **táctico** donde se engloban tareas encargadas de mantener la interacción con el entorno como los cambios de carril y el **estratégico**, al que pertenecen tareas de más alto nivel como el razonamiento y la planificación de rutas (ver figura 27).

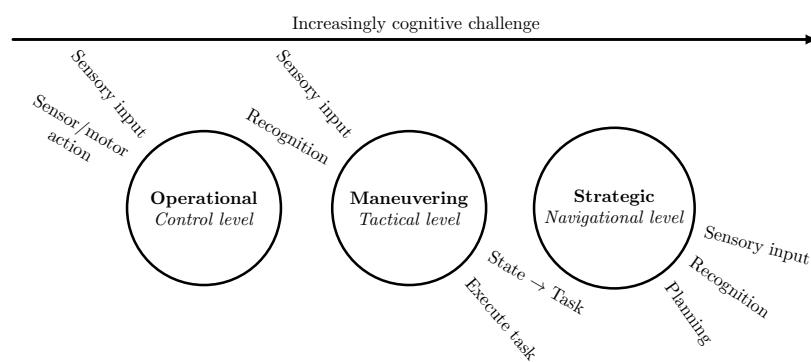


Figura 27: Los tres niveles jerárquicos que describen la tarea de conducción según [Michon, 1985]: *estrategia* (i.e. las decisiones generales), la *maniobra* (i.e. decisiones durante la conducción de más corto plazo) y *control* (i.e. automatismos).

A pesar de que se trata de una clasificación subjetiva, es un modelo ampliamente aceptado dentro del área de los **ITS**. Algunos estudios llegan incluso a definir intervalos de tiempo de razonamiento para las tareas de cada nivel, como por ejemplo [Alexiadis et al., 2004], donde se llegan a proponer tiempos que separan unos niveles de otros: alrededor de 30 s para las tareas del nivel de planificación, entre 5 s a 30 s para las tareas de nivel táctico y por debajo de los 5 s para las

tareas de control.

Otro modelo jerárquico de tres niveles muy referenciado en la literatura es el *skill-rule-knowledge* de [Rasmussen, 1986], una generalización del modelo propuesto por [Michon, 1985] al comportamiento y razonamiento humano. Postula que éste se puede basar en **habilidades** (actividades completamente automatizadas de la forma *percepción → ejecución*), **reglas** (situaciones familiares o estereotipadas de la forma *percepción → reconocimiento de la situación → planificación → ejecución*) y **conocimiento** (actividades conscientes que implican resolución de problemas y toma de decisiones de la forma *percepción → reconocimiento de la situación → toma de decisión → planificación de la ejecución*) que suelen ser necesarias en situaciones poco familiares). Algunos trabajos se basan en este modelo en lugar del de [Michon, 1985] como el presentado por [Chaib-draa and Levesque, 1994] donde abarca los tres niveles de abstracción representando en un escenario urbano tres tipos diferentes de situaciones: rutinaria, familiar y no familiar.

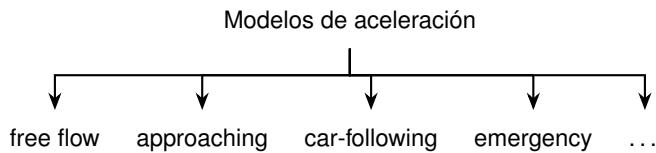
El comportamiento de un conductor al volante tiene una relación directa con el nivel de abstracción táctico. Se puede entender como el nivel encargado de planificar acciones a corto plazo para conseguir objetivos a corto plazo. Las tareas de control son automáticas e influyen poco o nada en la toma de decisiones de tareas como *cuánto acelerar en esta situación* o *cuándo cambiar de carril*. Las tareas estratégicas están a un nivel más alto de abstracción (e.g. la ruta a seguir hasta mi destino) y tampoco afectan demasiado al comportamiento en situaciones concretas ²⁰.

²⁰ No obstante algunos trabajos han demostrado que en ocasiones la planificación de la ruta sí afecta a decisiones normalmente asociadas al nivel táctico como por ejemplo la preferencia de un conductor por uno u otro carril de la vía [Wei et al., 2000, Toledo et al., 2003].

NUESTRO INTERÉS ES EL USO DE AGENTES como unidades en simulación. Después de todo trabajamos con Sistemas Multiagente. Sin embargo, y aunque en los trabajos más modernos exista una cierta predisposición hacia este paradigma, no todos los trabajos se basan en él. El auge de su uso coincide con el renacimiento de la Inteligencia Artificial, alrededor de los años 90, y los modelos que se describen en este apartado, sobre todo posteriores a esta fecha, se basan en este tipo de sistemas.

Las tipologías de agentes utilizadas es de todo tipo. Existen desde trabajos que explotan las características de los agentes reactivos (e.g. agentes que continuamente van realizando decisiones de control para mantenerse en la vía [Ehlert and Rothkrantz, 2001]) hasta aquellos que proponen complejos frameworks para definir comportamientos (e.g. cuatro unidades de funcionamiento interconectadas, *percepción, emoción, toma de decisiones y ejecución*, que gestionan el comportamiento inteligente del agente en cada situación [Al-Shihabi and Mourant, 2001]).

La información que manejan estos agentes dentro de los simuladores suele limitarse a tipologías de vehículo (e.g. utilitarios o vehículos de grandes dimensiones) y magnitudes físicas (tamaño, velocidad má-



xima). Dependiendo del trabajo, algunos autores añaden más conocimiento a los agentes; por ejemplo, en [Hidas, 2002], el autor incluye en los agentes, denominados **Driver-Vehicle Objects (DVOs)** (otra denominación del concepto de **DVU**), información adicional sobre el tipo de conductor y el nivel de conocimiento de la red de carreteras.

El resto del capítulo introducirá los modelos de comportamiento más conocidos y hará especial hincapié en el estado más reciente de modelos basados en técnicas de la Inteligencia Computacional.

Comportamientos modelados

Las tareas que se realizan en el nivel táctico del comportamiento son aquellas orientadas a circular dentro del flujo de tráfico interactuando con éste. En la literatura, estas tareas se centran en dos clases generales de problema diferentes (figuras 29 y 28): el de la **aceleración** y el del **cambio de carril**.

Los modelos de **aceleración** se ocupan de gestionar las alteraciones de la tasa de aceleración (positiva o negativa) en un entorno lineal como lo es un carril de tráfico.

El tráfico real, sin embargo, no está compuesto por un sólo carril, sino por varios. Los modelos de **cambio de carril** (figure ??) tienen como objetivo identificar cuándo el conductor desea cambiar de carril y realizar dicho cambio, ya sea porque quiere mejorar su circulación (e.g. quiere realizar un adelantamiento) o porque su ruta lo requiere (e.g. está próxima la rampa de salida que quiere tomar en una autopista).

En la literatura los modelos de aceleración han sido mucho más estudiados que los de cambio de carril, entre otras cosas por la dificultad en la captura de los datos y, por tanto, por su escasez.

El estudio del comportamiento en los cambios de carril es muy interesante debido a que tiene efectos opuestos según la carga de tráfico de la vía en la que se ejecutan. Por un lado, si la carga

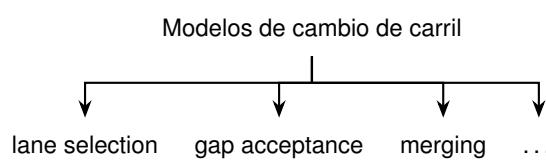
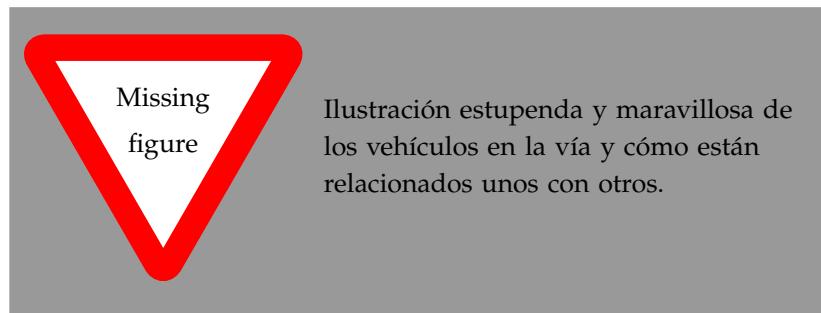


Figura 28: Tras la aparición de los modelos psico-físicos se comprobó que los umbrales en las percepciones y por tanto el comportamiento podía variar dependiendo de las situaciones. Por ello, el *car-following* no era más que una entre diferentes clases o regímenes de aceleración. Algunos de los regímenes más usados en la literatura son *free-flow*, *car-following*, *approaching*, y *emergency*, aunque algunos autores definen nuevos regímenes, cada uno con sus límites de aplicación.

Figura 29: El cambio de carril se divide tradicionalmente en una operación que involucra dos pasos. La selección de carril (*lane-selection*) al que cambiarse y la ejecución del cambio (*merging*). En la operación de *merging* se suele involucrar otra operación denominada *gap-acceptance*, aunque algunos autores la tratan como operación independiente. Otros autores pueden llegar a añadir operaciones más especializadas.

Figura 30: Representación de los vehículos en una vía junto con la nomenclatura a usar durante el resto de la tesis.



de tráfico es de ligera, mejora la velocidad media del flujo de la vía. Sin embargo, según aumenta la carga de tráfico, los cambios de carril comienzan a afectar a éste en formas de ondas de choque ([[Sasoh and Ohara, 2002](#), [Jin, 2006](#)]) e interferir incluso más que los modelos de *car-following* ([[Laval and Daganzo, 2006](#)]).

Nomenclatura

Para no llevar a equívocos, en la figura 30 se ilustran los actores típicos en una situación de tráfico junto con los nombres y su rol. A continuación los explicamos:

- **Lag car.**
- **Lead car.**

Modelado de conductores clásico

El **Modelo GHR**, presentado en [[Chandler et al., 1958](#)], es el modelo más conocido antes de la introducción del modelo de Gipps. Desarrollado a finales de los años 50 dentro de la *General Motors* (por ello también se le conoce como modelo **Generalized Model (GM)**), se caracteriza por el uso del concepto *estímulo → respuesta*, donde la respuesta del vehículo (el cambio en la tasa de aceleración) es debida a la activación de un estímulo (la variación en la distancia con el vehículo delantero) tras pasar un tiempo de retardo τ . Concretamente calcula el valor de la aceleración a en un instante t como:

$$a(t) = cv^m(t) \frac{\Delta v(t - \tau)}{\Delta x^l(t - \tau)} \quad (2)$$

Siendo t es el instante actual, $a(t)$ la aceleración del vehículo, $\delta v(t)$ y $\delta x(t)$ son la velocidad y distancia relativas al siguiente coche respectivamente, v la velocidad del vehículo y c, m, l y τ constantes, siendo ésta última el tiempo de reacción del conductor.

Los primeros trabajos sobre modelos de conducción datan de comienzo de los años 50 con estudios sobre el concepto denominado **car-following**, acuñado por [[Reuschel, 1950](#)]. Un vehículo está en una situación car-following cuando su velocidad está condicionada por el vehículo que se encuentra frente a él. En el primer modelo concreto ([[Pipes, 1953](#)]), el comportamiento responde a tratar de mantener un espacio variable en función de la velocidad.

Este modelo se puede considerar de una clase que denominaremos **mantenimiento de medida** dado que su objetivo es mantener constantemente una distancia segura, determinada a partir de la ecuación de la velocidad cuando el tiempo no baje de 1,02 segundos. Otros trabajos trabajan con el mantenimiento de otras medidas como distancia relativa al parachoque delantero o trasero.

Más adelante, a finales de la década se presentó el modelo **Gazis-Herman-Rothery (GHR)** (ver ecuación 2), el cual sirvió como base para el desarrollo de muchos modelos posteriores. Ese modelo dio origen a una nueva clase de modelos de conducción, los de tipo **estímulo → respuesta**.

Figura 31: Evolución de los tres tipos generales de modelo de *car-following*: mantenimiento de medidas, estímulo → respuesta y psico-físicos. Con la llegada de los psico-físicos se vio que *car-following* no era más que uno de tantos regímenes distintos dentro de los

~~En realidad los~~

modelos *estímulo → respuesta* son la evolución lógica de los modelos anteriores, donde se pasa de un cálculo de velocidad en función de la distancia (u otra medida) a un sistema de control donde la variable a controlar es la aceleración en función de uno o varios estímulos de entrada, además con un retardo simulando el tiempo de reacción. Algunas modificaciones sobre el algoritmo original son la asimetría en la tasa de cambio de aceleración y deceleración [Gazis et al., 1959] o la inclusión del efecto de segundos coches delanteros [Bexelius, 1968].

