

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, 22 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	23
Motivación	24
Objetivos	26
Supuestos	27
Restricciones	27
Estructura de la tesis	28
I Estado de la cuestión	29
Inteligencia Computacional	31
De Inteligencia Artificial a Inteligencia Computacional . . .	31
El rol del aprendizaje en la Inteligencia Computacional . . .	35
Epochs, conjuntos de entrenamiento, test y validación	36
Problemas del aprendizaje en la Inteligencia Computacional	37
Deep learning	37
Artificial Neural Networks	39
Funciones de activación	41
Estructura y clasificación de redes neuronales	43
Perceptrones multicapa	44
Redes de convolución	46
Solucionando los problemas de entrenamiento	50
Grafos computacionales	51
Lógica Difusa	52
Ampliando la teoría de conjuntos tradicional	53
Razonamiento	56

Sistemas de control difuso	57
Tipos de Sistemas de Control Difuso	59
El paradigma de los Agentes Inteligentes	60
Tipos de entorno	61
Arquitecturas	62
Sistema Multiagente	64
Simulación de tráfico	67
Clasificación de simuladores de tráfico	68
Tipos de simulador en función de la complejidad	68
Tipos de simulador en función del espacio y el tiempo	70
Modelos de microsimulación	71
Microsimulación basada en Autómatas Celulares	71
Microsimulación basada en sistemas multiagentes	72
Software de simulación	75
Entornos de simulación a estudiar	76
Entorno seleccionado: SUMO	77
La interfaz TraCI	78
Modelos de comportamiento	81
Comportamientos modelados	83
Nomenclatura	84
Modelado de conductores clásico	84
Modelado basado en	88
Papers a añadir o al menos relevantes	92
II Modelado de conductores: definición y diseño	93
Definición del problema	95
Modelo de comportamiento de conductor	97
Introducción	97
Captura de datos	97
Instrumentación del vehículo	98
Descripción de datos a extraer	98

Selección de rutas	98
Selección de sujetos	98
Preparación de los datos	98
Entrenamiento de modelos	98
Perspectiva general	98
Comportamiento longitudinal	98
Comportamiento en cambio de carril	98
Validación del modelo	98
Instrumentación del vehículo	99
Perfiles de conducción	100
Routes	102
Proceso de los datos	103
Descripción de los conjuntos de datos	103
Determinando la intencionalidad en el cambio de carril	105
Ejecucutando el cambio de carril	105
Ejecución del modelo en entornos de simulación	109
III Resultados y conclusiones	111
Resultados	113
Conclusiones	115
Aportaciones	115
Fururas líneas de investigación	115
Sistemas desarrollados	117
Outrun	117
Modelos de comportamiento	117
Entrenamiento de controladores difusos mediante Compu- tación Evolutiva (EC, Evolutionary Computation)	117
Otro software desarrollado	117
ScanBUS	117
Sistema para la captura de datos multidispositivo	117
Módulos de ROS	118

Índice de figuras

1.	Censo de conductores según género y edad	27
2.	Experimento mental de la <i>Habitación China</i> , por John Searle.	32
3.	Diferentes objetivos perseguidos por la Inteligencia Artificial.	34
4.	Separación clásica de conjuntos de datos en tareas de Machine Learning.	36
5.	Ciclo de aplicación de soluciones basadas en Inteligencia Computacional	39
6.	Capacidad de los modelos en función de la cantidad de datos	39
7.	Ilustración de una sección del neocórtez humano . . .	40
8.	Modelo de neurona artificial de McCulloch y Pitts . .	40
9.	Funciones de activación: sigmoidal y tangente hiperbólica.	41
10.	Funciones de activación: ReLU y Leaky-ReLU.	42
11.	Diferencias entre topologías de redes <i>feed-forward</i> y <i>recurrentes</i>	44
12.	Estructura general de una Convolutional Neural Network	47
13.	Descripción de la capa de convolución	48
14.	Ejemplo de las operaciones de <i>pooling</i>	49
15.	Ejemplo de grafo computacional.	51
16.	Derivadas parciales sobre un grafo computacional. .	52
17.	Gráfica de funciones de pertenencia triangular y trapezoidal.	54
18.	<i>t</i> -normas del mínimo y del producto algebraico. . . .	55
19.	<i>t</i> -normas del máximo y de la suma algebraica. . . .	55

20.	Diferencias entre <i>modus ponens</i> clásico y <i>modus ponens generalizado</i>	56
21.	Esquema de un sistema de control difuso	57
22.	Representación de conjuntos de reglas como matriz . .	58
23.	Representación de conjuntos de reglas como matriz . .	59
24.	Esquema de agente y sus propiedades.	61
25.	Arquitectura básica de un agente.	62
26.	Diferencias entre un agente sin y con modelo de entorno. .	63
27.	Arquitecturas de agente según su comportamiento. . .	64
28.	Diferencias entre colaboración y competitividad de agentes.	66
29.	Aspectos medibles del tráfico	68
30.	Clasificación de simuladores según granularidad . . .	69
31.	Alternativas a la clasificación por granularidad	70
32.	Ejemplo de simulador basado en Autómata Celular . .	70
33.	Ejemplo de modelo lineal en un espacio continuo . . .	71
34.	Ejemplo de efecto de ondas de choque en simulación de tipo Nagel-Schreckenberg	72
35.	Simulación de comportamiento en intersección basada en un	73
36.	Captura de pantalla del simulador MovSim	74
37.	Características obligatorias y deseables del simulador a elegir	75
38.	Captura de pantalla del simulador SUMO	77
39.	Ejemplo de forma de envío de mensajes a través de TraCI	78
40.	Arquitectura de la plataforma TraaS	79
41.	Los tres niveles jerárquicos de conducción según [Michon, 1985]	81
42.	Diferentes modelos de aceleración	83
43.	Operaciones involucradas en el proceso de cambio de carril	83
44.	Nomenclaturas a usar en los modelos de conducción .	84
45.	Evolución de los tres tipos generales de modelo de <i>car-following</i>	85

46.	Ejemplo de cambio de carril obligatorio frente a cambio de carril discrecional	86
47.	Efecto de la distancia en el tipo de cambio de carril según el modelo de [Gipps, 1986]	87
48.	Estructura del modelo de comportamiento propuesto por [Toledo et al., 2007]	88
49.	Principales áreas de aplicación de la en los Sistema Inteligente de Transporte (ITS, Intelligent Transport System)	89
50.	La inexactitud se tiene en cuenta en los modelos de la	90
51.	Ejemplo de aproximación <i>neuro-fuzzy</i> al control de señales de tráfico	91
52.	The instrumented Mitsubishi iMiEV.	99
53.	The instrumented vehicle schema.	101
54.	Comparación de indicadores entre los diferentes perfiles de conducción.	102
55.	Las dos rutas del experimento, (a) Ruta R_1 para entrenamiento y (b) Ruta R_2 para validación.	102
56.	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.	103
57.	Un ejemplo de mapa de profundidad capturado en una de las pruebas. La celda es más azul cuanto más cercano está el obstáculo detectado.	108

Índice de cuadros

1.	Tabla comparativa de los simuladores seleccionados	77
2.	Resumen de información extraída del vehículo instrumentado	100
3.	Indicadores de los datos extraídos de cada perfil	101
4.	Resumen de información extraída del vehículo instrumentado	104

Introducción

La **Inteligencia Artificial (AI)** como área de conocimiento ha experimentado un creciente interés en los últimos años. Esto no siempre ha sido así; desde su nacimiento ha ido alternando épocas de mucha actividad y de muy poca, debido a las altas expectativas que generan los nuevos avances en el área. Sin embargo, en la actualidad es muy difícil encontrar un campo que no se beneficie directamente de sus técnicas.

Una de las razones es su carácter multidisciplinar ya que, aunque pertenece al campo de la computación, es transversal a muchos otros campos de naturaleza muy diferente como, por ejemplo, la biología, la 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 (IC)**, 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 **IC**. Revisaremos las diferencias entre conceptos de diferentes autores en el capítulo **Inteligencia Computacional**.

Uno de los campos de aplicación es el de los **ITS**. 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 la que se estableció el marco de implantación de los **ITS** para toda la Unión Europea, quedando éstos definidos como:

[Los ITS ...] son aplicaciones avanzadas que, sin incluir la inteligencia 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.

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 modelos de conductor que responden a funciones más o menos complejas, alejadas de la realidad, además con pocas o ninguna opciones de personalización. Esto provoca que dichos modelos se adapten poco al comportamiento 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 [IC](#). Estos perfiles extraídos se aplicarán a un entorno de simulación basado en [Sistemas Multiagente](#). Así, una vez configurado el entorno, se podrán 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 2016, el número de vehículos a nivel mundial superó los 1,350 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 humano¹.

¹ 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 aun infraestructura para la recarga de vehículos eléctricos, dicho crecimiento ha superado el 120%.

- 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 lleguen 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. Con el ritmo actual, dependiendo de la fuente estamos o bien llegando obien ya hemos sobrepasado 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, relacionado no sólo con el número sino con la gravedad de los accidentes ([Imprilou 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]).

Éstos 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 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 *Sistemas Multiagente* 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 aproximada de comportamiento en situaciones de tráfico del 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: *La aplicación de técnicas pertenecientes al campo de la IC 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 *IC*. Para ello se consideran los siguientes objetivos específicos:

- Estudiar y aplicar técnicas de la *IC* sobre el área de la conducción.
- Realizar un estudio naturalista de conducción² 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.

² Un **estudio naturalista de conducción** basa 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.

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 1).

Restricciones

- Los modelos generados se corresponderán a entornos urbanos.
- Reduciremos el comportamiento del conductor a los de circulación en línea y en cambio de carril ³.
- La resolución máxima del modelo creado será de 10 Hz.
- En el caso de los modelos que hacen uso de *Artificial Neural Network*, no se pueden explicar las razones del comportamiento inferido.

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](#).

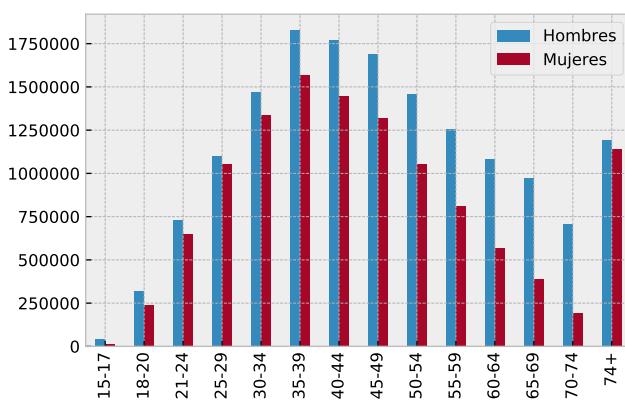


Figura 1: Último censo de conductores según género segmentado por edades. Fuente: Dirección General de Tráfico ([dgt.es](#)).

Estructura de la tesis

La tesis se divide en tres partes. La primera se corresponde con el estado de la cuestión, compuesta por los capítulos [Inteligencia Computacional](#), [Simulación de tráfico](#) y [Modelos de comportamiento](#), explicando en qué punto se encuentra la literatura de los temas en los que se apoya la presente tesis.

En la segunda parte se desarrolla el tema del modelado de conductores, donde en los capítulos [Definición del problema](#), [Modelo de comportamiento de conductor](#) y [Ejecución del modelo en entornos de simulación](#) se introduce el problema a resolver propuesto en la hipótesis, se desarrolla del modelo de conducción personalizado y se muestran la configuración de las simulaciones.

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 multitud de variables que determinan en mayor o menor grado su comportamiento. En la mayoría de los casos es muy complicado determinar el grado de efecto de estas variables, identificar las relaciones que existen entre las mismas o incluso determinar cuáles son.

No existe una definición globalmente aceptada de qué es la **Inteligencia Computacional (IC)**. Dependiendo del autor, se la considera desde un sinónimo de la **Inteligencia Artificial (AI)** hasta un campo completamente diferenciado.

En esta tesis hablaremos desde el punto de vista mayoritario, la **IC** como rama de la **AI** que agrupa todas aquellas técnicas que tratan de aprender soluciones a problemas a través del análisis de información presente en conjuntos de datos, en el entorno o ambos.