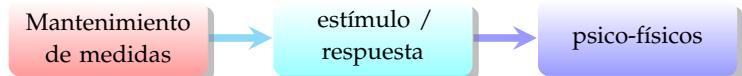
Los métodos de estas dos clases tienen un problema principal: suponen que el conductor es capaz de percibir incluso el más ínfimo cambio en las variables observadas, cuando la realidad no es así. Por ello, a mediados de los años 70 apareció una nueva clase de modelos de *car-following*, denominados posteriormente como **psicofísicos** [Wiedemann, 1974], donde se introduce el concepto de *umbral perceptual* como solución a dicha limitación. El *umbral perceptual* de una medida es el límite a partir del cual se percibe un cambio en dicha medida. Mediante su uso, las acciones de los vehículos se limitan únicamente a los cambios **perceptibles** en los coches delanteros.

A finales de la década, en 1978, se cumplió otro hito en el desarrollo de modelos de conducción. [Sparmann, 1978] define el primer modelo de cambio de carril, inspirándose en las clases de modelo psicofísico. El verdadero interés de este modelo es que sentó las bases de dos conceptos que perduran hoy en día. El primero, la diferenciación entre cambio a carriles rápidos y lentos ²¹. El segundo, la diferenciación entre la selección de carril o *lane-selection* y la ejecución del cambio o *merging*.

LA VIABILIDAD EN UN CAMBIO DE CARRIL se determina haciendo uso de modelos denominados *gap acceptance*, donde los vehículos calculan si caben o no en un determinado hueco y actúan en consecuencia.

En su origen los modelos de *gap acceptance* se desarrollaron para resolver situaciones en intersecciones e incorporaciones. En la actualidad son los modelos usados tras seleccionar el cambio de carril y antes de ejecutar físicamente el cambio, y dependiendo del modelo, es incluido como operación dentro de uno u otro.

En general se basan en una fórmula que determina si el cambio es viable o no en función de una serie de parámetros entre los que se incluye el hueco del carril destino. En la ecuación 3 se describe el modelo típico de un modelo de *gap acceptance*.



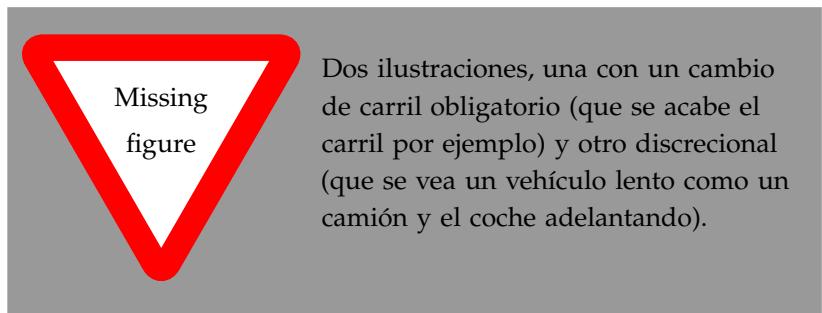
²¹ En [Sparmann, 1978] el autor entiende de la derecha como el carril lento y la izquierda como el carril rápido, y es como se entiende dentro del contexto de esta tesis. Sin embargo en otros países esta correspondencia es al revés.

El modelo típico del **gap acceptance** responde a la ecuación 3, donde en un momento t , el cambio a un carril l es viable ($f_{gl}(t) = 1$) o no ($f_{gl}(t) = 0$) dependiendo de si el espacio en el carril destino $g_l(t)$ es mayor o menor que un "hueco crítico" (en inglés critical gap) $g_l^{crit}(t)$.

$$f_{gl}(t) = \begin{cases} 0 & \text{si } g_l(t) < g_l^{crit}(t) \\ 1 & \text{si } g_l(t) \geq g_l^{crit}(t) \end{cases} \quad (3)$$

Por otro lado, existen autores que definen factores de influencia que modifican el modelo típico. Algunos ejemplos de factores pueden ser la velocidad absoluta del vehículo ([Gips, 1986, K. et al., 1996]), el tipo de cambio (**Mandatory Lane Change (MLC)** o **Discretionary Lane Change (DLC)**), usado en [Ahmed, 1999, Toledo et al., 2007]), la relativa con los vehículos delantero y trasero del carril destino ([Ahmed, 1999]) o incluso el peso de encontrarse o no en una situación de cooperación ([Ahmed, 1999, ?]).

Figura 32: Los cambios de carril se clasifican como aquellos necesarios para continuar con la conducción (obligatorios) y aquellos útiles para mejorar la situación de conducción (discretionales).



Dos ilustraciones, una con un cambio de carril obligatorio (que se acabe el carril por ejemplo) y otro discrecional (que se vea un vehículo lento como un camión y el coche adelantando).

CON LOS MODELOS PSICOFÍSICOS se llegó a la conclusión que no todas las situaciones eran iguales, sino que en función del entorno y el momento los umbrales podían variar, y que el *car-following* no era sino un subtipo más de una clase más amplia que se definió como *modelos de aceleración* (figura 28).

Debido a eso y a la irrupción de los modelos de cambio de carril, los posteriores modelos y *frameworks* desarrollados se componen de dos o más submodelos que responden a diferentes umbrales. Sin embargo, esto provoca que los modelos desarrollados sean más complejos ya que, cuantos más regímenes se tratan de agrupar en un mismo modelo, más aumenta el número de factores a generalizar y ajustar.

EL TRABAJO DE GIPPS es uno de los primeros modelos que agrupa varios regímenes distintos (concretamente *car-following* y *free-flow*) [Gipps, 1981]. Sin embargo consideramos más interesante su posterior trabajo, [Gipps, 1986] ya que puede considerarse como la primera solución para el cambio de carril.

Introduce el concepto de que los cambios de carril obedecen a diferentes motivaciones. Por un lado, los cambios pueden ser **obligatorios** (denominado en la literatura como *MLC*) cuando los vehículos se ven obligados a abandonar el carril que ocupan. Por otro lado, pueden ser **discretionales** cuando el cambio obedece a motivaciones más relacionadas con la mejora del confort o de la situación actual de conducción (ver figura 32).

En su modelo, Gipps propone un modelo para el cambio de carril al aproximarse a un cambio de dirección. Dicho modelo identifica tres distancias que caracterizan el comportamiento del conductor en función de cómo de lejos está dicho punto: (i) **lejos**, en el que no existe condicionamiento en la decisión de cambio de carril, (ii) **medio**, donde el conductor empieza a ignorar los cambios que dan ventaja de velocidad si no hacia carriles distanciados del de salida y (iii) **cerca** donde los vehículos deben estar en el carril de cambio de salida.

Otro concepto que incluye el modelo de Gipps y que exporta a modelos posteriores de cambio de carril es el de ampliar el número

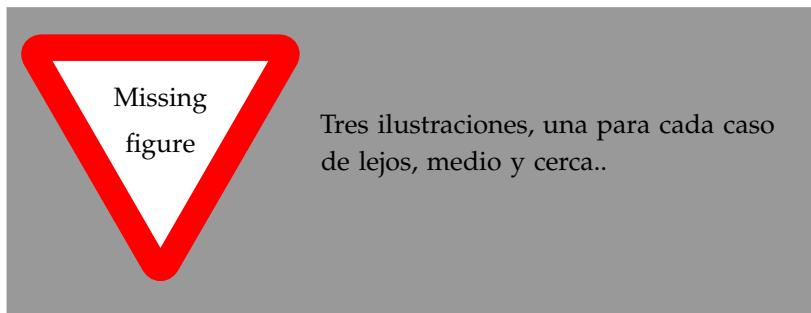


Figura 33: En el modelo de [Gipps, 1986], la distancia al punto (i.e. cerca, media distancia y lejos) determina el grado de obligatoriedad del cambio.

de *critical gaps* a más de un hueco: las distancias hasta el vehículo delantero y hasta el vehículo trasero, forzando a que durante el proceso de *gap-acceptance* las condiciones de ambos huecos tengan que ser aceptables.

Posteriormente, en [Weidemann and Reiter, 1992] se desarrolla un framework similar pero teniendo en cuenta los cambios a carriles lentos (para representar, por ejemplo, obstrucciones como accidentes o un vehículo lento) y a carriles rápidos (para situaciones como condiciones de la ruta). Además el autor incluye un modelo para influir en su desempeño en función del entorno actual (las características de los vehículos de alrededor) y el entorno potencial (la estimación de las características del entorno en momentos posteriores).

El modelo de Gipps sin embargo, sufre de dos problemas clave. Muchos de estos problemas fueron heredados por posteriores trabajos que basaban su funcionamiento en éste.

UN CAMBIO DE CARRIL NO SÓLO INVOLUCRA al conductor que lo ejecuta. En [?], el autor resalta el problema de que, en situaciones de congestión, el cambio ha de ser o bien forzado o bien a través de colaboración; en caso contrario, los vehículos no abandonarán el carril congestionado.

Uno de los primeros trabajos en abordar el comportamiento colaborativo es el de [Fritzsche and Ag, 1994]. En éste, se describe un modelo de microsimulación para analizar cuellos de botella (e.g. un accidente donde se bloquea uno de los carriles). Los autores describen el problema pero no consideran la modificación de los modelos en cambio de carril. [Yang and Koutsopoulos, 1996] sin embargo presenta un entorno de simulación (MitSIM) que introduce, entre otros, un modelo de cambio de carril en el que se habla específicamente de comportamiento colaborativo. Introducen el concepto de función de cortesía (*courtesy yielding function*) la cual afecta al modelo de *car-following* de un vehículo cuando otro intenta incorporarse al carril. Sin embargo, los detalles de dicho proceso no están especificados en el artículo.

En el trabajo de [Weidemann and Reiter, 1992] se proponen hasta cuatro clases diferentes de modelos de aceleración en función de las posiciones y velocidades relativas entre el vehículo sujeto y el siguiente: (i) **free-flow**, donde el comportamiento no se ve afectado por el del vehículo delantero, (ii) **car-following**, donde el comportamiento sí se ve influenciado por el vehículo delantero, obligando a disminuir la velocidad deseada en el conductor en cuestión, (iii) **approaching**, situación intermedia entre las dos anteriores y (iv) **emergency**, donde la situación es crítica (e.g. colisión inminente) obligando, normalmente, a respuestas extremas.

Esto es sólo un ejemplo, y diferentes autores identifican diferentes situaciones dependiendo del alcance del trabajo como por ejemplo el *close-following* o el *stop-and-go* [Toledo et al., 2003, Liu and Li, 2013].

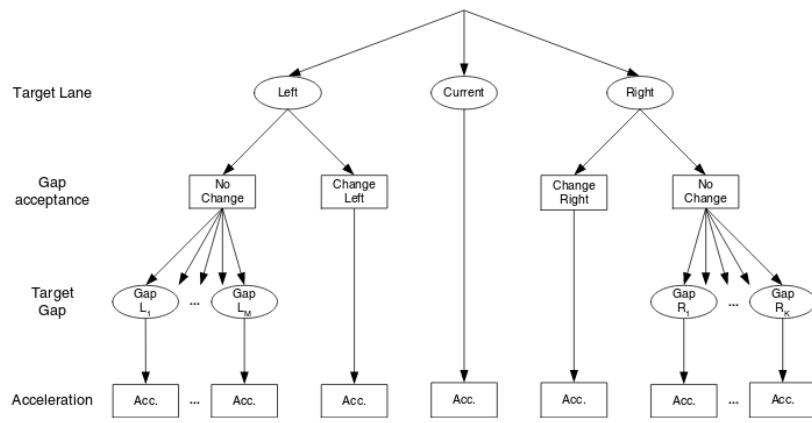


Figura 34: Estructura del modelo de comportamiento de los vehículos propuesto por [Toledo et al., 2007]. Este modelo se basa en el concepto de “objetivo a corto plazo” para elaborar un “plan a corto plazo” apoyándose, para ello, en un árbol de decisión. Aunque mantiene la clasificación, es probabilístico y existe opción de realizar un **DLC** en lugar de un **MLC** aún en situaciones donde acciones de ambas clases se activen. Para ello implementa agentes basados en utilidad donde ésta se calcula teniendo en cuenta cada uno de los nodos en un árbol de decisión. Fuente: [Toledo et al., 2007].