Por ello, la primera sección del capítulo ofrecerá una visión histórica de la aparición del concepto para justificar el por qué de esta definición. El resto del capítulo introducirá la importancia que tiene el aprendizaje dentro de este área, describirá las técnicas de **Redes Neuronales Artificiales (ANNs, Artificial Neural Networks)** y la **Lógica Difusa** necesarias para sentar las bases del posterior desarrollo de la tesis y dará nociones de qué son los agentes y por qué este punto de vista es útil para nosotros.

De Inteligencia Artificial a Inteligencia Computacional

Es difícil precisar el comienzo del interés del ser humano por la emular la inteligencia humana. Los silogismos en la antigua grecia para modelar el conocimiento como reglas, los autómatas mecánicos de los filósofos modernos (siglos XVII al XIX) donde los cuerpos vivos son como un reloj o la electricidad (siglos XIX y XX), capaz de animar constructos son sólo ejemplos de cómo cada nuevo avance en la ciencia ha ido acompañado de un intento de simular el cuerpo y mente humanos.

Podemos aventurarnos a decir que a principios del siglo XX se comienza a gestar el área de la **AI** con los trabajos relacionados con los principios del **conexionismo**⁴. Estas ideas saltaron al mundo

⁴ El enfoque del **conexionismo** postula que tanto la *mente* como el *conocimiento* son comportamientos complejos que emergen de redes formadas por unidades sencillas (i.e. neuronas) interconectadas. 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]).

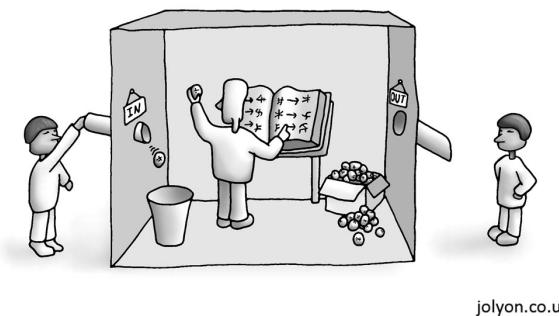
Figura 2: La *Habitación China* de John Searle es un experimento mental por el que se trata de demostrar la invalidez del Test de Turing. Partiendo de un Test de Turing donde la máquina ha aprendido a hablar chino. Reemplazamos la máquina por un humano sin idea de chino pero con un manual de correspondencias de ideogramas. Cuando una persona le manda mensajes en chino, esta otra responde usando el manual, por lo que podemos afirmar que la persona, y por tanto la máquina, no saben chino.

⁵ Muchos autores prefieren nombrar este hito, junto con el trabajo “*The organization of behavior*” [Hebb, 1949] de Donald O. Hebb como el punto de partida del área debido a su connotación computacional

⁶ El **Test de Turing** es una metodología para probar si una máquina es capaz de exhibir comportamiento inteligente similar al del ser humano. En ella, dos humanos (H_1 y H_2) y una máquina (M) están separados entre sí pero pudiendo intercambiarse mensajes de texto. H_2 envía preguntas a H_1 y M y éstos le responden. Si H_2 no es capaz de identificar qué participante es la máquina, se puede concluir que ésta es inteligente.

⁷ El concepto de “pensar” es un tema controvertido incluso en el propio ser humano: ¿es 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. Un ejemplo es el experimento de la habitación china (Figura 2), donde se demuestra la invalidez argumentando que la máquina ha aprendido a realizar acciones sin entender lo que hace y por qué lo hace. Sin embargo, ¿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?.

⁸ 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 y el 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.



jolyon.co.uk

de la computación hacia mediados del siglo XX, cuando Warren S. McCulloch y Walter Pitts publicaron su trabajo “*A logical calculus of the ideas immanent in nervous activity*” [McCulloch and Pitts, 1943]⁵, donde se describe el primer modelo artificial de una neurona.

El trabajo suscitó tanta expectación que se comenzó a especular sobre la posibilidad de emular (una vez más) la inteligencia humana en máquinas. Uno de los resultados fue la publicación de un artículo por parte de Alan Turing que comenzaba con la frase “*Can machines think?*” [Turing, 1950], introduciendo el famoso Test de Turing⁶ por el que el autor pretendía establecer una metodología para determinar si una máquina podía ser considerada inteligente y por tanto podría llegar a pensar⁷.

Pocos años después de la publicación del artículo, en el año 1956, se celebró la **Conferencia Dartmouth** [McCarthy et al., 1956]. En ésta, el tema de la conferencia fue la pregunta del artículo de Turing, y el área nació con entidad propia tras acuñarlo John McCarthy como **Inteligencia Artificial**.

A partir de este momento, la investigación en el área recibió mucha atención por parte de investigadores y gobiernos. Después de todo era un área nueva, muy prometedora y con mucho trabajo por delante. Tras su nacimiento el campo comenzó a dar resultados, pero la expectación y las promesas no permitían ver que los resultados se obtenían en problemas relativamente simples, muy formales y en general estériles, donde en realidad no era necesaria demasiada información para generar un conocimiento del entorno en el que los modelos se movían.

Dado que los estudios en el área estaban dominados por aquellos relacionados con las ideas del conexionismo, la publicación del libro “*Perceptrons*” [Minsky and Papert, 1969] de Marvin Minsky y Seymour Papert en 1969 supuso un notable varapalo para las investigaciones. En él se expusieron las limitaciones de los modelos de **ANNs** desarrollados hasta la fecha, y el impacto fue de tal envergadura que la investigación en el área se abandonó casi por completo. Concretamente el conexionismo prácticamente desapareció de la literatura científica durante dos décadas. Es lo que se conoce como el primer **AI Winter**⁸.

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 de nuevo los enfoques conexionistas, debido en gran parte a 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 empezaron a perder interés frente al nuevo avance del conexionismo ⁹. Esta época se suele identificar como el segundo *AI Winter*, ya que tanto la investigación como las inversiones en el área de *Sistemas Expertos* disminuyeron. Sin embargo, el efecto no fue ni mucho menos equiparable al de el primero.

Junto con el resurgir del conexionismo, otras técnica alineadas como la *Lógica Difusa* o los *Algoritmos Genéticos (GAs, Genetic Algorithms)* también ganaban popularidad, y entre ellas retroalimentaban los exitos gracias a sus sinergias. Esto provocó una explosión de terminologías para diferenciar las investigaciones en curso 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 durante estos años. En el nacimiento del campo, se buscan literalmente máquinas que piensen como humanos, o al menos seres racionales, con mente. Con el paso de los años, el área 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)*, *Sistema de Recomendación (RS, Recommender System)*, 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 *IC* es un campo diferenciado de la *AI*.

Podemos definir la *Inteligencia Computacional* como la “rama de la *AI* que aporta soluciones a tareas específicas de forma inteligente a partir

⁹ Esto, evidentemente no sentó bien a los autores prolíficos en *Sistemas Expertos*. Mientras que el enfoque en estos sistemas es el clásico en la computación, donde los problemas (en este caso el conocimiento experto) son resueltos mediante operaciones sobre un lenguaje de símbolos, el enfoque del conexionismo postula que la mente, el comportamiento intelectual, emerge de modelos a más bajo nivel. Por ello, algunas voces se alzan 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.

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.

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].

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

¹⁰ <http://cis.ieee.org/>

Se puede establecer el año 1994 como en el que la **Inteligencia Computacional** nace formalmente como área, coincidiendo con el cambio de nombre del *IEEE Neural Networks Council* a *IEEE Computational Intelligence Society*¹⁰. 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 **IC**."

Durante estos años ganaba popularidad también el concepto del **Soft Computing** en contraposición con el **Hard Computing**. El **Soft Computing** engloba las técnicas que buscan resolver problemas con información incompleta o con ruido. Debido a que el conjunto de técnicas definidas como constituyentes del **Soft Computing** son las mismas que se usan en la **IC** algunos autores consideran ambos términos equivalentes. Nosotros consideramos que el **Soft Computing** es un punto de vista de la computación a diferencia de la **IC**, la cual es un área de específica dentro de la **AI** hace uso de métodos incluidos en el concepto **Soft Computing**.

Hoy en día la **IC** es un área con muchas aplicaciones prácticas en una variedad muy distinta de campos de la ciencia y con muchos temas de investigación por explorar. Por ello, esta tesis pretende la exploración de una parte concreta de este área en el tema del modelado de comportamiento humano en el problema de la conducción.

Hard Computing y **Soft Computing** son la forma de referirse a la computación convencional frente al **Soft Computing**. El **Hard Computing** 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 **Soft Computing** 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 rol del aprendizaje en la Inteligencia Computacional

El cambio más notorio entre los dos puntos de vista de la **AI** tradicional y la de la **IC** es el concepto de entrenamiento, es decir, pasar de “desarrollar un programa para resolver un problema” a “entrenar un modelo para que aprenda la solución”. Éste es el concepto de **aprendizaje**, y al proceso de ajuste del modelo a la solución buscada **entrenamiento**.

Las técnicas de aprendizaje se clasifican dependiendo de la forma en la que entrena los modelos. Podemos identificar tres clases principales de técnicas de entrenamiento las cuales se describen a continuación:

- **Aprendizaje supervisado.** Supongamos que disponemos de un modelo denotado por $M_V(V, I) = O$ donde V es el conjunto de variables de determinan el comportamiento de M_V y donde I es un conjunto de valores (características) de entrada para las que el modelo obtiene un conjunto O de valores de salida. Entonces, la forma de entrenar al modelo, es decir el *algoritmo de entrenamiento* se encargaría de, a partir de 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 , modificar los valores de las variables del conjunto V para ajustar lo más posible O_i a O dado I_i .
- **Aprendizaje no supervisado.** Es el proceso por el cual un modelo aprende a partir de datos en brutos sus relaciones y extrae patrones, sin saber qué son esos datos ni recibir supervisión (a diferencia del aprendizaje supervisado donde los datos incluyen un valor de entrada y su salida correspondiente). En general, los algoritmos pertenecientes a esta categoría se dedican al problema del *clustering*, es decir, identificar grupos de elementos cercanos en el espacio basándose en la suposición de que el comportamiento de dos elementos es más parecido cuanto más cerca están el uno del otro. Algunos ejemplos de técnicas que basan su entrenamiento en un esquema no supervisado son los mapas autoorganizados (SOM), los autoencoders o las redes de creencia profunda (DBN de Deep Belief Network).
- **Aprendizaje por refuerzo.** En este paradigma, el algoritmo ajusta el modelo de acuerdo a políticas de recompensa o penalización en función de lo bien o lo mal que el modelo está desempeñando la tarea de acuerdo a una métrica determinada.

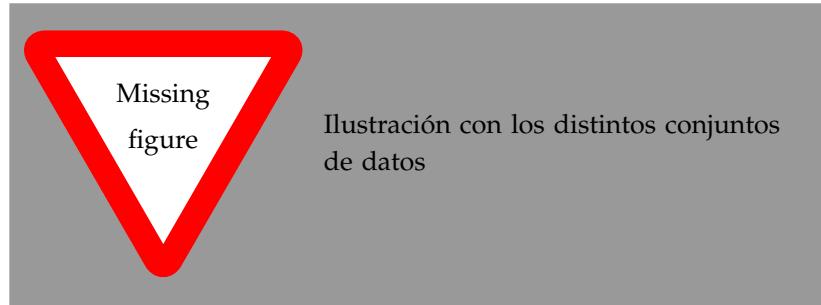
Algunos autores hacen uso de técnicas pertenecientes a ambos paradigmas en forma de aproximación híbrida para suplir deficiencias u optimizar/acerlar el aprendizaje. Un claro ejemplo lo podemos ver en [Hinton, 2006], donde los autores hacen uso de *autoencoders* como técnica no supervisada para la inicialización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

Esta tesis se dedica al modelado de comportamiento entrenando a partir de datos reales de conducción, por lo que el discurso se centrará únicamente en el esquema de aprendizaje supervisado. En él, los algoritmos de entrenamiento dependen del modelo a usar (no es lo mismo un algoritmo de entrenamiento para una red neuronal recurrente que para un perceptrón multicapa), y suelen ser usados principalmente para la solución a problemas de **clasificación** (i.e. determinar si un elemento dada sus características pertenece o no a determinado conjunto) y de **regresión**, esto es, ajustar las salidas de un modelo para ajustarse lo máximo posible al valor real del sistema modelado.

Epochs, conjuntos de entrenamiento, test y validación

La terminología general que se usa en la **IC** no es demasiado compleja (con la salvedad de los nombres de técnicas y algoritmos). A continuación se resumen los más importantes:

Figura 4: Los diferentes tipos de conjuntos de datos existentes.



Conjunto de datos o dataset Se refiere al conjunto total de ejemplos o datos del que disponemos. Se presupone que el dataset mantiene una distribución de datos similar al entorno real sobre el que pretendemos trabajar, lo cual puede no ser posible por multitud de factores.

Conjunto de entrenamiento El conjunto de entrenamiento (*training set* en inglés) se refiere al conjunto de datos que se usará para entrenar al modelo. Se corresponde un subconjunto del dataset, y es necesario que mantenga una distribución de datos lo más similar posible al dataset original para poder garantizar que el modelo entrenado represente a todos los sus datos.

Epoch Un epoch se suele referir a una iteración sobre todo el conjunto de ejemplos del conjunto de entrenamiento. Sin embargo, en la actualidad estos conjuntos pueden llegar a ocupar demasiado por lo que, dependiendo del contexto, un *epoch* se puede referir a una iteración sobre una porción del conjunto total de entrenamiento.

En otros contextos la definición de *epoch* se mantiene y a cada porciones se las denomina *batch* o *mini-batch*.

Conjunto de validación Es el subconjunto de datos del dataset principal destinado a probar la efectividad del modelo. Al comienzo del entrenamiento, se reserva este conjunto y a lo largo del entrenamiento se valida contra él para comprobar la efectividad del modelo entrenado.

Para una correcta validación es muy importante que, como el conjunto de entrenamiento, éste mantenga una distribución de ejemplos lo más similar posible al conjunto de datos original para evitar sesgos.

Conjunto de test Durante el proceso de entrenamiento, es una práctica habitual separar el conjunto de entrenamiento en dos partes, una más grande que se usará para el entrenamiento en el actual *epoch* y otra más pequeña que se usará para validar el modelo **durante el proceso de entrenamiento**¹¹.

Existen autores que prescinden de este conjunto y trabajan directamente con conjuntos de entrenamiento y validación.

¹¹ Para evitar sesgos, un buen conjunto de test debe ir cambiando constantemente entre epochs. Debido a esto, el modelo acabará aprendiendo también el conjunto de test y por tanto sus resultados no nos sirven para validar la efectividad del modelo entrenado.

Problemas del aprendizaje en la Inteligencia Computacional

El entrenamiento de modelos en la IC adolece de dos principales problemas que, además, no tienen una solución general (*non free-lunch theorem* [Wolpert and Macready, 1997]) ya que dependen tanto de la estructura de los datos sobre los que vamos a tratar como de los hiperparámetros¹² del modelo. Estos son la sobreespecialización y la subespecialización¹³.

El objetivo de un entrenamiento es conseguir modelos lo suficientemente buenos para que aprendan a generalizar sobre datos no conocidos, pero sin fallar demasiado. Cuando un modelo sufre de **sobreespecialización**, es porque aunque ha aprendido los ejemplos, falla a la hora de generalizar (podemos decir que ha aprendido los ejemplos *de memoria*). El caso contrario, la **subespecialización**, pasa cuando el modelo no es lo suficientemente complejo como para aprender el problema y por lo tanto generaliza demasiado. Hablaremos de casos concretos y soluciones más adelante.

¹² En general, hablaremos de *parámetros* cuando nos referimos a valores que son inherentes al modelo (e.g. en una ANN, los valores de los pesos) y de *hiperparámetros* cuando nos referimos a aquellos parámetros que modifican los elementos que modifican los parámetros (e.g. en una ANN, el factor de aprendizaje).

¹³ En la literatura también se habla de *high variance* o *over-fitting* y de *high bias* o *under-fitting* para referirse a la sobreespecialización y subespecialización respectivamente.

Deep learning

La tónica general en la ciencia es que cada pocos años algún término se escape del mundo de la investigación y se convierta en la palabra de moda que definirá los eslóganes de todas las empresas del ámbito en el que se mueven. Palabras con fuerza como Big Data, Cloud Computing, Web Services o SaaS que visten muy bien logos de

corporaciones y frases de consultores vendehúmos, pero que lo más normal es que sean conceptos ya existentes como proceso de datos masivos, computación distribuida o ejecución remota.

Al margen de esta pequeña crítica, sí es cierto que en ocasiones estas palabras captan sutilezas o información que las hace más adecuadas que los antiguos conceptos y que incluso pueden llegar a abrir en el futuro nuevas ramas dentro del área al que pertenecen. Un ejemplo podrían ser los *sistemas de recomendación*, un caso particular de *sistemas de filtrado de información* cuyo nombre estuvo de moda durante unos años y que en la actualidad conserva su entidad como una subrama de la rama principal.

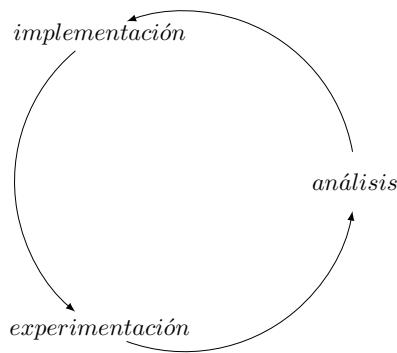
El *deep learning* es una de las palabras con la que últimamente se inunda la prensa, pero lo cierto es que desde la aparición del término, parece que los avances dentro de éste no tienen límites. En esta tesis se considera al *deep learning* a la nueva generación de enfoques en la que confluyen varios factores que han hecho posible una mejora sustancial en el entrenamiento y la operación de modelos con técnicas que, por otro lado, ya existían previamente. Estos factores son los siguientes:

- **Disponibilidad de datos.** En la última década, la cantidad de datos que generamos como especie ha crecido en muchos órdenes de magnitud. El abaratamiento de los costes de producción de dispositivos y de sensores o la acumulación temporal de los datos históricos son sólo dos factores que nos permiten en la actualidad el acceso a una cantidad ingente de datos con la que trabajar, algo impensable en la década anterior.
- **Capacidad computacional.** Aunque obvio, es un factor también crucial. Una mayor capacidad computacional es directamente proporcional a una mayor velocidad en el ciclo de experimentación de modelos (ver Figura 5). El verdadero impacto de esta década ha sido el del uso de las GPU de las tarjetas gráficas como plataforma donde distribuir el cómputo.
- **Algoritmos más eficientes.** Nuevos elementos como las funciones de activación *Rectified Linear Unit (ReLU)*, la representación de las redes como grafos computacionales para facilitar su distribución o innovaciones en los algoritmos de entrenamiento son algunas de las mejoras en este aspecto que redonda, como no, en la optimización de la capacidad computacional de las máquinas y por tanto sobre el ciclo de experimentación.

El adjetivo *deep*, en el contexto de las redes (e.g. redes neuronales, redes de creencia, ...), donde se origina este término, se refiere a una red con más capas de lo habitual¹⁴ (normalmente más de dos o tres). Este tipo de capas, históricamente ha sido más difícil de entrenar, debido entre otras cosas a las técnicas de entrenamiento (donde los parámetros tendían a diluirse según se aumentaba el número de

¹⁴ De aquí surge el nombre de **shallow network** o red superficial, en contraposición a **deep network** o red profunda.

Figura 5: Ciclo de aplicación de soluciones basadas en **Inteligencia Computacional**. Los sistemas inteligentes se conciben, se implementan y se prueba. En la fase de experimentación se determina cómo de bien o de mal lo está haciendo y cuales son las decisiones a tomar para el nuevo ciclo. Cuandto más rápido se puede realizar este ciclo, más soluciones se pueden probar, por ello el impacto de la mayor capacidad computacional y la optimización de las técnicas de entrenamiento es tan importante.



capas entre la entrada y la salida) o a la disponibilidad de conjuntos de datos, los cuales al no ser muy grandes, las redes grandes tendían a sobre especializarse. La Figura 6 introduce, con una ilustración un tanto informal, las capacidades del deep learning frente a las de los modelos entrenados antes de esta época.

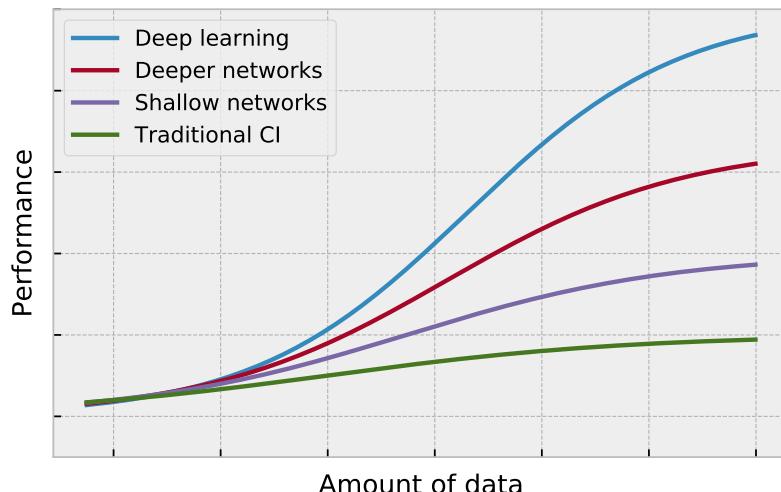


Figura 6: La enorme cantidad de datos junto con la capacidad computacional y la mejora de las técnicas de entrenamiento hacen posible que en la actualidad, con las técnicas asociadas al contexto del deep learning, los modelos entrenados sean más eficientes inteligentes. Imagen adaptada de la charla *How scale is enabling deep learning* de Andrew Y. Ng, accesible <https://youtu.be/LcfLo7YP804>.

Los factores antes mencionados han hecho posible la operación sobre redes más grandes y profundas con cantidades de ordenes de magnitud muy superiores. El resultado en la actualidad es que tenemos modelos que mejoran mucho los modelos anteriores, y no parece haber cota superior en su capacidad de aprendizaje ¹⁵.

Artificial Neural Networks

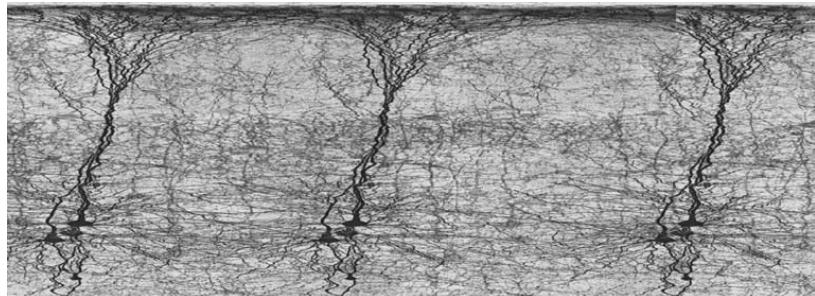
Son herramientas que tratan de replicar las funciones cerebrales de un ser vivo de una manera muy fundamental, esto es, desde sus componentes más básicos, las neuronas. Para ello se basan en estudios de neurobiología y de ciencia cognitiva moderna del cerebro humano ¹⁶.

Una ANN es independiente del problema a solucionar. Se la puede

¹⁵ Dentro de un contexto de aplicación específica, creemos que aún estamos muy lejos de crear vida inteligente.

¹⁶ 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 7) hace que, tecnológicamente hablando, nos quedemos con la sensación de estar aún a años luz de aproximarnos a la complejidad de un cerebro humano.