ADEMÁS, AL USAR ÁRBOLES SECUENCIALES, los factores son evaluados uno detrás de otro hasta encontrar una situación favorable, en cuyo caso el resto de factores no son evaluados. Por ejemplo, tal y como está formado el modelo de Gibbs, y algunos basados en éste como [?], un **MLC** inhibe cualquier posibilidad de realizar un **DLC** independientemente de la utilidad de ambos. Para evitar esta limitación, algunos autores hacen uso de técnicas como modelos probabilísticos. Ejemplos de avances en esta línea de trabajo pueden ser [Toledo et al., 2003, Toledo et al., 2007, Wei et al., 2000] (figura 34).

Modelado basado en Inteligencia Computacional

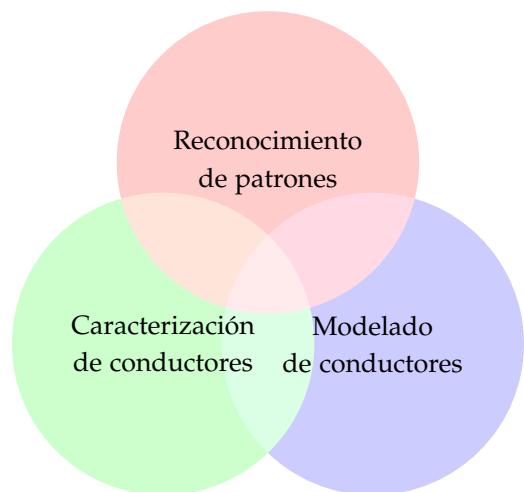
Desde mediados de los años 90 empezó a crecer el interés por aplicar la Inteligencia Artificial a los **ITS**. Hay dos razones por las que esto es así: la primera, los éxitos cosechados por la Inteligencia Computacional, los cuales atrajeron a investigadores de multitud de áreas incluida ésta. La segunda, el rápido desarrollo de la tecnología ²², que posibilita la existencia de conjuntos de datos masivos con la capacidad de explotarlos y aprender de ellos.

Algunos autores ([Zhang et al., 2011]) se atreven a afirmar incluso que el futuro de las **ITS** son las técnicas de la **CI**, y que los resultados que se puedan cosechar de técnicas basadas en el desarrollo convencional de sistemas es marginal comparado con las que se obtendrán con el nuevo enfoque.

Dentro de las **ITS**, las áreas de aplicación de la **CI** se centran en los siguientes conceptos: **reconocimiento de patrones**, **caracterización de conductores** y **modelado de conductores**. Es necesario mencionar que aunque se trate de áreas distintas, éstas suelen retroalimentarse, y es difícil encontrar estudios que se centren en una única área sin tocar el resto (figura 35). Por ello, aunque nuestro interés pueda centrarse en el modelado de conductores, es necesario conocer el estado de las demás áreas.

²² La tecnología es cada vez más barata, más precisa y con más funcionalidades. En la última década hemos vivido una explosión de dispositivos de todo tipo: teléfonos y relojes con GPS, acelerómetros y giroscopio, ordenadores completamente funcionales del tamaño de una moneda, sensores RADAR y LIDAR para uso amateur además de profesional y así un largo etcétera.

Figura 35: Las principales áreas de aplicación de la Inteligencia Computacional en los ITS son el reconocimiento de patrones, la caracterización de conductores y la modelización de los mismos. Aunque son áreas de aplicación distintas, los estudios en general tienden a solaparse. Por ejemplo, las técnicas de reconocimiento de patrones pueden usarse como forma de extracción de características para una caracterización de conductores y a su vez esta caracterización puede usarse como base para su modelado.



- En el **reconocimiento de patrones** las técnicas suelen trabajar en los temas de extracción de características y de predicción de comportamientos. La cantidad de datos que se pueden generar en un coche es tal que el trabajo a través de información en curdo es inviable (no digamos ya cuando los datos son extraídos de una flota de vehículos).
- La **caracterización de conductores** es interesante debido a que permite la identificación de perfiles de conducción y su clasificación de acuerdo a indicadores extraídos de su manera de conducir.
- El **modelado de conductores** nos permite la reproducción de comportamientos en simulación.

Las técnicas de **CI** más utilizadas en el modelado de conductores son las Redes Neuronales Artificiales y la Logica Difusa. Es comprensible dado que las primeras son una de las técnicas principales en la rama del Aprendizaje Automático, y la segunda por ser una manera sencilla y cercana a la manera de razonar del ser humano.

Sólo por la propia naturaleza de las técnicas, los modelos incluyen siempre más de un único comportamiento: al ser entrenados con datos de conductores reales y manejar la información con incertidumbre, “aprenden” a comportarse en diferentes regímenes (e.g. *free-flow*).

LAS REDES NEURONALES ARTIFICIALES fueron el punto de entrada de la **CI** en la modelización de conductores. Los perceptrones multicapa y sus nuevas técnicas de entrenamiento se habían convertido en la nueva solución universal para todo aquel problema del que se dispusiese de datos.

Las **ANN** se han aplicado mucho sobre el campo de los **ITSs** en general, no sólo en modelado de conductores sino en prácticamente todos los aspectos como la clasificación de conductores [Díaz Álvarez et al., 2014], la conducción autónoma [Huval et al., 2015]

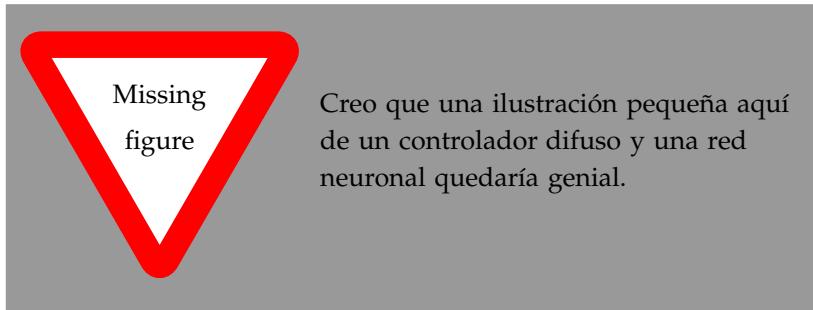


Figura 36: Al trabajar con métodos como las Redes Neuronales Artificiales o la Logica Difusa, la inexactitud y la incertidumbre son ciudadanos de primera clase y forman parte de los modelos.

o la predicción ([Dougherty et al., 1993, Chan et al., 2012] entre muchos otros.

El primer trabajo de la literatura sobre la aplicación de **ANNs** al modelado de conductores es [Fix and Armstrong, 1990], donde los autores desarrollan un controlador para imitar el comportamiento de un conductor.

En la misma década, en [Hunt and Lyons, 1994] se desarrolló uso de **ANN** aplicadas al entorno del vehículo para identificar el entorno y determinar cuándo y cómo realiza el conductor un cambio de carril.

Los modelos hasta el momento hacen uso de datos extraídos de conductores reales pero desde entorno de simulación. El primer trabajo en usar datos reales de vehículos instrumentados para este cometido es [Jia et al., 2003]. A partir de las entradas correspondientes a velocidad relativa, espacio relativo, velocidad y velocidad deseada (para ello, clasifican al conductor de agresivo, normal, conservador) determinan la aceleración/deceleración del vehículo. No lo aplican a ningún simulador, sólo que los valores se ajustan. Otros trabajos similares son [Panwai and Dia, 2007, Khodayari et al., 2012].

El trabajo de [Simonelli et al., 2009] también se apoya en datos extraídos de entornos reales. El interés de este estudio radica en que es el primero en realizar una comparativa entre el desempeño de una arquitectura *fast-forward* (e.g. perceptrón multicapa) frente a una recurrente (e.g. red de Elman). El por qué del uso de redes recurrentes es por su capacidad de reconocer patrones dinámicos, los cuales son de esperar en este tipo de comportamientos.

EL PRIMER USO DE LA LOGICA DIFUSA fue contemporáneo al de las **ANN**, y también aplicada a un modelo de *car-following*. Después de todo los modelos psicofísicos aparecieron debido a que la percepción del conductor no es absoluta, sino imprecisa.

Estos modelos parten de la hipótesis de que la información que maneja el conductor a la hora de tomar decisiones proviene de un análisis no demasiado detallado de la situación que le rodea; es decir, la percepción y el comportamiento humanos son estímulos percibidos de manera aproximada. Por tanto, el resultado debe ser fruto de un

proceso de razonamiento que tenga en cuenta esa imprecisión en los estímulos, y la lógica difusa es ideal para modelar la incertidumbre del mundo real y por tanto de las percepciones de los conductores.

[Kikuchi and Chakroborty, 1992] es el primer trabajo documentando sobre el tema. En él, los autores aplicaron la lógica difusa sobre un modelo de aceleración de tipo *car-following*. Utilizaron el modelo **GHR** (Ecuación 2) como base y determinaron las entradas al modelo como valores de pertenencia a conjuntos difusos. Las entradas del modelo eran las distancia y velocidad relativa entre el vehículo modelado y el delantero y la variación en la aceleración del vehículo delantero. Como salida, el cambio en la tasa de aceleración sobre el vehículo modelado.

[McDonald et al., 1997, Wu et al., 2003] desarrollaron del modelo **Fuzzy L0gic motorWay SIMulation (FLOWSIM)** con similares características pero incluyendo el comportamiento de cambio de carril, además en dos categorías: al carril **lento** (principalmente para evitar incordiar a los vehículos que se aproximan por detrás a velocidades superiores, usan dos variables, presión del vehículo trasero y satisfacción en el gap del carril destino) y al **rápido** (para ganar velocidad, variables: velocidad ganada con el cambio y oportunidad, es decir, seguridad y confort con el cambio).

Los modelos hasta el momento se basaban en conjuntos difusos y reglas definidos *ad-hoc*. En [Chakroborty and Kikuchi, 2003] se introduce el concepto de personalización, ajustando los conjuntos difusos de las variables de entrada y salida a valores extraídos de conductores reales. Este ajuste se realiza mediante la representación del controlador difuso como red neuronal y posterior ajuste mediante entrenamiento de dicha red (*back-propagation*). A este trabajo le siguen muchas otras aproximaciones *neuro-fuzzy* como [Ma, 2004, Zheng and McDonald, 2005]

Más adelante, en [Das and Bowles, 1999], dentro de su simulador **Autonomous Agent SIMulation Package (AASIM)** añaden los conceptos de **MLC** y **DLC** a los comportamientos de cambios de carril. En **MLC** las reglas tienen en cuenta la distancia al siguiente punto característico (e.g. una salida) y el número de cambios de carril necesarios. En **DLC** deciden si cambiar o no basándose en el nivel de satisfacción del conductor y en el nivel de congestión en los carriles adjacentes ²³.

LAS REDES NEURONALES ARTIFICIALES Y LA LOGICA DIFUSA NO SON las únicas técnicas usadas para determinar comportamientos en conductores.

Por ejemplo, en [Maye et al., 2011] se presenta un modelo no supervisado, online y basado en redes bayesianas donde se infiere el comportamiento del conductor haciendo uso de una **Intertial Measurement Unit (IMU)** y una cámara. De la **IMU** se extraen datos que se separan en fragmentos para luego relacionarlos con las imágenes obtenidas.

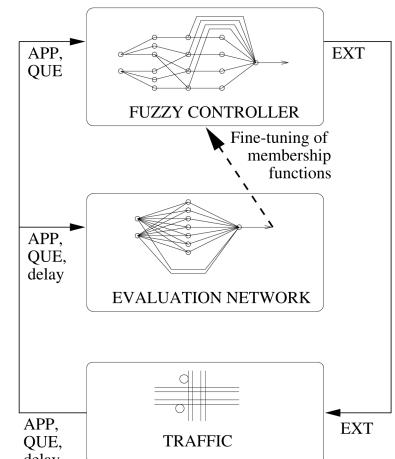


Figura 37: Un ejemplo de aproximación *neuro-fuzzy*, en este caso aplicado al control de señales de tráfico. El Sistema de Control Difuso se implementa como una Red Neuronal Artificial de tipo *feed-forward* en lugar de con una representación tradicional. Además, el sistema completo lleva integrado un subsistema basado en también en Redes Neuronales Artificiales *feed-forward* que ajusta las funciones de pertenencia a través de entrenamiento por refuerzo. Fuente: [Bingham, 2001]

²³ Este trabajo se apoya en trabajos anteriores que separan las jerarquías en situaciones de **MLC** y **DLC** como [Yang and Koutsopoulos, 1996, Halati et al., 1997, ?]. Estos trabajos, no obstante, no pertenecen al área de la **CI**.

das de la cámara. Otro trabajo similar pueden ser [Van Ly et al., 2013] el cual se apoya también en la segmentación de los datos extraídos de una **IMU** pero con técnicas distintas (concretamente *clustering* basado en **Support Vector Machine (SVM)** y en *k-medias*) y sin cámara.