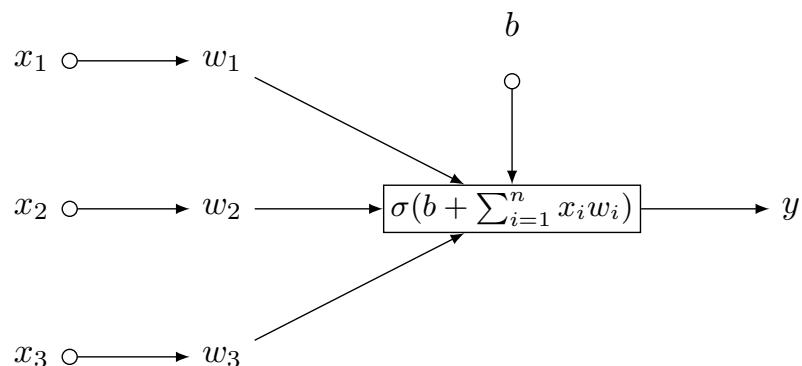
Figura 7: Sección del neocórtex humano, región asociada a las capacidades cognitivas y que supone alrededor de un 76 % del volumen total del cerebro humano. Está distribuido en 6 capas y miles de columnas que las atraviesan, cada una con alrededor de 10,000 neuronas y un diámetro de 0,5mm. Como dato 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 EPFL, <http://bluebrain.epfl.ch/>.



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 8 ([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 por los que fluye la información de manera unidireccional, donde algunas de dichas unidades sirven de entrada al sistema (i.e. entradas o sensores), otras sirven de salida del sistema (i.e. salidas y actuadores) y otras como elementos internos (i.e. ocultas), y donde los pesos de sus conexiones se ajustan mediante un proceso denominado *entrenamiento*, imitando los principios de la teoría hebbiana [Hebb, 1949].

Figura 8: 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.



Este primer modelo de neurona proponía una función escalón para determinar si la neurona se activaba o no, como analogía del funcionamiento de la neurona artificial. Sin embargo, este modelo es muy limitado. La verdadera potencia de las redes surge del tanto del uso de funciones de activación no lineales como de la agrupación de estas neuronas en estructuras más complejas.

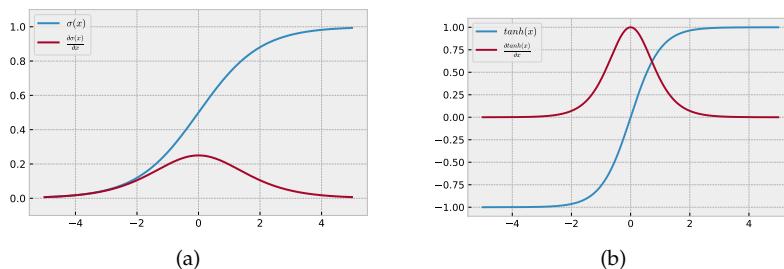
Funciones de activación

El valor de salida de una neurona queda determinado por la aplicación de una función sobre la entrada neta a ésta. Esta función dictamina el grado de activación de la neurona y para un correcto funcionamiento de la red en términos generales debe ser no lineal¹⁷.

Las funciones de activación clásicas usadas en redes neuronales han sido la función sigmoidal (eq. 1) y la tangente hiperbólica (eq. 2). Por un lado, son funciones no lineales que mantienen normalizados las activaciones de las neuronas, y por otro, son derivables a lo largo de todo el dominio de los reales, siendo su derivada además fácilmente computable. En la Figura 9 se muestra una representación gráfica de éstas funciones junto con su derivada.

$$\sigma(x) = \frac{1}{1 + e^{-x}} \quad \frac{d\sigma(x)}{dx} = \sigma(x) \cdot (1 - \sigma(x)). \quad (1)$$

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

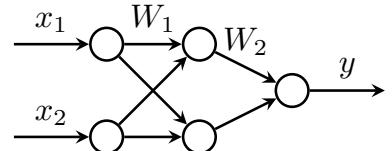


En general, la tangente hiperbólica es superior a la sigmoidal en todos sus aspectos. Computacionalmente es menos costoso el cálculo de una función sigmoidal, pero con la potencia de cómputo actual esta diferencia puede considerarse despreciable. Además de tener una forma similar a la sigmoidal, la tangente hiperbólica permite enviar señales de activación negativas. Además, tiende a centrar los datos de una capa a otra en lugar de mantener un sesgo hacia el 0,5 (recordemos que mientras que la sigmoidal está definida en el intervalo (0, 1), la tangente hiperbólica se encuentra definida entre los intervalos (-1, 1). Por tanto, en un caso general, la tangente hiperbólica suele ser preferible a la sigmoidal.

Aún así, en algunas situaciones sí podría tener sentido el uso de funciones sigmoidales en lugar de tangentes hiperbólicas. Por ejemplo, en un problema de clasificación, mantener en la capa de salida funciones de activación sigmoidales permite ajustar la salida en el intervalo (0, 1), sin necesidad de realizar una posterior normalización.

Sin embargo, el principal problema de estas funciones es cuando los valores netos de las entradas son muy grandes. En ese caso, las funciones se acercan a los extremos, aproximándose sus gradientes

¹⁷ Supongamos una red neuronal con una estructura como la siguiente:



Supongamos, además, la función de activación de las neuronas es lineal (e.g. $f(x) = x$). La salida se puede expresar como:

$$\begin{aligned} \mathbf{y}^1 &= f(W_1 * \mathbf{x} + \mathbf{b}_1) \\ y &= \mathbf{y}^2 = f(W_2 * \mathbf{y}^1 + \mathbf{b}_2) \end{aligned}$$

Por tanto:

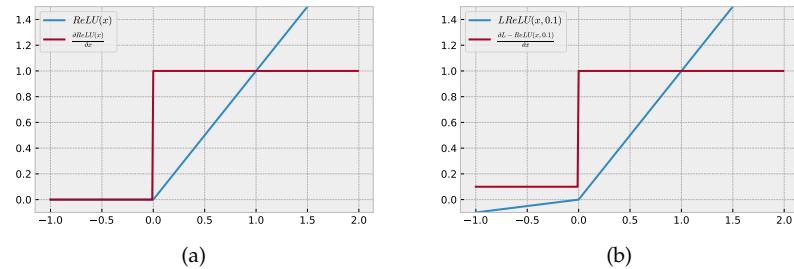
$$\begin{aligned} y &= f(W_2 * f(W_1 * \mathbf{x} + \mathbf{b}_1) + \mathbf{b}_2) \\ &= (W_2 * W_1) * \mathbf{x} + (W_2 * \mathbf{b}_1 + \mathbf{b}_2) \end{aligned}$$

Figura 9: Las funciones de activación sigmoidal y tangente hiperbólicas. Ambas se han usado como funciones de activación. Siendo \mathbf{x} el vector columna de salida de la capa anterior, \mathbf{b} el vector de los bias de la capa actual y \mathbf{W} el vector columna de las entradas.

Es decir, usando funciones lineales da igual el número de capas que tengamos, ya que la composición de dos funciones lineales es siempre una función lineal y la arquitectura se reducirá a una única capa de activación lineal. Por ello no tiene demasiado sentido el uso de funciones lineales en las capas ocultas de una red.

a 0, y por tanto frenando el aprendizaje. Este error es denominado frecuentemente como *vanishing gradient* en la literatura. Es una de las razones por las que en la fase de inicialización de una red neuronal se tendía a valores en torno al 0 en los pesos. Unos valores altos hacían que las entradas netas fuesen muy grandes ralentizando el aprendizaje.

Figura 10: La función de activación (a) **ReLU** evita el problema del estancamiento cuando la entrada neta de la neurona es muy alta. La función de activación **Leaky ReLU** (en este ejemplo, con $\epsilon = 0,1$) es una de las posibles soluciones cuando se permite que la entrada neta a la red sea menor que 0, ya que en el caso de la función **ReLU** la derivada es 0 y por tanto el gradiente no nos indica hacia dónde ha de descender el error.



Uno de los avances dentro del Deep Learning (ver ??) fue el uso de un tipo de función de activación denominada **Rectified Linear Unit (ReLU)** (eq. 3). Ésta posee una forma muy simple pero es increíblemente efectiva. Por un lado, evita el problema del *vanishing gradient* dado que su derivada es constante en el intervalo $(0, \infty)$. Por otro, su cálculo es tremadamente simple comparado con el resto de funciones (no deja de ser un máximo). En la figura 7

$$ReLU(x) = \max(0, x) \quad \frac{dReLU(x)}{d(x)} \approx \begin{cases} 0 & \text{if } x < 0 \\ 1 & \text{if } x \geq 0 \end{cases} \quad (3)$$

$$L - ReLU_\epsilon(x) = \max(0, x) \quad \frac{dL - ReLU_\epsilon(x)}{d(x)} \approx \begin{cases} \epsilon & \text{if } x < 0 \\ 1 & \text{if } x \geq 0 \end{cases} \quad (4)$$

Este tipo de neurona tiene una serie de características que merece la pena comentar:

- No existe derivada en 0. El aprendizaje, basado en el descenso del gradiente, se encuentra con una indeterminación en 0. Sin embargo, es fácilmente subsanable incluyendo el valor 0 o 1 en la derivada en el punto 0. Aunque no es matemáticamente correcto, computacionalmente se está reemplazando la derivada por una función muy aproximada al ésta en la que el algoritmo de descenso del gradiente se comporta de forma muy similar. Por ejemplo, la representación en la Figura 10, la derivada es 1 en el punto $x = 0$.
- Sin límite superior y derivada 0 para $x < 0$. Estas características pueden ser una ventaja o una desventaja. Las activaciones muy fuertes representan relaciones entre elementos muy próximos entre sí, aunque tienen el riesgo de anular el impacto del resto de entradas, pudiendo ralentizar el aprendizaje. Por otro lado, en el

momento que el gradiente de una neurona se hace 0, ésta “*muere*”, pudiendo ser una desventaja ya que la red dispone de menos neuronas para representar un modelo, como una ventaja, al ajustarse el modelo al número de neuronas necesarias. En general, las desventajas en estos aspectos aparecen cuando el factor de aprendizaje es notoriamente alto.

Por último, la función de activación *Leaky ReLU* (eq. 4) es una evolución de la *ReLU* para el uso en problemas donde es ventajoso que una neurona no llegue a tener nunca un gradiente de 0. Van acompañadas de un parámetro $\epsilon \approx 0$ que determina la pendiente para todos aquellos valores menores de 0. Un ejemplo de esta neurona se ilustra en la figura 10. Su uso no está muy extendido, pero existen casos en los que su uso está justificado.

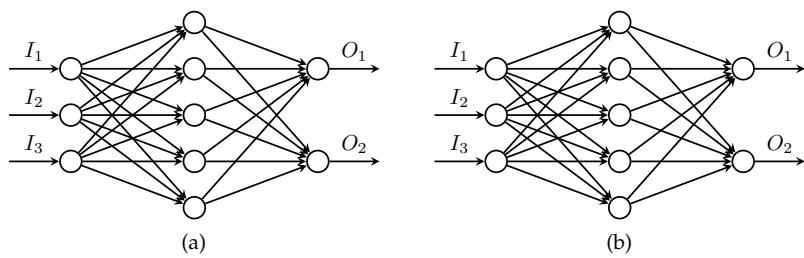
Estructura y clasificación de redes neuronales

Las *ANN* reciben ese nombre debido a que son sistemas formados por multitud de neuronas artificiales simples. Dependiendo de cómo su topología y configuración, éstas serán más adecuadas para unos u otros problemas (e.g. unas serán más adecuadas para regresión, clasificación, predicción, *clustering*).

Típicamente, estas redes se componen de neuronas conectadas, por lo que se puede pensar en ellas como un grafo ponderado donde los nodos se corresponden a las neuronas, las aristas a las conexiones entre entradas y salidas y los pesos de las aristas a los pesos de las conexiones de entrada. Existen diferentes topologías o arquitecturas dependiendo de qué forma toma el grafo que modela las neuronas y sus conexiones. Estos dos tipos son los siguientes:

- Redes **feed-forward** (o prealimentadas). Sus representación como grafo no presenta ningún ciclo (por tanto ninguna retroalimentación entre neuronas, como se ilustra en la figura 11). Es la topología más usada en aplicaciones prácticas debido a su sencillez y su efectividad. En ellas el flujo de información sigue un camino desde un conjunto de neuronas denominadas *entradas* hasta otro conjunto de neuronas denominado *salidas*. No es requisito que las neuronas se agrupen en capas, aunque suele ser la estructura común. A las redes de más de dos/tres 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 suele denominar *profundas* o *deep*. Representantes clásicos de esta categoría pueden ser el *Perceptrón Multicapa* [Rumelhart et al., 1985], los autoencoders [Hinton, 2006] o los *Mapas Auto-Organizados (SOM)* [Kohonen, 1998].
- Redes **recurrentes**. Éstas, a diferencia de las prealimentadas, tienen al menos un ciclo dentro de su representación, de tal manera que el flujo de información de salida de una neurona puede llegar a

Figura 11: Diferencias entre los grafos que representan las ANNs de tipos (a) *feed-forward* y (b) recurrentes. Las redes recurrentes presentan ciclos entre sus nodos que permiten la retroalimentación interna entre las neuronas. Suelen ser más útiles a la hora de modelar eventos en el tiempo, aunque su entrenamiento es más complejo.**TODO!**CAMBIAR LA FIGURA DE LA RECURRENTE



afectar a su propio estado. Aunque estas topologías representan de una forma más fiel las bases biológicas de las ANN, históricamente han sido más complejas a la hora de operar y entrenar debido a sus relaciones entre nodos. Sin embargo, durante la última década han aparecido nuevas técnicas que facilitan su operación. Algunos casos particulares de este tipo de arquitectura son las Redes de Hopfield [Hopfield, 1982] o las redes Long-Short Term Memory (LSTM) [Hochreiter and Schmidhuber, 1997].

Esta tesis se centra en redes pertenecientes al primer grupo, concretamente en las topologías Perceptrón Multicapa y Convolutional Neural Network (CNN), con las que se cerrará la presente sección. Ambos tipos de redes han probado su efectividad en diferentes dominios, aunque las segundas están demostrando su efectividad con las nuevas técnicas surgidas a partir del *deep learning*.

Perceptrones multicapa

En un perceptrón multicapa, las neuronas se encuentran agrupadas en capas de tal manera que todas las salidas de las neuronas de una capa se conectan a todas las entradas de cada una de las neuronas de la capa siguiente. A las primera y últimas capas de la red se las denomina respectivamente capa de entrada y de salida, mientras que las capas intermedias son denominadas capas ocultas. Hemos visto algunas ilustraciones a lo largo del presente capítulo, pero en la Figura ?? se puede ver en detalle este tipo de red.

La forma de calcular la salida de un perceptrón multicapa es la siguiente. Supongamos que tenemos dos capas, $l - 1$ y l , compuestas por n^{l-1} y n^l neuronas respectivamente cada una con su función de activación f , y como conexiones (pesos) entre ambas capas tendremos una matriz W^l de dimensiones $(l - 1, l)$. Cada una de las neuronas deberá tener una entrada de bias, por lo que tendremos también un vector columna \mathbf{b}^l que los representará. La salida s^l de esa capa será un vector columna que se obtendrá a partir de la siguiente función que se muestra en la ecuación 5:

$$\mathbf{s}^l = f(W^l \mathbf{s}^{l-1} + \mathbf{b}^l) \quad (5)$$

Como se puede observar, la salida s^l de la capa l depende directamente de la salida de la capa $l - 1$. El proceso de calcular la salida es

incorporando las entradas en la capa de entrada e ir iterando hasta la capa de salida.

El algoritmo base de aprendizaje en un perceptrón multicapa de denomina *back-propagation* [Rumelhart et al., 1985] y se basa en la regla delta [Widrow and Hoff, 1960] para el perceptrón simple. Éste usa una técnica denominada *descenso del gradiente* donde lo que se intenta es minimizar el valor de una función $f(x)$ que representa el error (en realidad el *coste*¹⁸) de la red calculando como aumenta o disminuye esta con pequeñas variaciones de x .

El algoritmo trata de aplicar un error desde la salida de la red sobre todos los pesos de la misma en función de cuánto han colaborado en dicho error (bajo la suposición de que, cuando mayor es un error, más ha contribuido). El problema es cómo se calcula este error. En la última capa es sencillo (conocemos las salidas real y esperada), pero la genialidad del algoritmo es que el error en las interiores lo determina a partir del error de las sucesivas capas apoyándose en el descenso del gradiente. Es decir:

Supongamos que nos encontramos en la capa l para la cual conocemos el error de salida que denotaremos como el vector columna $\delta\mathbf{s}^l$, y donde todas las neuronas usan la función de activación f . Entonces, de acuerdo al gradiente, el error neto que produce nuestra salida se corresponde con:

$$\delta\mathbf{e} = \delta\mathbf{s}^l \circ f'(W^l \cdot \mathbf{s}^{l-1} + \mathbf{b}^l) \quad (6)$$

Nótese que \circ denota al producto de Hadamard (elemento a elemento). También es interesante fijarse en que $W^l \cdot \mathbf{s}^{l-1} + \mathbf{b}^l$ se corresponde al cálculo de la entrada neta de la capa l antes de aplicarle la función de activación f .

Una vez conocido este error podemos pasar a calcular cómo varían los parámetros de la capa (eq. 7b y 7c) y el error de salida (eq. 7a) de la capa anterior de la siguiente manera:

$$\delta\mathbf{s}^{l-1} = W^{l\top} \cdot \delta\mathbf{e} \quad (7a)$$

$$\delta W^l = \delta\mathbf{e} \cdot \delta\mathbf{s}^{l-1} \quad (7b)$$

$$\delta\mathbf{b}^l = \delta\mathbf{e} \quad (7c)$$

El valor $\delta\mathbf{s}^{l-1}$ será el valor de entrada para la capa anterior, mientras que los valores δW^l y $\delta\mathbf{b}^l$ serán los gradientes calculados. La forma más común de aplicar el error es la que se muestra en las ecuaciones 8a y 8b:

$$W^l = W^l + \alpha \delta W^l \quad (8a)$$

¹⁸ Existen dos términos asociados al error en ANNs: el error y el coste, que en la literatura se denominan *loss* y *cost* respectivamente. El primero se refiere al error existente entre la salida de la red neuronal y la salida esperada de un ejemplo en concreto, mientras que el segundo se refiere al error sobre todo el conjunto de entrenamiento. Es de esperar que durante el proceso de entrenamiento este error baje.

Existen diferentes funciones para calcular el error y el coste de un modelo en concreto. En el caso del coste, lo más común es usar la media entre todo el conjunto de entrenamiento como:

$$C(M) = \frac{1}{m} \sum_{i=1}^m L(\hat{y}_i, y_i)$$

Donde M es el modelo actual, m el número total de ejemplos y L la función de error entre las salidas \hat{y} e y de cada ejemplo i . En algunos casos se usa la media ponderada para dar más importancia a determinados casos, pero no es lo común.

Sin embargo, en el caso del error, existen varias funciones dependiendo de cuál sea la tarea. Por ejemplo, para tareas de regresión, lo común es usar el error cuadrático medio o su raíz cuadrada. Para tareas de clasificación, una función de error muy útil que ha remplazado a la función *hit* (relación de aciertos-fracasos) es la entropía cruzada, que tiene la siguiente forma:

$$L(\hat{y}_i, y_i) = (1 - y) \log(1 - \hat{y}) - y \log \hat{y}$$

La razón es que en esta última se capturan sutilezas como lo cercano que ha estado un acierto. Por ejemplo, si nuestro objetivo es una salida de clasificación $(1, 1, 0)$ y tenemos dos modelos, uno que dice $(0, 9, 0, 9, 0, 6) \rightarrow (1, 1, 1)$ y otro que dice $(0, 6, 0, 6, 0, 6) \rightarrow (1, 1, 1)$, según la función *hiy*, ambos modelos funcionan igual mientras que según la entropía cruzada el primero funciona mejor que el segundo.

Curiosamente, la entropía cruzada también funciona bien en problemas de regresión, aunque es más sencillo interpretar los resultados de un *RMSE* y por tanto su uso no está extendido.

$$\mathbf{b}^l = \mathbf{b}^l + \alpha \delta \mathbf{b}^l \quad (8b)$$

Esta es la forma original del algoritmo de back propagation. Al valor α que aparece en las ecuaciones se le denomina factor de aprendizaje y como se puede apreciar, su valor determina lo rápido que cambian los pesos. Suele tomar valores entre 0,1 y 0,01, pero dependiendo de la evolución del entrenamiento y de la variación del algoritmo, éste puede llegar a tomar valores mucho más bajos. Algunas de las variaciones sobre el algoritmo inicial son las siguientes:

- **Momento de inercia** [Qian, 1999]. Al algoritmo se le añade un factor por el cual los movimientos en el mismo sentido durante sucesivos epochs se acumulan. De esta manera, el riesgo de caer en mínimos locales disminuyen al ser capaz el algoritmo de “saltar” pequeños baches.
- **Adagrad ??, RMSProp ?? y Adadelta** [Zeiler, 2012]. Los tres se basan en el mismo principio. Al factor de entrenamiento de la red se le añade la existencia de un factor de entrenamiento por cada peso de tal manera que se actualiza de forma diferente en función de la evolución de su gradiente individual.
- **Adam** [Kingma and Ba, 2014]. Este algoritmo combina los funcionamientos del momento junto con los del gradiente adaptativo del algoritmo RMSProp. Es el más utilizado en los últimos años.

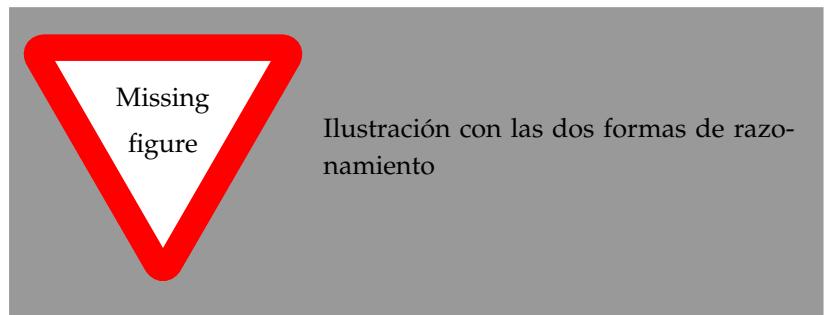
Redes de convolución

Tal y como su nombre indica, las **CNNs** son redes que usan convoluciones para su funcionamiento. Aunque también está estructurada en capas, su funcionamiento es diferente. Para comenzar, su estructura se puede dividir en dos regiones bien diferenciadas, una que se dedica a la extracción de características de la entrada (la denominaremos *extracción de patrones*) y otra que se dedica a la clasificación o regresión de la entrada a partir de las características extraídas (que denominaremos *predicción*). En la Figura 12 podemos ver una ilustración del esquema general de una **Convolutional Neural Network** donde se identifican ambas regiones.

La región de inferencia es en esencia un **Perceptrón Multicapa**. A éste le llega un conjunto de entradas deducidas en la región anterior y realiza la operación de regresión o de clasificación que le corresponda. El funcionamiento se explica en el apartado anterior dedicado a este tipo de redes. La región de extracción de patrones es más interesante y merece más contenido.

Convoluciones Una convolución es una operación matemática que funciona como filtro sobre una estructura espacial (en este contexto,

Figura 12: Estructura general de una [Convolutional Neural Network](#). Se pueden observar sombreadas de distinto color la relación existente entre las regiones de identificación de patrones y la de predicción. La primera actúa sobre la entrada extrayendo características consideradas relevantes mientras que la segunda traeja sobre esas características en lugar de sobre todo el espacio de entrada.



normalmente de matriz o de cubo) para identificar patrones y/o para transformar estructuras identificadas. Para simplificar la explicación en este apartado, supondremos que la entrada se trata de una matriz bidimensional (e.g. una imagen de un sólo canal de color), y que los filtros son también bidimensionales, pero es común tener espacios de entrada de más dimensiones¹⁹. La ecuación I describe el resultado de aplicar una convolución de un filtro sobre una matriz.

$$\begin{pmatrix} 8 & 7 & 4 & 7 \\ 0 & 1 & 1 & 3 \\ 3 & 4 & 7 & 1 \\ 5 & 2 & 5 & 0 \end{pmatrix} \circledast \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} = \begin{pmatrix} 9 & 8 & 7 \\ 4 & 8 & 2 \\ 5 & 9 & 7 \end{pmatrix}$$

El símbolo \circledast denota la operación de convolución. Esta operación hace recorrer el filtro por todas las posiciones hasta que recubre todo el espacio inicial²⁰. Para cada posición, se realizará el sumatorio del producto elemento a elemento, y el resultado se asignará en dicha posición. Esto implica que el tamaño de la matriz resultante es inversamente proporcional al tamaño del filtro. Concretamente, si (M_w, M_h) , (F_w, F_h) y (R_w, R_h) son el ancho y el alto para las matrices origen, filtro y resultado, se cumple que:

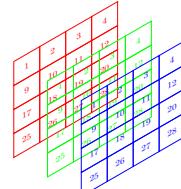
$$R_w = M_w - F_w + 1 \quad (9)$$

$$R_h = M_h - F_h + 1 \quad (10)$$

No obstante este funcionamiento de las convoluciones tiene un problema: las matrices tienden a ser cada vez más pequeñas con cada operación de convolución. En el contexto del *deep-learning*, el número de capas es finito y viene determinado por el tamaño del filtro.