En [Bando et al., 2013] describen otro modelo no supervisado, éste offline, basado en un modelo bayesiano no paramétrico para la clusterización, combinándolo con un modelos supervisado (**Latent Dirichlet Allocation (LDA)**) para la clusterización a más alto nivel. El trabajo de [Bender et al., 2015] usa una aproximación similar pero sin la segunda clusterización.

Otra aproximación es el de los **Modelos Ocultos de Markov (HMMs, Hidden Markov Models)**. Por ejemplo, los trabajos [Kuge et al., 2000, Sekizawa et al., 2007] aplican entrenamiento supervisado sobre estos modelos para reconocer los eventos que están provocando los conductores (concretamente cambiando de un carril a otro). En [Hou et al., 2011] van un paso más allá desarrollando un modelo capaz de estimar si el conductor va a realizar un cambio a la derecha o a la izquierda a partir del ángulo de giro del volante (con una precisión de 0,95 en una ventana temporal de 1,5 segundos y de 0,83 en una ventana de 5 segundos).

Por último, [Aghabayk et al., 2013] presenta un modelo basado en **LOcal LInear MOdel Tree (LOLIMOT)** que son similares a una aproximación *neuro-fuzzy* del comportamiento. Intenta incorporar imperfecciones perceptuales en un modelo de *car-following*.

ESTAS TÉCNICAS NO SÓLO SE USAN PARA modelar comportamientos complejos o modelos enteros. Algunos trabajos se ocupan de características o aspectos de un modelo concreto. Por ejemplo, los trabajos [Hatipkarasulu, 2003, Zheng et al., 2013] se ocupan exclusivamente del cálculo de tiempo de respuesta del conductor en modelos de *car-following*, el primero con controladores difusos y el segundo con **ANNs**.

Papers a añadir o al menos relevantes

Analysis of Recurrent Neural Networks for Probabilistic Modeling of Driver Behavior (de Morton J, Wheeler T, Kochenderfer M). Explora redes recurrentes para estudiar modelos de aceleración. Review of microscopic lane-changing models and future research opportunities. Su propio nombre dice de qué va. A review of intelligent driving style analysis systems and related artificial intelligence algorithms. Este lo mismo sirve para completar algún hueco que haya quedado en el estado del arte. Es relativamente reciente.

Modelado de conductores: defición y diseño

Definición del problema

Modelo de comportamiento de conductor

Introducción

TODO! Hay que dejar claro que este capítulo es el de la metodología. Si al final no queda claro, le cambiamos el nombre a metodología, pero me gusta más como “odelo de comportamiento de conductor”

Captura de datos

TODO! Lo siguiente son cosas que estoy pensando

Comentarios sobre las variables que estoy valorando introducir. Cuando se habla del carril l , o se refiere al carril en el que se encuentra el vehículo, -1 al carril de la derecha y $+1$ al carril de la izquierda.

- **Nube de puntos.** Esta tiene que venir sí o sí porque además ya la usamos en la ejecución de cambio de carril.
- **Velocidad del vehículo.**
- **Máxima velocidad del carril 1.** Ésta la tengo que añadir yo posteriormente. Para el simulador afortunadamente me viene dada.
- **Distancia al vehículo delantero.** Creo que esta me la puedo quitar porque tengo la información del lidar.
- **Diferencia de velocidad con el vehículo delantero.** Lo mismo que con la distancia. Al meter varios frames sucesivos temporalmente como entrada en el modelo, tenemos la diferencia de posición que viene a ser la velocidad.
- **Dead end l .** La máxima distancia que se puede seguir en el carril l antes de que no se pueda circular más. Es o si no hay carril, de esta manera nos quitamos de un plumazo el tener una variable para los carriles.
- **Siguiente salida.** Distancia al siguiente cambio de sentido (esto es, que no se tiene que girar a la derecha).

- **Siguiente salida.** Sentido de giro en la siguiente salida (1 o -1 dependiendo de si es izquierda o derecha).
- **Distancia a siguiente semáforo.** Distancia al siguiente semáforo.
- **Estado del siguiente semáforo.** Pues eso, si está en rojo, amarillo o verde.
- **Siguiente estado del siguiente semáforo.** El estado que toca después. Lo pongo porque creo que puede influir en el comportamiento del conductor cuando se aproxima a un semáforo en verde que se acaba de poner en amarillo.

Instrumentación del vehículo

Descripción de datos a extraer

Selección de rutas

Selección de sujetos

Preparación de los datos

Entrenamiento de modelos

Perspectiva general

Comportamiento longitudinal

Free flow

Car following

Lane exit

Comportamiento en cambio de carril

Toma de decisión de cambio de carril

Ejecución de cambio de carril

Validación del modelo

TODO! Esto lo saco del paper de modelling blablabla de lane execution. Lo pongo aquí porque lcreo que lo suyo es poner la metodologías entera aquí. Además el tema de instrumentación, los recorridos, etcétera esta bien que estén aquí englobados. Hay que tener cuidado de, si se pone una cronología, que sea cierta, es decir, primero lane execution, luego lane intention, ...

Para entrenar, comparar y validar los diferentes modelos de aceptación de cambio de carril, se ha seguido el siguiente método. Se utiliza un vehículo instrumentado para registrar los datos de conducción, primero, para el reconocimiento de patrones de conducción para clasificar los controladores en dos subconjuntos y segundo, para la construcción de conjuntos de datos para el proceso de formación de modelos. En el segundo caso, un operador estará presente en el vehículo y le pedirá a los sujetos que realicen un cambio a la izquierda o a la derecha en diferentes situaciones (principalmente con y sin automóviles en el entorno) mientras se graban los datos. Esos eventos luego se registran como intención de cambio de carril porque los sujetos deben ejecutar el cambio de carril si es posible. De esta forma, garantizamos que cada ejecución de cambio de carril (o imposibilidad de ejecución) está directamente relacionada con una intención de cambio de carril.

Las secciones que siguen describen cómo se instrumenta el vehículo, y se obtienen datos, cómo se procesan a una representación adecuada para entrenar a los modelos y, finalmente, cómo se entrena a los modelos.

Instrumentación del vehículo

El vehículo usado para el desarrollo de esta tesis es un Mitsubishi iMiEV (Figura 38). Éste ha sido instrumentado con los siguientes dispositivos:



Figura 38: The instrumented Mitsubishi iMiEV.

Cuadro 2: Valores capturados por el vehículo instrumentado y sus dominios. Los valores de 0, 1 y 2 se corresponden con los cambios de carril, siendo 0 cambio a la izquierda, 1 no cambio y 2 a la derecha.

Variable	Descripción	Domini
LiDAR	Coordenadas de todos los puntos capturados por el dispositivo.	[–200

- **LiDAR.** **TODO!**explicación de qué es un LiDAR. Algo cortito del estilo mide la distancia a través la diferencia entre la emisión de pulsos de luz y su reflejo en un sensor. A lo mejor queda bien en un sidenote. El usado es un Velodyne VLP-16, un LiDAR circular de 16 canales verticales separados 2° , dando un FOV (-15, 15) grados de apertura vertical). Este dispositivo se conecta a la máquina a través del puerto ethernet y se usa para la obtención de información del entorno del conductor. El LiDAR está situado en el techo del vehículo orientando los (0, 0) grados en al dirección y sentido de éste y con el plano horizontal paralelo al suelo.
- **Keyboard.** **TODO!**No sé si ponerlo, pero al menos habría que mencionarlo, aquí o más adelante en el apartado específico de la ejecución de cambio de carril. A lo mejor se podría disfracar como un "gestionador de eventos." algo así y poner un mando o unos botones de recreativa, que eso siempre queda bien. Un teclado común conectado a través del puerto usb para funcionar como detector de eventos.
- **CAN Bus.** **TODO!**Explicar qué es el CAN Bus, y si es necesario, un poquitín su funcionamiento (aunque sea en un sidenote, así queda todo más completito.). El BUS del vehículo se conecta a través del puerto USB al ordenador **TODO!**a lo mejor hay que explicar un poco más la conexión, el tipo de cable y tal. Es usado para la extracción de la interacción del conductor con el vehículo.
- **GPS Receiver.** **TODO!**A lo mejor este no es necesario ponerlo. Yo creo que se podría decir que se usa para obtener una segunda medida de velocidad y aceleración, así como para la localización espacial de subconjuntos de datos interesantes para su estudio.

Todos los dispositivos se conectan a un ordenador con sistema operativo Ubuntu GNU/Linux sobre Intel i7-6400U CPU con 8GB de memoria RAM. El esquema del vehículo instrumentado se detalla en la Figura 39.

Todos los dispositivos han sido, o bien configurados para una captura a 10Hz de frecuencia, o bien remuestreados a ésta. La tabla 2 resume los datos recogidos y su dominio.

Perfiles de conducción

Para tener suficiente información con la que trabajar, se han usado datos pertenecientes a conjuntos de conductores con diferentes perfiles de conducción. Todos los sujetos empleados en el experimento tiene el mismo rango de edad y género.

Figura 39: The instrumented vehicle schema.



Tras obtener sus datos en rutas de prueba preliminares en entorno urbano, los sujetos se separaron en dos conjuntos de conductores con perfiles de conducción diferentes tal y como se describe en [Díaz Álvarez et al., 2014]. Los sujetos cuyos datos eran difícilmente caracterizables no fueron incluidos en ninguno de los conjuntos.

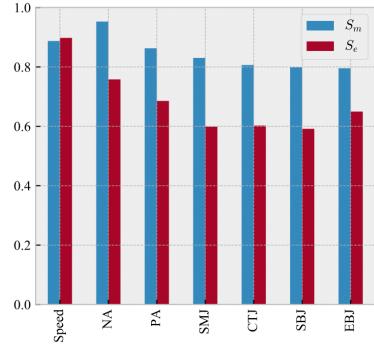
Los parámetros usados para la clasificación corresponden al promedio y las varianzas de los 7 indicadores (uno para la velocidad, dos para la aceleración y cuatro para los tirones) para un total de 14 indicadores. Los valores se han normalizado para resaltar la diferencia entre ambos perfiles de la siguiente manera:

- **Velocidad.** Media normalizada en el intervalo [0, 20] y varianza en el intervalo [0, 400].
- **Aceleración y jerk.** Media y varianza en el intervalo [0, 2].

Los valores brutos se representan en la Tabla 3. Sus perfiles se han extraído de los datos de GPS y CAN Bus como se describe en [1]. La figura muestra el perfil normalizado para los datos de los dos subconjuntos de controladores (llamados Sm y Se).

Indicador	S_m		S_e	
	μ	σ	μ	σ
Speed	17.75	349.31	17.96	307.25
Negative acceleration (NA)	1.91	4.44	1.52	1.91
Positive acceleration (PA)	1.73	3.68	1.37	1.87
Starting movement jerk (SMJ)	1.66	3.50	1.20	1.69
Cruising track jerk (CTJ)	1.61	2.34	1.21	0.99
Starting brake jerk (SBJ)	1.60	2.20	1.18	1.49
Ending brake jerk (EBJ)	1.59	2.40	1.30	2.00

Cuadro 3: Indicadores extraídos a partir de los datos de cada perfil de conducción.