La solución a este problema es la aplicación de una técnica denominada *padding* que aumenta la matriz de origen para que la matriz resultante tras la operación de convolución quede del mismo tamaño²¹. El cálculo del tamaño del padding para que la matriz resultado sea del mismo tamaño deberá ser:

¹⁹ De hecho, para analizar imágenes lo normal es trabajar sobre los diferentes canales de color como capas diferentes, por lo que una imagen es en realidad una matriz tridimensional de dimensiones (w, h, c) donde w es el ancho, h es el alto y c es el número de canales.



²⁰ La operación básica recorre las posiciones una a una. Existe un parámetro, denominado *stride* que puede modificar este comportamiento. Nosotros nos ceñiremos a un *stride* de tamaño 1×1 para simplificar el discurso.

²¹ En general, el padding genera la cantidad de celdas con o alrededor de la imagen para que a la hora de aplicarlo el resultado sea igual. Es una solución válida, pero dependiendo del problema puede interesar usar diferentes formas de padding. Por ejemplo, en nuestro problema, el cálculo de los valores de padding horizontal es diferente. En la parte dedicada al desarrollo de la tesis se explicará el cálculo y el por qué.

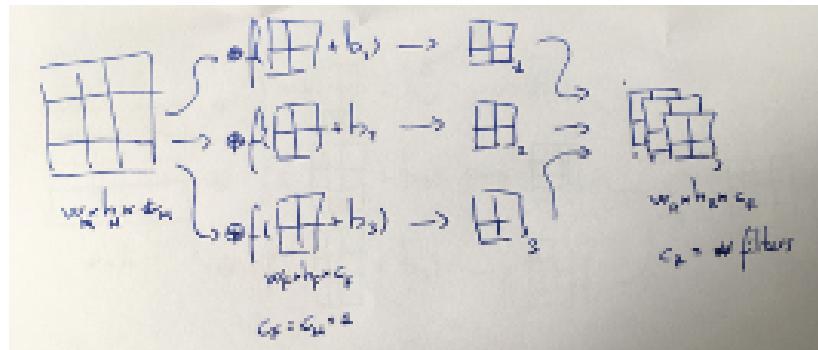
$$P_w = \frac{F_w - 1}{2} \quad (11)$$

$$P_h = \frac{F_h - 1}{2} \quad (12)$$

Por tanto es necesario que los filtros sean de dimensión impar.

Capas de convolución Las capa más importante en una CNN son las capas de convolución. Éstas se componen de un número variable de filtros que representan las características que queremos extraer de la matriz de entrada. La entrada a esta capa de convolución será la matriz a la que aplicar los filtros, y la salida será una no linearización sobre el resultado de la convolución, modificándose en el proceso la dimensión de la matriz resultado. La figura ?? ilustra esto.

Figura 13: Descripción de la capa de convolución. Como entrada se recibe una matriz tridimensional con una capa de profundidad. Los filtros han de tener la misma profundidad, mientras que la matriz resultado tiene como profundidad el número de filtros. A las convoluciones (más un bias) se les aplica una función no lineal como en los Perceptrón Multicapa. Por tanto, son los valores del filtro junto con los bias los parámetros que el algoritmo de aprendizaje debe aprender.



El proceso de aprendizaje en una red de convolución sin embargo no es trivial. Sigue apoyándose en el cálculo de los gradientes en función de los parámetros, que en este caso son los valores del filtro junto con un bias por cada filtro. Las ecuaciones de este cálculo en una capa bidimensional (para cada uno de los filtros) son las siguientes:

$$\delta W_f = \sum_{w=0}^{F_h} \sum_{h=0}^{F_w} S^{l*} \times \delta C_{w,h} \quad (13a)$$

$$\delta b_f = \sum_{w=0}^{F_h} \sum_{h=0}^{F_w} \delta C_{w,h} \quad (13b)$$

En estas ecuaciones, $\delta C_{w,h}$ se refiere al gradiente del coste respecto a la salida de la capa de convolución correspondiente al filtro f en la celda (w, h) . Así mismo, S^{l*} se corresponde con la región (slice) de la entrada que se usó para calcular el correspondiente $\delta C_{w,h}$.

Estas ecuaciones son más complejas en función de cómo varía el número de filtros y de dimensiones de los mismos, pero al final usa los mismos principios que los Perceptrones Multicapa.

Pooling y normalización Junto con las capa de convolución, normalmente se usan otros dos tipos de capa diferentes: las capas de *pooling* y las de *normalización*.

El *pooling* es una operación de discretización del espacio de entrada. El objetivo es reducir las dimensiones de las entradas para las siguientes capas. Esto mejora el coste computacional y ayuda contra la sobre-especialización, ya que hay menos parámetros sobre los que trabajar para la misma entrada.

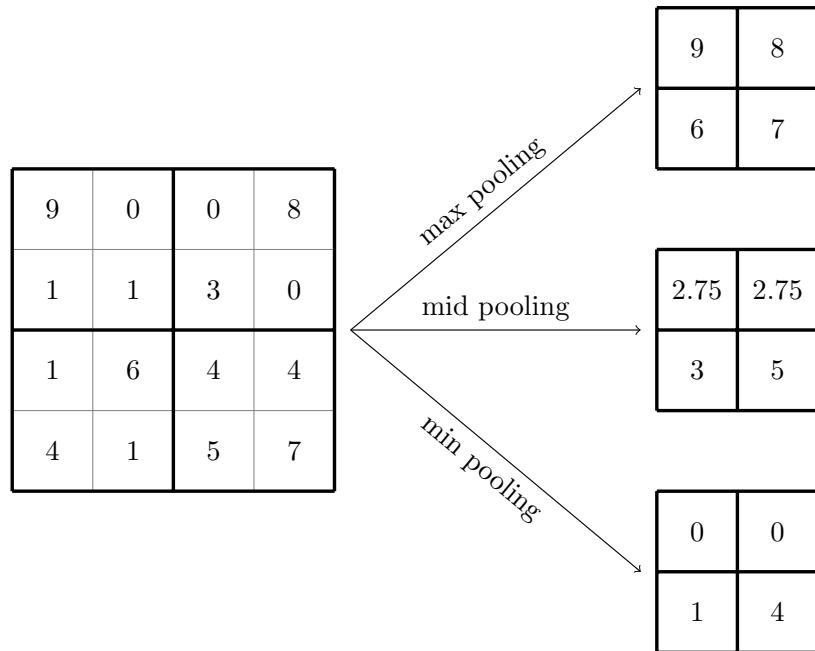


Figura 14: Ejemplo de las operaciones de *pooling* sobre una matriz de entrada de dos dimensiones. El filtro de tamaño 2×2 recorre la matriz en saltos de 2 celdas horizontales y 2 verticales extrayendo de cada región filtrada el mayor valor, el menor o el valor medio dependiendo de si la operación es *max-pooling*, *min-pooling* o *mid-pooling*.

Su funcionamiento es similar al de las convoluciones (después de todo se tratan también de filtros), con la diferencia de que se mueven en ventanas no solapables devolviendo un valor para cada ventana y reduciendo por tanto el tamaño de la entrada. Existen tres tipos fundamentales que son el máximo, el mínimo y la media, y su diferencia estriba en el cálculo del valor de salida a partir de los valores de la entrada. En la Figura ?? se ilustra un ejemplo de la operación de **max-pooling**²².

La normalización es otra operación, esta a nivel de capa, que reafirma las diferencias entre los valores existentes en la matriz, similar a un aumento de contraste en una imagen. Permite destacar diferencias y en general los resultados demuestran que acelera el proceso de aprendizaje de una CNN drásticamente cuando se incluye como capa oculta.

En los últimos años, los tipos más comunes de normalización son el **local response normalization (LRN)** [Robinson et al., 2007] y el **batch normalization** [Ioffe and Szegedy, 2015].

²² Es curioso que por un lado se inventen mecanismos para mantener los tamaños de entradas-salidas entre convoluciones y luego no se use el comportamiento de las convoluciones con un *stride* del tamaño del filtro para realizar esa reducción. De hecho existen algunos estudios que afirman que el uso del *pooling* conlleva un aumento computacional sobre el uso de convoluciones para reducir el tamaño sin implicar una pérdida de rendimiento a la hora de predecir. Un ejemplo de esto lo tenemos en [Howard et al., 2017] donde afirman:

"We find that max-pooling can simply be replaced by a convolutional layer with increased stride without loss in accuracy on several image recognition benchmarks"

Solucionando los problemas de entrenamiento

Anteriormente hablábamos de los problemas a los que se enfrentan los procesos de entrenamiento en la [IC](#). En líneas generales, los problemas de subespecialización pueden darse por tres razones diferentes:

- La red no es lo suficientemente grande. Puede ocurrir que el conjunto de datos requiera de más propiedades o parámetros. La solución es incrementar el número de capas o de neuronas por capa a fin de que el modelo acabe aprendiendo al menos los datos del conjunto de entrenamiento.
- El modelo no ha sido entrenado lo suficiente. Este caso es más raro, pero puede ser que el modelo sea lo suficientemente complejo para requerir más ciclos de entrenamiento. También puede ser que algunos hiperparámetros como el factor de entrernamiento o las funciones de activación están predisponiendo al modelo a aprender más lentamente.
- La topología del modelo no se adecúaa los datos. Puede ocurrir que el modelo que estamos tratando de aprender aprenda mucho mejor en una topologías que en otras. Por ejemplo, para aprender a clasificar imágenes, las [CNNs](#) funcionan mejor, entre otra serie de razones, porque mantiene una coherencia espacial entre los píxeles de la imagen desde el principio, cosas que con un [Perceptrón Multicapa](#) no ocurre. Quizá representar los parámetros de entrada de una forma diferente o probar otra topología puede hacer que aprenda el problema de forma diferente.

El caso de la sobreespecialización es quizá algo más complejo. Cuando un modelo está sobreentrenado falla al generalizar, aunque los datos del conjunto de entrenamiento estén perfectamente aprendidos. Suele ser causado por dos razones principales:

- No hay suficientes datos, por lo que el modelo no generaliza porque no sea capaz, sino porque todavía le queda por aprender. La solución suele ser aumentar la cantidad de datos que existen en el conjunto de entrenamiento.
- La red está sobredimensionada. Este suele ser el caso más común, y es que la red tiene tantos parámetros que al final a aprendido a predecir uno a uno casi todos los ejemplos del conjunto de entrenamiento. Existen dos posibles soluciones no excluyentes, la simplificación de la red y la aplicación de técnicas de regularización.

En el caso concreto de la regularización, el objetivo de esta técnica es tratar de penalizar o limitar el entrenamiento a través de la

inhibición de los parámetros. Existen diferentes técnicas para la regularización de parámetros, como las l_1 y l_2 (explicadas en detalle en [Ng, 2004]). En esta tesis se ha escogido una técnica de regularización denominada dropout [Srivastava et al., 2014].

En ésta, la idea es en cada epoch de entrenamiento del modelo, se desactivan una serie de neuronas de manera aleatoria. Estas neuronas no participan ni en la predicción ni en el posterior reajuste de pesos. Es muy fácil de implementar, computacionalmente muy eficiente y mejora sustancialmente el proceso de aprendizaje en redes que sufren de una alta especialización. La Figura ?? describe un ejemplo de funcionamiento en tres epochs de un perceptrón multicapa con una tasa de dropout del 0,5.