(a)

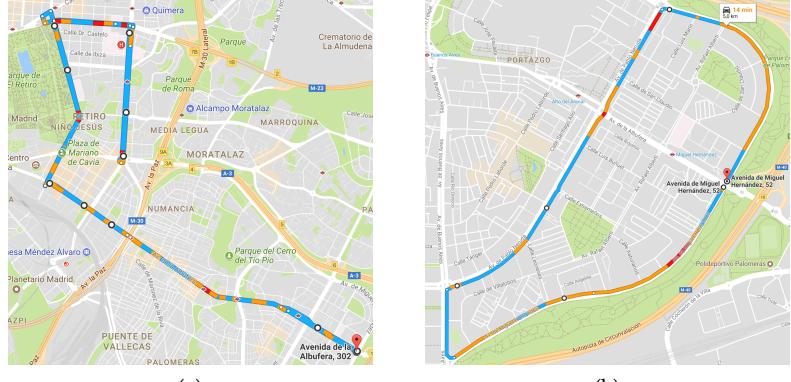


(b)

Routes

Para la captura de datos se prepararon dos rutas, R_1 para utilizar los datos extraídos como datos de entrenamiento de los modelos y R_2 como datos para el proceso de validación de los mismos. La Figura 41 muestra los mapas para éstas. Ambas son rutas urbanas con una duración de aproximadamente 20 minutos, con tramos de uno y dos carriles, y en las que las velocidades máximas permitidas oscilan entre los 20 y los 50 km h^{-1} . Fueron realizadas entre las 11:00am y las 12:00pm en días laborables, permitiendo una circulación con suficientes vehículos para conducir, pero sin demasiados como para impedir la circulación.

Figura 41: Las dos rutas del experimento, (a) Ruta R_1 para entrenamiento y (b) Ruta R_2 para validación.



Los sujetos realizaron la ruta primero para familiarizarse con el

Figura 40: Comparación de indicadores entre los diferentes perfiles de conducción.

circuito. Posteriormente, las rutas fueron realizadas y los datos extraídos para el resto del proceso.

Proceso de los datos

Tras la captura, los datos fueron preprocesados antes de entrenar los modelos. Los datos de cada modelo fueron preprocesados de manera distinta, por lo que este proceso quedará descrito más adelante en las secciones dedicadas a éstos.

Descripción de los conjuntos de datos

The datasets are built upon the sequences described above, but we need a way to feed up our models with a sense of time. For this purpose, the inputs in the datasets will be created with one or more previous rows of the sequences they include to add this temporary sense to the model in the form of a sliding temporal window.

Throughout the document we will talk about the concept moment. We will say that the moment t_i is the row of data that occurred $\frac{i}{10}$ seconds ago, and therefore t_0 is the data corresponding to the present moment. Figure 42 shows a simplified schematic outlining this concept.

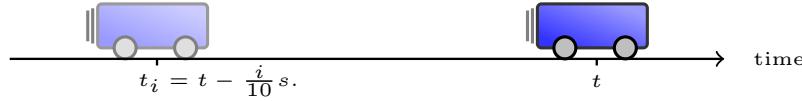


Figura 42: Given the row of data in time t , we define the moment T_i as the row of data that occurred in time $t - \frac{i}{10}$ seconds.

We will define four datasets implementing different moments. All the datasets will have the same number of outputs (3 being change left, no change and change right) but a different number of inputs according to the temporal window. Table 4 describes the datasets used during the first stage of the experiment. The datasets are named $ds_{i,t}$ and $ds_{i,v}$, correspond to the i -th dataset for the training and validation partition respectively, and all of them correspond to the profile S_m . The dataset named dsE_v is a single validation dataset and corresponds to the profile S_e . The reason of having only one validation dataset for profile S_e is due to the results obtained in stage 1 (see below in results section 5).

The choice of 5 and 10 frames (500ms and 1s according with the frequency of 10Hz) as key time points in the temporal windows is not arbitrary. Here we assume that the process of lane-change execution is a task that involves the visual cortex and the prefrontal cortex in the brain. Thus, we could assume that the response time is between 0.2s to 1.2s [Lipton et al., 2015].

All the training and validations have been executed on a compu-

Name	left	no change	right	inputs	moments	size	profile	
$ds1_t$	9804	44376	9804	4322	t_5	63984	S_m	Cuadro 4: Descripción de los conjuntos de datos utilizados en los experimentos.
$ds1_v$	410	1016	410	4322	t_5	63984	S_m	
$ds2_t$	8544	42408	8544	4322	t_{10}	63984	S_m	
$ds2_v$	312	976	312	4322	t_{10}	63984	S_m	
$ds3_t$	8544	42408	8544	6483	t_5, t_{10}	63984	S_m	
$ds3_v$	312	976	312	6483	t_5, t_{10}	63984	S_m	
$ds4_t$	6504	38800	6504	8644	t_5, t_{10}, t_{20}	63984	S_m	
$ds4_v$	144	896	144	8644	t_5, t_{10}, t_{20}	63984	S_m	
dsE_v	151	725	151	8644	t_5, t_{10}, t_{20}	63984	S_m	

ter different from the one in the vehicle using the library TensorFlow [28] from Google. The operative system is a Debian GNU/Linux 9.1 (Stretch) over an Intel(R) Core(TM) i7-6700K CPU at 4.00GHz de 16GB of RAM and a nVidia TITAN X with 12GB of GDDR5X RAM and 3585 CUDA cores at 1.53GHz. We have used the training method called Adam (Adaptive Momentum Estimator [29]), a very successful gradient descent algorithm which combines the idea of momentum with the RMSProp [30]. Adam algorithm relies in 4 parameters, β_1 , β_2 , ϵ and η . The parameter β_1 is associated to the momentum, while β_2 is the one associated with RMSProp. ϵ corresponds to a very small value to avoid divisions by zero and the last one, η , is the learning rate. In our case, we will maintain the default parameters proposed in the paper for β_1 , β_2 , ϵ and η (0.9, 0.999, 1e-08 and 0.001 respectively) and set a learning rate deduced by a trial and error process. To avoid memory exhaustion problems and follow closely the evolution of the training processes, a minibatch of 1.000 items is implemented. For the purposes of including the lane-change intention features in the CNN architectures, a modification over the general layout has been implemented. In it, the LIDAR images are presented to the network input whereas the lane-change intention features are normalized and presented to the dense layers right after all the pattern detection phase. That is the reason of the variation on the number of neurons. The intuition behind this decision is that, after all the pattern detection phase, it is expected to have some of them recognized right before the dense layers, i.e. the classification phase. We can consider lane-change intention features as already recognized patterns and thus it is not necessary to add a transformation of them into the pattern recognition phase. So, if there are N neurons specified in that layers and the dataset specifying M lane-change intention features, the dense layer is composed of N neurons. To find the optimal architecture for this purpose, the training process is separated into two stages. In the first one (stage 1), a shallow analysis was performed over a superficial study on a wide range of architectures for all the training sets. Specifically, the training process has been performed over 135 ANNs (90 MLPs and 45 CNNs as described in Error: no se encontró el origen de la referencia and Error: no se encontró el origen de la referencia respectively). The networks were trained during 10.000 epochs of 1.000 mini-batches and then validated for each

of the proposed datasets (described in Table). In the second one (stage 2), a more in-depth study over the most interesting architectures for both families is performed, improving the deficiencies detected (as shown in the results section) and training the models until the validation error is stabilized.

TODO! Creo que habría que indicar en la introducción y luego aquí que queremos reproducir nuestro modelo en SUMO, que determina los cambios de carril como teleportaciones de un carril a otro.

Broadly speaking, the lane-change problem within the cognitive scheme is associated with the tactical level (also manoeuvre level) on a three-layer scheme where the tasks of intermediate cognitive process are grouped.

Determinando la intencionalidad en el cambio de carril

Ejecucutando el cambio de carril

Methodology

In order to train, compare and validate the different lane-change acceptance models, the following method has been followed. An instrumented vehicle is used for recording driving data, firstly for driving patterns recognition to classify drivers into two subsets and, secondly, for constructing datasets for the models training process. In the second case, an operator will be present in the vehicle and will ask to the subjects to execute a left or right change in different situations (mainly with and without cars in the surroundings) while data is being recorded. Those events are then logged as lane-change intention because the subjects must execute the lane-change if possible. In this way, we guarantee that each lane-change execution (or impossibility of execution) is directly linked to a lane-change intention. The sections that follow describes the how the vehicle is instrumented, and data are obtained, how they are processed to a suitable representation to train the models and, finally, how the models are trained.

Both Lane change intention and Lane change action are variables captured through the driver's communication to an operator located in the co-pilot's seat so some sort of noise is expected in the starting and ending point on each of the sequences of lane change manoeuvres. Experimental results have shown that a disposition where the left change and right change are in opposite sides (i.e. with the value of no change in the middle) speeds up the training convergence.

Multilayer Perceptrons

In a MLP, the neurons are arranged in layers so that all the neuron

outputs on one layer are the inputs for all the neurons in the next layer. The first and last layers are called input layer and output layer respectively. The inner layers are called hidden layers. As shown in Figure 2 (b) (a two-layered MLP), in this architecture the inputs are usually presented as vectors. As they have been used extensively in several areas with great success, we use them here to make a comparison of the improvement of the use of CNNs over MLPs. Error: no se encontró el origen de la referencia depicts the final MLP architectures used in this work. Each number on the “architecture” column represents the number of neurons in each of the hidden layers. The output layers contain 3 neurons corresponding to the activation neurons for left-change, no action and right change. Also, each row represents a set of topologies, where points out the dataset employed to train this architecture (described in Table later in section 4. Methodology) and symbolizes the dropout applied to all their hidden layers. This results in a total of 90 MLP networks to work with. Name 1 Layers Architecture MLP₁–

MLP2–

MLP3–

MLP4–

MLP5–

MLP6–

¹ The names represent different networks depending on the dataset used for training and the dropout rate. Table 1. MLP networks used in this work

CNN

Three architectures, each of them with different dropout rates, have been used in the comparison and are shown in Error: no se encontró el origen de la referencia. Name 1 Layers Architecture CNN₁–

CNN2–

CNN3–

¹ The names represent different networks depending on the dataset used for training and the dropout rate. Table 2. CNN networks used in this work. Each element in the “architecture” column corresponds to a different kind of layer, being a convolution layer of C channels and size, a max-pool layer of size with a step of and a dense layer of neurons. In our case, the input layers of a convolution operation are padded with zeros to maintain the same size on the output. As with the MLP architectures, the output layer contains 3 neurons corresponding to the activation neurons for left-change, no action and right change and each row represents a set of topologies, where points to the dataset employed to train this architecture (described in Table) and symbolizes the dropout applied to all their hidden

layers. This results in a total of 45 CNN networks to work with.

Once all the data have been logged, the training and validation sets will be drawn from the intervals during the driving in which there has been a lane-change intention, regardless of whether it has been executed. Then, an exploratory training process (we call it stage 1) will be performed against the models exposed in section 3 with the different temporal versions of the sequences extracted from route by and will be validated against the validation sets extracted from route by and . After this, the best datasets and models will be chosen for a more in-deep training stage.

The data logged for both and contains several information about the environment in no-intention situations. This information is not relevant for the purposes of the study and therefore those data are removed from the data sets. Situations in roundabouts and crossroads are also discarded. The remaining data are then grouped in sequences of continuous events and treated separately since each of them are ordered sets with full temporally meaning. After that, and before creating the full training and validation datasets, two more processes are accomplished to augment the available data and making it suitable for the ANNs.

Data augmentation Complex problems require lots of data for the training process. Otherwise, they may succumb to the problem known as overfitting, where the model is capable of learning almost every example in the training data but fails in generalizing examples not seen before. Our problem seems to fit with this description. Even more, the numbers of possible lane changes executed during driving is quite small. It is therefore necessary to artificially increase the training data set (in validation sets, data augmentation makes no sense). In our case we will use the technique of symmetry or mirroring and a technique developed for this problem that we have called shaking. In the process of mirroring, we assume that the cognitive mechanisms that drive the lane change execution towards one side in a situation are the same that drives the execution to the opposite side in a mirrored situation (with respect to the plane). An example of this concept is illustrated in Figure .

(a)

(b) Figure 7. An example of the mirroring technique: (a) A regular frame captured from the lidar; (b) The same frame after the mirroring process. By augmenting data with mirroring, the number of examples is doubled with the advantage of not losing any precision in the data. In the technique we have called shaking, we take advantage of the imprecise nature of the LIDAR data in three-dimensional space. Considering an accuracy of $\pm 3\text{cm}$, we have a sphere of imprecision of 3cm of radius for each point in which we can situate it randomly. The name shaking refers as if we shake all the points of a frame (Figure). The process is as follows. For each sequence we get all the frames

and, for each of them, we go through each point and apply it a new random shift of . After this process, a new sequence is created with the same lane-change intention and action, and similar but slightly different environment to the original one.

(a)

(b) Figure 8. An example of two frames after a shaking process over the frame displayed in Figure : (a) Shaking of ; (b) Shaking of . The intuition that this solution can help reducing the variance of the problem is that we assume that high precision is not required and that, therefore, two similar point clouds will produce similar effects. Due to the LiDAR imprecision, we can play with all the 3cm spaces around each point without, in theory, lose any precision. In our case, new sequences are generated for each original and mirrored frame. For each point of the frame, its position is shifted being , and a uniform value belonging to the open interval or , depending on the stage of the training process. 4.4.2. Data representation These sequences contain, apart from the intention and execution of lane changes of the driver, the surroundings of the vehicle as a point cloud. The problem here lies in the requirement of the ANNs that need the input as a fixed set of elements, and the number of points in a point cloud varies depending on the number of elements in the surrounding where the laser impacts. For this solution a representation as a 2D deepness map is used.

In this sense, the data acquired by our laser scanner is transformed into the image domain. Thus, the distance information of each point is projected into a 2D matrix where the vertical axis corresponds to the elevation angles and the horizontal axis is the azimuth angles. The former is discretized between -12 and 3 degrees with a step of 2.5 degrees, while the azimuth values corresponds to a 360 degrees wide field of view, discretized with a step of 1 degree. This configuration allows the transformation of all the point clouds into images with the same resolution of 6×360 (Figure 43).

Figura 43: Un ejemplo de mapa de profundidad capturado en una de las pruebas. La celda es más azul cuanto más cercano está el obstáculo detectado.



One problem that occurs using LiDAR sensors is that the divergence between points grows with the distance to them. Due this divergence in the data, the interval for the vertical field of view has been selected considering the sparse data acquired with high values of the elevation angles. All the values in the deepness map are normalized to $[0, 1]$. For this, a minimum distance of $0m$ and a maximum of $25m$ are specified, if all the events over this distance are not relevant for this problem (lane-change acceptance in an urban environment).

Ejecución del modelo en entornos de simulación

Resultados y conclusiones

Resultados

Mínimo de resultados, los datasets, porque además en materia de cambio de carril hay poco.

Frase para poner en los resultados a raíz de los que estoy obteniendo. Quizá si hablo un poco más de este caso relleno más y parece más consistente: La configuración de la salida en el caso de los cambios de carril es cuanto menos curiosa. Cuando la salida se realiza de forma que el no-cambio de carril se encuentra en un extremo (e.g. 0, 1, 2 siendo 0 el no cambio y 2 el cambio), el entrenamiento devuelve modelos peores que en el caso en el que el no-cambio de carril se encuentra en el medio (e.g. -1, 0, 1). Me aventuro a predecir que en los casos en los que la salida al problema a modelar son los estados con transiciones entre ellos dentro del problema, mantener la relación de cercanía entre transiciones de estado mejora los resultados del entrenamiento y del modelo aprendido (en nuestro caso, -1, 0, 1 mantiene una distancia de 1 transición, mientras que en el caso 0, 1, 2 hay una distancia que no se corresponde con la distancia en transiciones). Esta predicción es una generalización de la experiencia obtenida en estos experimentos y requiere de más estudio.

Conclusiones

¿Pierde rendimiento el sistema cuando se aplican los modelos a escenarios significativamente diferentes de los escenarios de test? Si sí, un trabajo futuro y algo para escribir en conclusiones sería hablar de este defecto y de cómo subsanarlo.

Aportaciones

Futuras líneas de investigación

¿A lo mejor se podría tirar por el campo de las V2X desde esta tesis?

¿Entrar en el tema de la mesosimulación?

Intersection model (lo he visto nombrar por primera vez en http://elib.dlr.de/89233/1/SUMO_Lane_change_model_T.pdf)
Habrán creado también un concepto así rotundas?

No sé, pero me parece lógico tratar de realizar una disociación entre vehículo y conductor en lugar de contemplar el binomio vehículo/conductor como uno sólo. Es más, creo que resultaría interesante evaluar comportamientos de conductores sobre diferentes tipologías de vehículos. La librería de todas formas debería soportar esta disociación (y se debería indicar).

Hemos dejado fuera comportamientos interesantes de estudiar: cruces, rotundas (Driving behavior at a roundabout: An hierarchical Bayesian regression analysis), ...

Parece que se presta poca atención sobre el tema de vehículos pesados (creo que he encontrado en total un par de referencias), y su forma de funcionar es diferente. Puede ser interesante de cara a perfeccionar los simuladores con esta tipología de vehículos y de cara a ser de utilidad a empresas de transporte.

Los cambios de carril no son inmediatos, toman en torno a los 3 segundos, y no he visto que se tenga en cuenta. Todo se centra en la decisión del cambio de carril, pero a la hora de ejecutar van a saco. Quizá habría que prestarle un poco más de atención a este comportamiento.

Para evaluar la efectividad de determinadas técnicas se mira el comportamiento en nivel macro tanto del modelo real como del modelo simulado. De hecho es lo que haré en esta tesis. Sin embargo, no parece que sea el modo más correcto de evaluar la precisión de los modelos. Quizá habría que rebuscar más por este lado para ver cómo se comportan en nivel micro modelos reales y modelos simulados.

At first glance, the obtained outputs of the final stage of the experiments verify that: 1. It is possible to adjust the problem of lane-change execution to a specific driving profile on a different circuit after enough training with a high certainty. 5. As could be expected, a model trained for a profile performs worse when validated against data from a different one, even while driving in the same route with the same driving conditions (due to its different driving styles). So, the developed models intrinsically distinguish driving profiles. 6. The comparison realized between CNN and MLP show that the former ones can generalize much better than the latter in this problem, although they require more training. In fact, by the results obtained in Table , it can be observed that, whereas the training accuracy has slightly decreased in all the architectures, in the case of CNN2– the network has experienced a remarkable increase in the validation accuracy. This fact reinforces the belief that the problem required a network of this size with a moderately-to-high regularization factor to ensure the reduction of overspecialization. Beyond these results, another result has been obtained that is worth highlighting: the difference on how CNNs and MLPs differentiate between subjects. A new training process was run after the one in the paper to see if the resulting values coincide, and the results were similar. It seems that our models based on CNNs differentiate better between profiles than the ones based on MLPs. The reasons for this behaviour may have to do with the way in which CNNs recognize patterns as opposed with MLPs, but further research is required. Wrapping up, the use of CNNs for tasks where the input topology is space-dependent (meaning that the position of the required patterns in the space are relevant) is extremely effective to model temporal events when their windows are narrow and surpasses the precision of the MLPs without loss of time. The last result obtained is the adequacy of the proposed shaking technique to the problem. It has proven to be extremely useful in artificially augmenting the size of the dataset and therefore helps lowering the overfitting problem. We think that, for situations like this one, where the point clouds have errors or where a degree of error or inaccuracy is allowed, this technique can help improving dramatically the size of the datasets to increase the quality of the results.

Sistemas desarrollados

Este capítulo describe todos los sistemas y el software desarrollados e implementados para realizar la tesis. Éstos son tanto los encargados de la captura de datos de los conductores, los que trabajan directamente con el simulador para integrar los controladores generados y los desarrollos para la generación de Software.

Outrun

Biblioteca para la incorporación de modelos de conductor personalizados en SUMO. Hace uso de [TraCI](#).

Modelos de comportamiento

Entrenamiento de controladores difusos mediante Computación Evolutiva (EC, Evolutionary Computation)

Otro software desarrollado

ScanBUS

ScanBUS es un software para la identificación de paquetes enviados por dispositivos a través del Bus CAN del vehículo.

Sistema para la captura de datos multidispositivo

Para la obtención de los datos de conducción se ha desarrollado un sistema que permite la conexión a múltiples dispositivos desde diferentes interfaces. Las razones para su desarrollo son las siguientes

- Sincronización automática de datos de dispositivo en intervalos configurables de tiempo. El sistema permite la configuración de la frecuencia de captura sincronizando los datos recibidos a esa frecuencia.

- Diseño extensible a otros dispositivos. Es software está diseñado para facilitar en la medida de los posible la introducción de nuevos dispositivos usando, para ello, las interfaces apropiadas.
- Hardware compacto. El sistema está integrado en un ordenador de tipo Raspberry PI, aunque es factible su integración en otros sistemas siempre y cuando funcionen con un sistema GNU/Linux e incluyan el hardware necesario para las capturas.

Nagel-Schreckenberg

```
$ python3 main.py \
    --highway_len 250 \
    --iterations 100 \
    --num_cars 50 \
    --max_speed 5 \
    --p 250 \
    --output output.pdf
```


Bibliografía

[Ful,]

[par, 2010] (2010). Directive 2010/40/eu of the european parliament and of the council of 7 july 2010 on the framework for the deployment of intelligent transport systems in the field of road transport and for interfaces with other modes of transport text with eea relevance. *Official Journal of the European Union*, 50:207.

[Aghabayk et al., 2013] Aghabayk, K., Forouzideh, N., and Young, W. (2013). Exploring a local linear model tree approach to car-following. *Computer-Aided Civil and Infrastructure Engineering*, 28(8):581–593.

[Ahmed, 1999] Ahmed, K. I. (1999). Modeling Drivers ' Acceleration and Lane Changing Behavior. *Transportation*, Ph.D:189.

[Al-Shihabi and Mourant, 2001] Al-Shihabi, T. and Mourant, R. R. (2001). A framework for modeling human-like driving behaviors for autonomous vehicles in driving simulators. *The 5th International Conference on Autonomous Agents.*, (June):286–291.

[Alexiadis et al., 2004] Alexiadis, V., Colyar, J., Halkias, J., Hranac, R., and McHale, G. (2004). The next generation simulation program. *ITE Journal (Institute of Transportation Engineers)*, 74(8):22–26.

[Balmer et al., 2004] Balmer, M., Cetin, N., Nagel, K., and Raney, B. (2004). Towards truly agent-based traffic and mobility simulations. *Proceedings of the Third International Joint Conference on Autonomous Agents and Multiagent Systems*, 1:60–67.

[Bando et al., 2013] Bando, T., Takenaka, K., Nagasaka, S., and Taniuchi, T. (2013). Unsupervised drive topic finding from driving behavioral data. In *Intelligent Vehicles Symposium (IV), 2013 IEEE*, pages 177–182. IEEE.

[Barlovic et al., 1998] Barlovic, R., Santen, L., Schadschneider, A., and Schreckenberg, M. (1998). Metastable states in cellular automata for traffic flow. *Eur. Phys. J. B*, 5:793–800.

[Behrisch et al., 2011] Behrisch, M., Bieker, L., Erdmann, J., and Krajzewicz, D. (2011). Sumo—simulation of urban mobility: an overview. In *Proceedings of SIMUL 2011, The Third International Conference on Advances in System Simulation*. ThinkMind.

- [Bender et al., 2015] Bender, A., Agamennoni, G., Ward, J. R., Worrall, S., and Nebot, E. M. (2015). An unsupervised approach for inferring driver behavior from naturalistic driving data. *IEEE Transactions on Intelligent Transportation Systems*, 16(6):3325–3336.
- [Bexelius, 1968] Bexelius, S. (1968). An extended model for car-following.
- [Bingham, 2001] Bingham, E. (2001). Reinforcement learning in neurofuzzy traffic signal control. *European Journal of Operational Research*, 131(2):232–241.
- [Brilon and Wu, 1999] Brilon, W. and Wu, N. (1999). Evaluation of cellular automata for traffic flow simulation on freeway and urban streets. *Traffic and Mobility*.
- [Casas et al., 2011] Casas, J., Perarnau, J., and Torday, A. (2011). The need to combine different traffic modelling levels for effectively tackling large-scale projects adding a hybrid meso/micro approach. *Procedia-Social and Behavioral Sciences*, 20:251–262.
- [Chaib-draa and Levesque, 1994] Chaib-draa, B. and Levesque, P. (1994). Hierarchical model and communication by signs, signals, and symbols in multi-agent environments. *on Modelling Autonomous Agents in a Multi-*
- [Chakroborty and Kikuchi, 2003] Chakroborty, P. and Kikuchi, S. (2003). Calibrating the membership functions of the fuzzy inference system: Instantiated by car-following data. *Transportation Research Part C: Emerging Technologies*, 11(2):91–119.
- [Chan et al., 2012] Chan, K. Y., Dillon, T. S., Singh, J., and Chang, E. (2012). Neural-Network-Based Models for Short-Term Traffic Flow Forecasting Using a Hybrid Exponential Smoothing and Levenberg-Marquardt Algorithm. *IEEE Transactions on Intelligent Transportation Systems*, 13(2):644–654.
- [Chandler et al., 1958] Chandler, R. E., Herman, R., and Montroll, E. W. (1958). Traffic Dynamics: Studies in Car Following. *Operations Research*, 6(2):165–184.
- [Clymer, 2002] Clymer, J. (2002). Simulation of a vehicle traffic control network using a fuzzy classifier system. In *Proceedings 35th Annual Simulation Symposium. SS 2002*, pages 285–291. IEEE Comput. Soc.
- [Cybenko, 1989] Cybenko, G. (1989). Approximation by superpositions of a sigmoidal function. *Mathematics of control, signals and systems*, 2(4):303–314.
- [Das and Bowles, 1999] Das, S. and Bowles, B. (1999). Simulations of highway chaos using fuzzy logic. *18th International Conference of the North American Fuzzy Information Processing Society - NAFIPS (Cat. No.99TH8397)*, pages 130–133.