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 IC, sino que son simplemente una vía para representar las operaciones en forma de grafo, de tal manera que es fácil ver cuál es el flujo de los datos y las operaciones que se realizan sobre ellos.

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 15 describe un posible grafo computacional para un perceptrón simple.

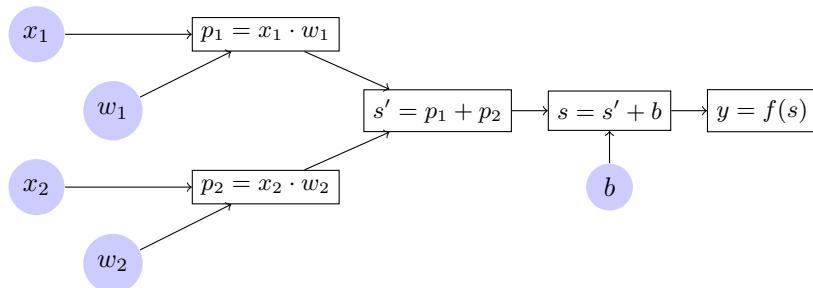


Figura 15: Ejemplo de grafo computacional para un perceptrón simple de dos neuronas de entrada y una de salida.

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 15 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 16 de derecha a izquierda).

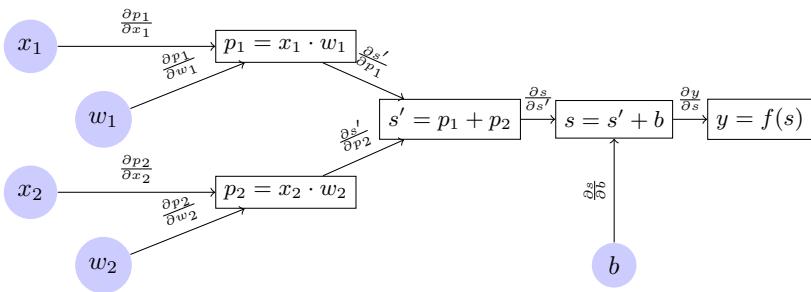


Figura 16: Es fácil representar las derivadas parciales sobre un grafo computacional y, a partir de ahí, obtener el efecto de las variaciones de una variable sobre el resto.

Com regla general, para conocer el gradiente de una variable a con respecto a otra variable b , se aplicaría la *regla de la cadena multivariable*, que es equivalente a sumar todos los posibles caminos que van de a a b del grafo, multiplicando las derivadas parciales de cada arista.

²³ 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.

²⁴ Las lógicas multivaluadas son aquellas donde las premisas pueden tomar más de dos valores.

Lógica Difusa

La lógica matemática ²³ (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. La lógica difusa se puede considerar como perteneciente al conjunto de lógicas multivaluadas ²⁴, donde se trabaja con este tipo de conceptos.

La idea tras este punto de vista es que raramente el ser humano piensa en términos absolutos o completamente definidos. En su lugar, su forma de razonar conlleva una serie de abstracciones que diluyen el carácter de las observaciones que a su vez pueden ser también imprecisas. Ya que en la lógica clásica es imposible expresar la complejidad que implica la incertidumbre de este proceso de razonamiento, en la **Lógica Difusa** los conceptos de verdad o mentira se relajan, existiendo una transición entre estados no abrupta, sino suave y progresiva.

A lo largo de la sección se describirá el concepto de conjunto difuso, cómo se usan como mecanismo de razonamiento y su utilidad dentro del área de sistemas de control.

Ampliando la teoría de conjuntos tradicional

La teoría de conjuntos difusos nació a mediados del siglo XX de la mano de Lofti A. Zadeh [Lofti A., 1965] como solución para la representación de procesos de inferencia conociendo a priori los grados de verdad de las premisas ²⁵. A diferencia de los conjuntos tradicionales (*crisp* en la terminología de la Lógica Difusa), los conjuntos difusos expresan el grado de pertenencia de un elemento a la categoría representada por el conjunto.

Variables lingüísticas El primer concepto importante a entender en la teoría de conjuntos difusos es el de **variable lingüística**, las cuales sirven de base para toda la teoría porque sus posibles valores son, precisamente, conjuntos difusos. Se define como:

“[...] variable whose values are words or sentences in a natural or artificial language [...]” [Zadeh, 1975]

El uso de variables lingüísticas permite representar fenómenos del mundo real como variables cualitativas. Por ejemplo, la variable *velocidad*, puede tomar los valores *atrás rápido*, *atrás lento*, *parado*, *adelante lento* y *adelante rápido*. Es cierto que la variable velocidad a su vez está definida sobre \mathbb{R} , y que los conjuntos tienen una definición sobre este dominio, pero la imprecisión que nos dan los términos (conjuntos difusos) se hereda a lo largo de todo el proceso de deducción, de manera similar a como lo hacen los humanos.

Conjuntos difusos Una posible definición de conjunto difuso podría ser *dada 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 $\mu_F(x) \in [0, 1] \forall x \in X$* . A la función $\mu_F(x)$ se la denomina **función de pertenencia**, y caracteriza únicamente a un conjunto difuso del dominio de X

²⁶ ²⁷

La teoría de conjuntos clásica también define los conjuntos de acuerdo a una función, en este caso denominada *función característica*. Las ecuaciones 14a y 14b muestran las diferencias entre los valores de f siendo ésta una función característica de la teoría de conjuntos clásica o una función de pertenencia de la teoría de conjuntos difusos respectivamente.

$$f(x) = \begin{cases} 0 & \text{if } x \notin F \\ 1 & \text{si } x \in F \end{cases} \quad (14a)$$

$$f(x) = \mu_F(x) \quad (14b)$$

Mientras que un conjunto tradicional se basa en si pertenece o no, un conjunto difuso puede ²⁸ tomar todos los valores posibles en el intervalo $[0, 1]$.

²⁵ La Lógica Difusa es uno de esos casos curiosos en la ciencia donde la aplicación nace antes que la propia teoría.

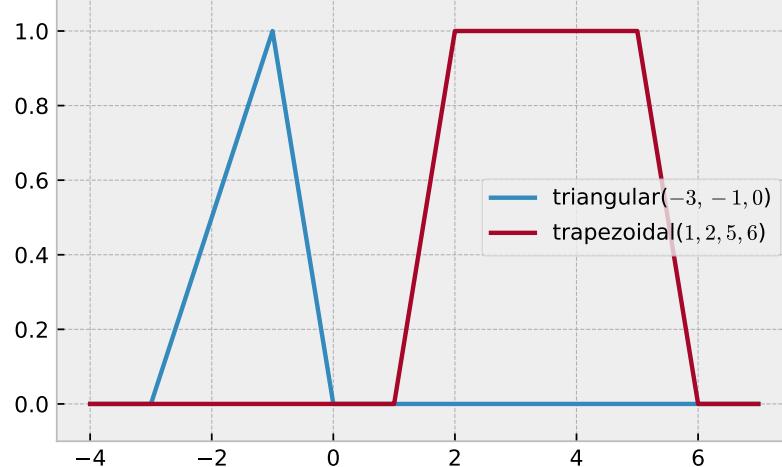
²⁶ Según Zadeh, la teoría de conjuntos difusos debería servir de ejemplo de lo que él denominó *principio de extensión* [Zadeh, 1975], esto es, el proceso por el cual generalizar cualquier teoría definida en un dominio discreto hacia su versión continua.

²⁷ Un ejemplo clásico que pone de relieve la utilidad es el de una variación de la paradoja del montón (o paradoja sotiles, donde se aplica el método inductivo para demostrar que un montón de arena es y no es un montón de arena) por parte de [Klir and Yuan-Yu, 1997]. En el artículo se argumenta que si tenemos un montón de arena y vamos quitando los granos uno por uno, llegará un momento que no tendremos dicho montón. Está claro que un grano, dos, tres no son un montón de arena, pero un montón menos uno, dos o tres granos tampoco deja de serlo. Desde el punto de vista de la lógica clásica, es imposible determinar el límite de granos donde un montón de arena deja de serlo. Este ejemplo representa una de las muchas situaciones donde es inevitable la incertidumbre.

²⁸ Puede porque también se puede definir un conjunto *crisp* desde el punto de vista de la lógica difusa.

Funciones de pertenencia Esta función de pertenencia μ_F puede tomar cualquier forma siempre que estén definidas en el dominio $[0, 1] \subset \mathbb{R}$ (independientemente de si son discretas o no), pero lo habitual es definirlas como funciones triangulares o trapezoidales. Un ejemplo de estas funciones se ilustra en la figura ??.

Figura 17: Las funciones de pertenencia triangular y trapezoidal son las dos funciones más usadas a la hora de definir conjuntos difusos, tanto manualmente como en técnicas de ajuste. La razón es su sencillez, ya que captan la esencia de la imprecisión a la hora de definir un término sobre un dominio.



Las funciones de pertenencia caracterizan a los conjuntos difusos. Su notación habitual es la que se presenta en la ecuación ??, donde F es el conjunto definido, μ su función de pertenencia y x el valor real del dominio sobre el que se define la variable lingüística.

$$F = (x, \mu_F(x)) | x \in U \quad (15)$$

Dos casos particulares de funciones de pertenencia son las funciones cuadradas (que son equivalentes a conjuntos *crisp*) y las funciones singleton (que el valor de pertenencia lo tiene un único valor en todo el dominio).

Operaciones entre conjuntos Las mismas nociones de operaciones en teoría de conjuntos son aplicables a la teoría de conjuntos difusos. Una operación entre conjuntos difusos es aquella que genera un nuevo conjunto difuso a partir de uno o más conjuntos difusos de entrada definidos sobre el mismo universo de discurso. Son las operaciones t -norma, t -conorma y complemento.

²⁹ Las propiedades son:

- Comutativa: $T(x, y) = T(y, x)$.
- Asociativa: $T(x, T(y, z)) = T(T(x, y), z)$.
- Monótona: $x \leq z, y \leq t \rightarrow T(x, y) \leq T(z, t)$.
- Identidad: $\exists 1 | T(x, 1) = x$.

La t -norma es la generalización del operador conjunción en lógica clásica. Son un conjunto de aplicaciones de la forma $T : [0, 1] \times [0, 1] \rightarrow [0, 1]$ tales que cumplen las propiedades *comutativa*, *asociativa*, *monótona* e *identidad*²⁹. Dos de las operaciones más usadas como t -norma se ilustran en la Figura 18. Existen muchas más, pero en controladores difusos se suelen usar el mínimo o, en su defecto, el producto.

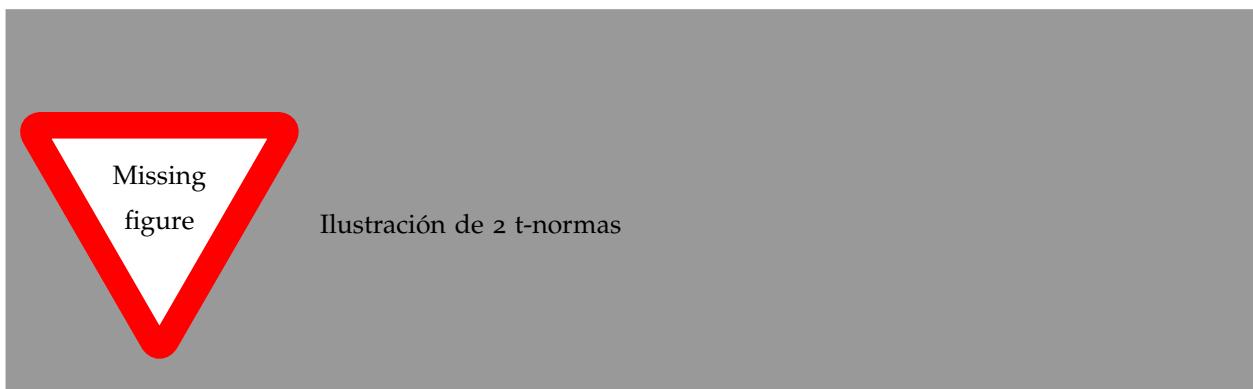
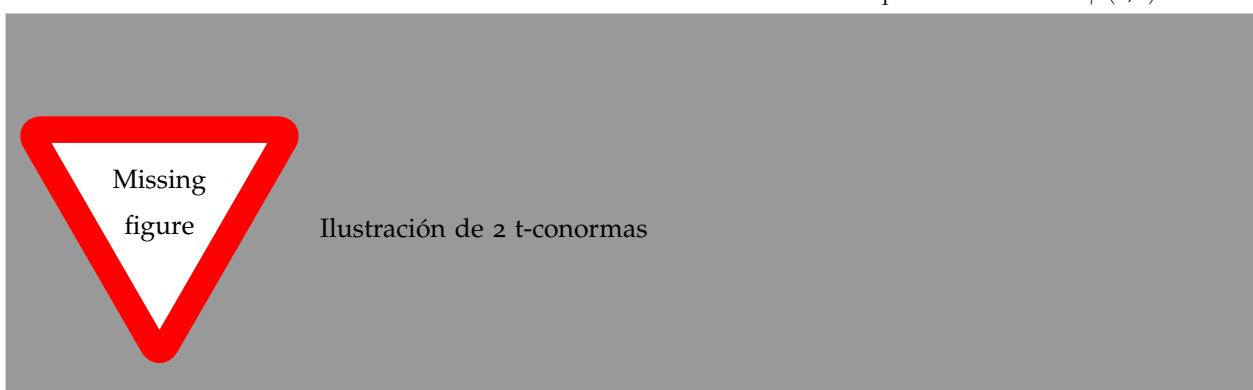


Figura 18: Las t -normas de mínimo y de producto algebraico son dos representantes de un conjunto muy amplio de funciones.