- [Das et al., 1999] Das, S., Bowles, B. A., Houghland, C. R., Hunn, S. J., and Zhang, Y. (1999). Microscopic simulations of freeway traffic flow. In *Proceedings of the Thirty-Second Annual Simulation Symposium IEEE Computer Society*, pages 79–84. IEEE Comput. Soc.
- [Dia, 2002] Dia, H. (2002). An agent-based approach to modelling driver route choice behaviour under the influence of real-time information. *Transportation Research Part C: Emerging Technologies*, 10(5):331–349.
- [Díaz Álvarez et al., 2014] Díaz Álvarez, A., Serradilla García, F., Naranjo, J. E., Anaya, J. J., and Jiménez, F. (2014). Modeling the driving behavior of electric vehicles using smartphones and neural networks. *IEEE Intelligent Transportation Systems Magazine*, 6(3):44–53.
- [Dijkstra, 1972] Dijkstra, E. W. (1972). The humble programmer. *Communications of the ACM*, 15(10):859–866.
- [Dingus et al., 2006] Dingus, T. A., Klauer, S. G., Neale, V. L., Petersen, A., Lee, S., Sudweeks, J., Perez, M., Hankey, J., Ramsey, D., Gupta, S., et al. (2006). The 100-car naturalistic driving study, phase ii-results of the 100-car field experiment. Technical report.
- [Dougherty et al., 1993] Dougherty, M. S., Kirby, H. R., and Boyle, R. D. (1993). The use of neural networks to recognise and predict traffic congestion. *Traffic engineering & control*, 34(6):311–4.
- [Dresner and Stone, 2004] Dresner, K. and Stone, P. (2004). *Proceedings of the Third International Joint Conference on Autonomous Agents and Multiagent Systems : AAMAS 2004 : New York City, New York, USA : July 19-23, 2004*. IEEE Computer Society.
- [Du and Swamy, 2006] Du, K. and Swamy, M. (2006). Neural networks in a softcomputing framework.
- [Ehlert and Rothkrantz, 2001] Ehlert, P. a. M. and Rothkrantz, L. J. M. (2001). A reactive driving agent for microscopic traffic simulation. *Modelling and Simulation 2001*, pages 943–949.
- [Finin et al., 1994] Finin, T., Fritzson, R., McKay, D., and McEntire, R. (1994). KQML as an agent communication language. *Proceedings of the third international conference on Information and knowledge management - CIKM '94*, pages 456–463.
- [Fix and Armstrong, 1990] Fix, E. and Armstrong, H. (1990). Modeling human performance with neural networks. *1990 IJCNN International Joint Conference on Neural Networks*, pages 247–252 vol.1.
- [Fritzsche and Ag, 1994] Fritzsche, H.-t. and Ag, D.-b. (1994). A model for traffic simulation. *Traffic Engineering & Control*, 35(5):317–321.
- [Galil and Rao, 2000] Galil, a. and Rao, S. (2000). -"Application of Agent Technology to Telecommunication Management Services".

- [Gazis et al., 1959] Gazis, D. C., Herman, R., and Potts, R. B. (1959). Car-Following Theory of Steady-State Traffic Flow. *Operations Research*, 7(4):499–505.
- [Gipps, 1981] Gipps, P. G. (1981). A behavioural car-following model for computer simulation.
- [Gipps, 1986] Gipps, P. G. (1986). A model for the structure of lane-changing decisions. *Transportation Research Part B*, 20(5):403–414.
- [Gu et al., 2015] Gu, M., Billot, R., Faouzi, N.-e. E., Lyon, D., and Hassas, S. (2015). Multi-Agent Dynamic Coupling for Cooperative Vehicles Modeling. pages 4276–4277.
- [Halati et al., 1997] Halati, A., Lieu, H., and Walker, S. (1997). CORSIM-corridor traffic simulation model. *Traffic Congestion and Traffic Safety in the*.
- [Hatipkarasulu, 2003] Hatipkarasulu, Y. (2003). A Variable Response Time Lag Module for Car Following Models Using Fuzzy Set Theory. (225).
- [Hebb, 1968] Hebb, D. (1968). The organization of behavior.
- [Hidas, 2002] Hidas, P. (2002). Modelling lane changing and merging in microscopic traffic simulation. *Transportation Research Part C: Emerging Technologies*, 10(5-6):351–371.
- [Hinton, 2006] Hinton, G. E. (2006). Reducing the Dimensionality of Data with Neural Networks. *Science*, 313(5786):504–507.
- [Hou et al., 2011] Hou, H., Jin, L., Niu, Q., Sun, Y., and Lu, M. (2011). Driver Intention Recognition Method Using Continuous Hidden Markov Model. *International Journal of Computational Intelligence Systems*, 4(3):386–393.
- [Hunt and Lyons, 1994] Hunt, J. G. and Lyons, G. D. (1994). Modelling dual carriageway lane changing using neural networks. *Transportation Research Part C*, 2(4):231–245.
- [Huval et al., 2015] Huval, B., Wang, T., Tandon, S., Kiske, J., Song, W., Pazhayampallil, J., Andriluka, M., Rajpurkar, P., Migimatsu, T., Cheng-Yue, R., Mujica, F., Coates, A., and Ng, A. Y. (2015). An Empirical Evaluation of Deep Learning on Highway Driving. *arXiv*, pages 1–7.
- [Imprailou et al., 2016] Imprailou, M.-I. M., Quddus, M., Pitfield, D. E., and Lord, D. (2016). Re-visiting crash-speed relationships: A new perspective in crash modelling. *Accident Analysis & Prevention*, 86:173–185.
- [Jia et al., 2003] Jia, H., Juan, Z., and Ni, A. (2003). Develop a car-following model using data collected by 'five-wheel system'. *IEEE Conference on Intelligent Transportation Systems, Proceedings, ITSC*, 1:346–351.

- [Jin et al., 2016] Jin, J., Ma, X., Koskinen, K., Rychlik, M., and Kosonen, I. (2016). Evaluation of fuzzy intelligent traffic signal control (fits) system using traffic simulation. In *Transportation Research Board 95th Annual Meeting*, number 16-4359.
- [Jin, 2006] Jin, W.-L. (2006). A kinematic wave theory of lane-changing vehicular traffic.
- [K. et al., 1996] K., A., M., B.-A., H., K., and R., M. (1996). Models of freeway lane changing and gap acceptance behavior. *Transportation and traffic theory*, 13:501–515.
- [Kerner et al., 2008] Kerner, B., Klenov, S., and Brakemeier, A. (2008). Testbed for wireless vehicle communication: A simulation approach based on three-phase traffic theory. In *Intelligent Vehicles Symposium, 2008 IEEE*, pages 180–185. IEEE.
- [Khodayari et al., 2012] Khodayari, A., Ghaffari, A., Kazemi, R., and Braunstingl, R. (2012). A Modified Car-Following Model Based on a Neural Network Model of the Human Driver Effects. In *IEEE on Systems, Man, and Cybernetics*, volume 42, pages 1440–1449.
- [Kikuchi and Chakroborty, 1992] Kikuchi, S. and Chakroborty, P. (1992). Car-following model based on fuzzy inference system. 1(1365).
- [Kiszka et al., 1985] Kiszka, J. B., Kochański, M. E., and Sliwiński, D. S. (1985). The Influence of Some Parameters on the Accuracy of a Fuzzy Model. *Industrial Applications of Fuzzy Control*, pages 187–230.
- [Krajzewicz et al., 2012] Krajzewicz, D., Erdmann, J., Behrisch, M., and Bieker, L. (2012). Recent development and applications of SUMO - Simulation of Urban MObility. *International Journal On Advances in Systems and Measurements*, 5(3&4):128–138.
- [Krajzewicz et al., 2002] Krajzewicz, D., Hertkorn, G., Rössel, C., and Wagner, P. (2002). Sumo (simulation of urban mobility) – an open-source traffic simulation. In *Proceedings of the 4th Middle East Symposium on Simulation and Modelling (MESM20002)*, pages 183–187.
- [Krauss et al., 1997] Krauss, S., Wagner, P., and Gawron, C. (1997). Metastable states in a microscopic model of traffic flow. *Physical Review E*, 55(5):5597–5602.
- [Kuge et al., 2000] Kuge, N., Yamamura, T., Shimoyama, O., and Liu, A. (2000). A Driver Behavior Recognition Method Based on a Driver Model Framework. *Structure, 109(1dm)*:469–476.
- [Laval and Daganzo, 2006] Laval, J. A. and Daganzo, C. F. (2006). Lane-changing in traffic streams. *Transportation Research Part B: Methodological*, 40(3):251–264.

- [Lerner et al., 2010] Lerner, N., Jenness, J., Singer, J., Klauer, S., Lee, S., Donath, M., Manser, M., and Ward, N. (2010). An exploration of vehicle-based monitoring of novice teen drivers. *Final Report. NHTSA, Report No. DOT HS*, 811:333.
- [Lipton et al., 2015] Lipton, Z. C., Berkowitz, J., and Elkan, C. (2015). A critical review of recurrent neural networks for sequence learning. *arXiv preprint arXiv:1506.00019*.
- [Liu and Li, 2013] Liu, R. and Li, X. (2013). Stability analysis of a multi-phase car-following model. *Physica A: Statistical Mechanics and its Applications*, 392(11):2660–2671.
- [Ma, 2004] Ma, X. (2004). Toward An Integrated Car-following and Lane- changing Model by A Neural Fuzzy Approach. pages 1–15.
- [Margolus, 1993] Margolus, N. (1993). CAM-8: a computer architecture based on cellular automata *.
- [Maye et al., 2011] Maye, J., Triebel, R., Spinello, L., and Siegwart, R. (2011). Bayesian on-line learning of driving behaviors. In *Robotics and Automation (ICRA), 2011 IEEE International Conference on*, pages 4341–4346. IEEE.
- [McCarthy et al., 1956] McCarthy, J., Minsky, M., Rochester, N., and Shannon, C. (1956). Dartmouth conference. In *Dartmouth Summer Research Conference on Artificial Intelligence*.
- [McCulloch and Pitts, 1943] McCulloch, W. S. and Pitts, W. (1943). A logical calculus of the ideas immanent in nervous activity. *The Bulletin of Mathematical Biophysics*, 5(4):115–133.
- [McDonald et al., 1997] McDonald, M., Wu, J., and Brackstone, M. (1997). Development of a fuzzy logic based microscopic motorway simulation model. *IEEE Conference on*.
- [Michon, 1985] Michon, J. A. (1985). A critical view of driver behavior models: what do we know, what should we do? In *Human behavior and traffic safety*, pages 485–524. Springer.
- [Minderhoud, 1999] Minderhoud, M. (1999). *Supported Driving:Impacts on Motorway Traffic Flow*. PhD thesis.
- [Minsky and Papert, 1969] Minsky, M. and Papert, S. (1969). Perceptrons.
- [Muñoz et al., 2010] Muñoz, J., Gutierrez, G., and Sanchis, A. (2010). A human-like torcs controller for the simulated car racing championship. In *Proceedings of the 2010 IEEE Conference on Computational Intelligence and Games*, pages 473–480. IEEE.
- [Munoz et al., 2001] Munoz, L., Gomes, G., Yi, J., Toy, C., Horowitz, R., and Alvarez, L. (2001). Integrated meso-microscale traffic simulation of hierarchical ahs control architectures. In *Intelligent Transportation Systems, 2001. Proceedings. 2001 IEEE*, pages 82–87. IEEE.

- [Nagel and Schreckenberg, 1992] Nagel, K. and Schreckenberg, M. (1992). A cellular automaton model for freeway traffic.
- [Nagel et al., 1998] Nagel, K., Wolf, D. E., Wagner, P., and Simon, P. (1998). Two-lane traffic rules for cellular automata: A systematic approach. *Physical Review E*, 58(2):1425–1437.
- [OICA, 2015] OICA (2015). Motorization rate 2014 – worldwide.
- [Osogami et al., 2012] Osogami, T., Imamichi, T., Mizuta, H., Morimura, T., Raymond, R., Suzumura, T., Takahashi, R., and Idé, T. (2012). IBM Mega Traffic Simulator.
- [Pakkenberg and Gundersen, 1997] Pakkenberg, B. and Gundersen, H. J. (1997). Neocortical neuron number in humans: effect of sex and age. *The Journal of comparative neurology*, 384(2):312–20.
- [Panwai and Dia, 2007] Panwai, S. and Dia, H. (2007). Neural agent car-following models. *IEEE Transactions on Intelligent Transportation Systems*, 8(1):60–70.
- [Pipes, 1953] Pipes, L. (1953). An operational analysis of traffic dynamics. *Journal of applied physics*.
- [Poslad, 2007] Poslad, S. (2007). Specifying Protocols for Multi-Agent Systems Interaction. *ACM Transactions on Autonomous and Adaptive Systems*, 2(4, Article 15):25.
- [Quaassdorff et al., 2016] Quaassdorff, C., Borge, R., Pérez, J., Lumbreras, J., de la Paz, D., and de Andrés, J. M. (2016). Microscale traffic simulation and emission estimation in a heavily trafficked roundabout in madrid (spain). *Science of The Total Environment*, 566:416–427.
- [Ramón and Cajal, 1904] Ramón, S. and Cajal, S. (1904). *Textura del Sistema Nervioso del Hombre y de los Vertebrados*, volume 2. Madrid Nicolas Moya.
- [Rasmussen, 1986] Rasmussen, J. (1986). *Information Processing and Human-Machine Interaction: An Approach to Cognitive Engineering*. North-Holland.
- [Reuschel, 1950] Reuschel, A. (1950). Fahrzeuggbewegungen in der Kolonne. *Osterr. Ing. Archiv.*, 4(1):193–215.
- [Rumelhart et al., 1985] Rumelhart, D. E., Hinton, G. E., and Williams, R. J. (1985). Learning internal representations by error propagation. Technical report, DTIC Document.
- [Russell et al., 2003] Russell, S. J., Norvig, P., Canny, J. F., and Malik, Jitendra M, E. D. D. (2003). *Artificial intelligence: a modern approach*, volume 2. Prentice hall Upper Saddle River.
- [Sasoh and Ohara, 2002] Sasoh, A. and Ohara, T. (2002). Shock Wave Relation Containing Lane Change Source Term for Two-Lane Traffic Flow. *Journal of the Physical Society of Japan*, 71(9):2339–2347.

- [Sekizawa et al., 2007] Sekizawa, S., Inagaki, S., Suzuki, T., Hayakawa, S., Tsuchida, N., Tsuda, T., and Fujinami, H. (2007). Modeling and recognition of driving behavior based on stochastic switched ARX model. *IEEE Transactions on Intelligent Transportation Systems*, 8(4):593–606.
- [Shiose et al., 2001] Shiose, T., Onitsuka, T., and Taura, T. (2001). Effective information provision for relieving traffic congestion. In *Proceedings Fourth International Conference on Computational Intelligence and Multimedia Applications. ICCIMA 2001*, pages 138–142. IEEE.
- [Simonelli et al., 2009] Simonelli, F., Bifulco, G. N., and Martinis, V. D. (2009). Human-Like Adaptive Cruise Control Systems through a Learning Machine Approach. *Applications of Soft Computing*, pages 240–249.
- [Sparmann, 1978] Sparmann, U. (1978). *Spurwechselvorgänge auf zweispurigen BAB-Richtungsfahrbahnen*.
- [Suzumura and Kanezashi, 2012] Suzumura, T. and Kanezashi, H. (2012). Highly scalable x10-based agent simulation platform and its application to large-scale traffic simulation. *Proceedings of the 2012 IEEE/ACM 16th*.
- [Toledo et al., 2003] Toledo, T., Koutsopoulos, H. N., and Ben-Akiva, M. (2003). Modeling Integrated Lane-Changing Behavior. *Transportation Research Record*, 1857(1):30–38.
- [Toledo et al., 2007] Toledo, T., Koutsopoulos, H. N., and Ben-Akiva, M. (2007). Integrated driving behavior modeling. *Transportation Research Part C: Emerging Technologies*, 15(2):96–112.
- [Tordeux et al., 2011] Tordeux, A., Lassarre, S., and Roussignol, M. (2011). A study of the emergence of kinematic waves in targeted state car-following models of traffic. <http://cybergeo.revues.org>.
- [Trask ANDREWTRASK et al.,] Trask ANDREWTRASK, A., Gilmore DAVIDGILMORE, D., and Russell MATTHEWRUSSELL, M. Modeling Order in Neural Word Embeddings at Scale.
- [Turing, 1950] Turing, A. M. (1950). Computing machinery and intelligence. *Mind*, 59(236):433–460.
- [Van Hoorn et al., 2009] Van Hoorn, N., Togelius, J., Wierstra, D., and Schmidhuber, J. (2009). Robust player imitation using multiobjective evolution. In *2009 IEEE Congress on Evolutionary Computation*, pages 652–659. IEEE.
- [Van Ly et al., 2013] Van Ly, M., Martin, S., and Trivedi, M. M. (2013). Driver classification and driving style recognition using inertial sensors. In *Intelligent Vehicles Symposium (IV), 2013 IEEE*, pages 1040–1045. IEEE.

- [Wagner et al., 1997] Wagner, P., Nagel, K., and Wolf, D. E. (1997). Realistic multi-lane traffic rules for cellular automata. *Physica A: Statistical Mechanics and its Applications*, 234(3–4):687–698.
- [Wegener et al., 2008] Wegener, A., Piórkowski, M., Raya, M., Hellbrück, H., Fischer, S., and Hubaux, J.-P. (2008). TraCI. *Proceedings of the 11th communications and networking simulation symposium on - CNS '08*, (August 2016):155.
- [Wei et al., 2000] Wei, H., Lee, J., Li, Q., and Li, C. (2000). Observation-Based Lane-Vehicle Assignment Hierarchy: Microscopic Simulation on Urban Street Network. *Transportation Research Record*, 1710(1):96–103.
- [Weidemann and Reiter, 1992] Weidemann, R. and Reiter, U. (1992). Microscopic Traffic Simulation, The Simulation System-Mission. *University Karlsruhe, Germany*, 2:54.
- [Wiedemann, 1974] Wiedemann, R. (1974). Simulation des Straßenverkehrsflusses. Technical report.
- [Wooldridge et al., 1995] Wooldridge, M., Jennings, N. R., Adorni, G., Poggi, A., Allen, J. F., Bates, J., Bell, J., BELNAP, N., PERLOFF, M., Bratman, M. E., Israel, D. J., Pollack, M. E., Brooks, R., Brooks, R. A., Bussmann, S., Demazeau, Y., Castelfranchi, C., Chaib-Draa, B., Moulin, B., Mandiau, R., Millot, P., Chang, E., Chapman, D., Chellas, B. F., Cohen, P. R., Levesque, H. J., Cohen, P. R., Perrault, C. R., Cutkosky, M., Engelmore, R., Fikes, R., Genesereth, M., Gruber, T., Mark, W., Tenenbaum, J., Weber, J., Downs, J., Reichgelt, H., Emerson, E. A., Halpern, J. Y., Fagin, R., Halpern, J. Y., Vardi, M. Y., Fisher, M., Gasser, L., Gasser, L., Braganza, C., Herman, N., Genesereth, M. R., Ketchpel, S. P., Georgeff, M. P., Greif, I., Guha, R. V., Lenat, D. B., Haas, A. R., Halpern, J. Y., Halpern, J. Y., Moses, Y., Halpern, J. Y., Vardi, M. Y., Hayes-Roth, B., Hewitt, C., Huang, J., Jennings, N. R., Fox, J., Jennings, N. R., JENNINGS, N. R., Jennings, N., Varga, L., Aarnts, R., Fuchs, J., Skarek, P., Kaelbling, L. P., Kraus, S., Lehmann, D., Kripke, S. A., Maes, P., Maes, P., McCabe, F. G., Clark, K. L., Mukhopadhyay, U., Stephens, L. M., Huhns, M. N., Bonnell, R. D., Müller, J. P., Pischel, M., Thiel, M., Newell, A., Simon, H. A., Norman, T. J., Long, D., PAPAZOGLOU, M. P., LAUFMANN, S. C., SELLIS, T. K., Perlis, D., Perlis, D., Perloff, M., Poggi, A., Reichgelt, H., Sacerdoti, E. D., Searle, J. R., Shoham, Y., Thomas, B., Shoham, Y., Schwartz, A., Kraus, S., Thomason, R. H., Varga, L., Jennings, N. R., Cockburn, D., Vere, S., Bickmore, T., Wavish, P., Graham, M., Weerasooriya, D., Rao, A., Ramamohanarao, K., and Wooldridge, M. (1995). Intelligent agents: theory and practice. *The Knowledge Engineering Review*, 10(02):115.
- [Wu et al., 2003] Wu, J., Brackstone, M., and McDonald, M. (2003). The validation of a microscopic simulation model: A methodological case study. *Transportation Research Part C: Emerging Technologies*, 11(6):463–479.

- [Wymann et al., 2013] Wymann, B., Espi, E., Guionneau, C., Dimitrakakis, C., and Sumner, A. (2013). TORCS : The open racing car simulator e. at <http://torcs.> . . . , pages 1–4.
- [y Cajal, 1888] y Cajal, S. R. (1888). *Estructura de los centros nerviosos de las aves*.
- [Yang and Koutsopoulos, 1996] Yang, Q. and Koutsopoulos, H. N. (1996). A microscopic traffic simulator for evaluation of dynamic traffic management systems. *Transportation Research Part C: Emerging Technologies*, 4(3 PART C):113–129.
- [Zhang et al., 2011] Zhang, J., Wang, F.-Y., Wang, K., Lin, W.-H., Xu, X., and Chen, C. (2011). Data-Driven Intelligent Transportation Systems: A Survey. *IEEE Transactions on Intelligent Transportation Systems*, 12(4):1624–1639.
- [Zheng et al., 2013] Zheng, J., Suzuki, K., and Fujita, M. (2013). Car-following behavior with instantaneous driver-vehicle reaction delay: A neural-network-based methodology. *Transportation Research Part C: Emerging Technologies*, 36:339–351.
- [Zheng and McDonald, 2005] Zheng, P. and McDonald, M. (2005). Application of Fuzzy Systems in the Car-Following Behaviour Analysis. pages 782–791. Springer Berlin Heidelberg.

Índice alfabético

- approaching, 63
- back propagation, 47
- back-propagation, 71
- car-following, 51, 58, 63–67, 70–72
- courtesy yielding function, 67
- critical gap, 65
- critical-gap, 67
- emergency, 63
- fast-forward, 70
- free flow, 63
- free-flow, 66, 69
- gap acceptance, 65
- gap acceptance, 63, 65
- gap-acceptance, 67
- lane selection, 63
- lane-changing, 66
- lane-selection, 65
- merging, 63, 65
- neuro-fuzzy, 71, 72
- recurrente, 70
- umbral perceptual, 65

Cómo citar esta tesis

Si deseas citar esta tesis, lo primero gracias. Me alegro de que te sirva para tu investigación. Si lo deseas, incluye el siguiente código en bibtex:

```
@phdthesis{blazaaid20167,  
  author = {Alberto Díaz Álvarez},  
  abstract = {XXX},  
  pages = {XXX},  
  title = {Modelado de comportamiento de conductores con técnicas de Inteligencia Computacional},  
  url = {XXX},  
  year = {9 de febrero de 2018}  
}
```

Acerca del código fuente

La presente tesis lleva consigo numerosas horas de programación y, por tanto, muchísimas líneas de código. Éste se encuentra en formato electrónico como datos adjuntos a la memoria y no como capítulo o anexo a ésta, una forma más manejable para su consulta y a la vez respetuosa con el medio ambiente. No obstante sí es posible que existan pequeños fragmentos de código para apoyar explicaciones. En caso de necesitar los fuentes y no estar disponibles los datos anexos a la memoria, puedes contactar directamente conmigo en alberto.diaz@upm.es.