La t -conorma (o s -norma), por otro lado, generaliza el concepto de unión. Una t -conorma s se define a partir de una t -norma t como $S = 1 - T(1 - x, 1 - y)$, es decir, la función complementaria a la t -norma. De esta forma se consigue que las Leyes de Morgan se cumplan de una forma generalista. Dos ejemplos de t -conormas se ilustran en la Figura 19

Las propiedades que ha de cumplir una t -conorma son las mismas que las de la t -norma ³⁰.

³⁰ De hecho, son deducibles a partir de la t -norma con una salvedad, la identidad, que se define como $\exists 0 | S(x, 0) = x$.



El complemento de un conjunto vendrá calculado a partir de la función de pertenencia de dicho conjunto. Esto es similar a la teoría de conjuntos clásica donde el complemento de un conjunto contiene todos aquellos elementos que no estaban originariamente en dicho conjunto.

Formalmente, si F es un conjunto definido por la función de pertenencia μ_F , la función de pertenencia que definirá al conjunto complementario F^C será $\mu_{F^C} = \mu_F(x) \circ f$, siendo f una aplicación de la forma $f : [0, 1] \rightarrow [0, 1]$ tal que cumple las propiedades de **equivalencia con teoría de conjuntos clásica, estrictamente decreciente e involucion** ³¹.

El operador más común para el complemento es el probabilístico,

Figura 19: Las t -conormas de máximo y de suma algebraica son las complementarias a t -normas mostradas previamente, y como ocurre en estas, existen tantas t -conormas como t -normas definidas.

³¹ Las propiedades son:

- Equivalencia con teoría de conjuntos clásica en el caso de usar conjuntos *crisp*, es decir, $f(1) = 0 \wedge f(0) = 1$.
- Estrictamente decreciente: $\forall x, y \in [0, 1], x > y \implies f(x) < f(y)$.
- Involución: $\forall x \in [0, 1], f(f(x)) = x$.

³² El **complemento de Yager** es toda una familia de complementos que obedece a la fórmula $f_\lambda(x) = (1 - x^\lambda)^{1/\lambda}$ para $\lambda = 0$. El **complemento de Sugeno** es otra familia cuya función es diferente, $f_\lambda(x) = \frac{1-x}{1-\lambda x}$. A partir de ambas se puede definir el complemento de Zadeh variando el valor de λ .

denominado también *complemento de Zadeh en Lógica Difusa*. Existen sin embargo otros complementos, como el complemento de Yager o el complemento de Sugeno, ambos deducidos de una familia de funciones denominadas negaciones fuertes ³².

Razonamiento

El razonamiento o inferencia es el proceso de obtener consecuencias a partir de hechos (premisas). En *Lógica Difusa*, el proceso de inferencia lógica se amplía a través de relaciones difusas, trasladando valores de verdad aproximados.

Toda regla difusa está compuesta por un antecedente y un consecuente. Tanto el uno como el otro están compuestos de sentencias que asignan valores de variables difusas (conjuntos difusos) a éstas. La forma más común de representar las reglas difusas es a través de reglas IF ... THEN, donde las relaciones entre elementos (i.e. AND, OR y NOT) se corresponden con las operaciones de *t-norma*, *t-conorma* y complemento. La Figura ?? detalla los componentes de estas reglas.

La implicación lógica $A \implies B$ es equivalente a $\neg A \vee B$, pero en lógica difusa esta equivalencia arroja valores contra-intuitivos. Una definición ampliamente extendida es la del uso de una *t-norma*, como es el caso de la **Implicación de Mamdani**, pero existen muchos tipos diferentes ³³.

Modus Ponens Generalizado Se trata de la extensión del método del **modus ponens** para obtener el valor de verdad de un consecuente a partir de un antecedente ambos difusos. En la Figura 20

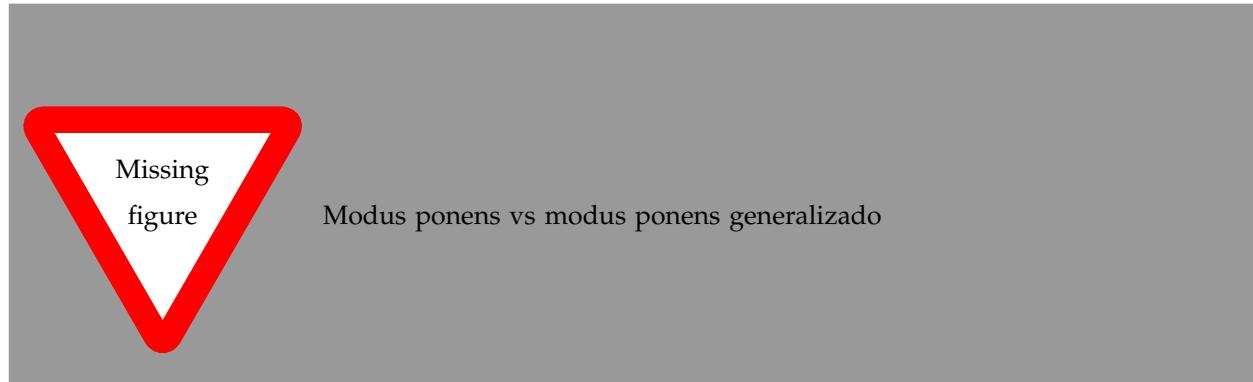


Figura 20: Diferencias entre *modus ponens* clásico y *modus ponens generalizado*. Debido a que los valores de las reglas de inferencia difusa son conjuntos difusos, la conclusión de un silogismo es por tanto un conjunto difuso, no un valor absoluto de verdad o mentira.

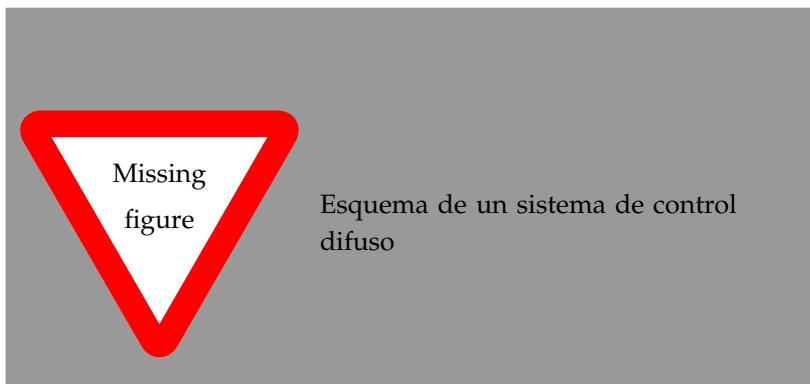
La *regla composicional de inferencia* es el método para determinar qué grado de asignamos a un consecuente a partir de las premisas de los antecedentes en un proceso de *modus ponens generalizado*, o dicho de otro modo, cómo obtener la función de pertenencia del conjunto resultado a partir de las funciones de pertenencia de las premisas.

Sin entrar demasiado en detalle, cada regla de un sistema difuso de la forma $IF\ x\ is\ A\ THEN\ y\ is\ B$ se puede representar como una función I como $I(\mu_A(x), \mu_B(y))$ [La regla composicional de Zadeh]. REVISAR ESTO PORQUE CREO QUE EN LA TESIS DE MASTER ESTÁ MAL O RARO.

Sistemas de control difuso

Los () 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 entradas³⁴. Su uso se aplica tanto a sistemas de control manual como a sistemas donde los elementos a controlar son muy difíciles o incluso imposibles de diseñar (e.g. procesos de control con un alto número de variables relacionadas entre si).

Éste se puede definir como un componente que asigna salidas de dominios no difusos (*crisp*) a entradas de dominios también no difusos (*crisp*) de forma no lineal haciendo uso de lógica difusa. La arquitectura de estos sistemas está compuesta por cuatro bloques: *fuzzificador*, *bloque de reglas*, *razonador* y *defuzzificador*. En la Figura 21 se describe el esquema de un sistema de control difuso general.



³⁴ Dado que el control difuso puede extenderse a casi cualquier problema de control tradicional hace que se haya usado en multitud de campos, no sólo en el industrial. Esto hace que el mismo concepto pueda encontrarse con muchos nombres diferentes, como Sistemas de Lógica Difusa, Sistemas de Inferencia Difusa, Sistemas Expertos Difusos, Sistemas Basados en Reglas Difusas o simplemente Sistemas Difusos. [COGER LAS REFERENCIAS DE LA TESIS DE MASTER]

Figura 21: Un sistema de control difuso es un componente compuesto por cuatro bloques principales: fuzzificador, bloque de reglas, razonador y defuzzificador.

Fuzzificador Este bloque recibe las entradas *crisp* al sistema (los cuales pertenecerán al dominio sobre el que las variables están definidas) y las transforma a valores de pertenencia de sus respectivos conjuntos difusos.

Bloque de reglas Define el conjunto de reglas que gobiernan el proceso deductivo del sistema de control difuso. A estas reglas se les puede aplicar pesos en caso de que se quiera simbolizar la mayor importancia de una regla frente al resto. Se suele representar como un parámetro $w_i > 0 \in \mathbb{R}$ para la regla i -ésima, el cual se multiplica al resultado de la regla para ponderar la importancia de la misma.

La forma más común de representación es a través de reglas de la forma IF ... THEN ... por su facilidad de comprensión.

Sin embargo, otra representación muy usada (sobre todo como representación interna) es la de m matriz n -dimensionales (una por cada conjunto difuso de salida), donde cada dimensión se corresponde con una variable lingüística, con los conjuntos difusos de éstas como índices y con los valores que toman esas intersecciones como los valores de esa salida. En la Figura 22³⁵.

³⁵ En el desarrollo de la tesis veremos un caso en el que esta representación es útil a la hora de inferir un controlador difuso a partir de un conjunto de entradas. Figura 22: Un ejemplo de representación matricial de un conjunto de reglas difusas. Las reglas activas se simbolizan como un 1 (o como un valor real en caso de querer aplicar pesos).



Razonador Se encarga de realizar todo el procesamiento de inferencia haciendo uso de las entradas fuzzy (las salidas del bloque fuzzificador) y las reglas del bloque de reglas.

A partir de éstos realizará un proceso de deducción (normalmente *modus ponens generalizado*) tras el cual se obtendrán las funciones de pertenencia para cada uno de los conjuntos difusos resultado de las conclusiones del proceso de inferencia.

Defuzzificador Realiza el proceso inverso del bloque fuzzificador. Dado que a la salida del razonador sólo contamos con valores de pertenencia al conjunto difuso de salida, es necesario una transformación al dominio de la variable lingüística de salida para convertirlo en un valor interpretable por el sistema a controlar.

$$f(x) = \frac{\int_F x \cdot \mu_F(x) \cdot dx}{\mu_F(x)} \quad (16a)$$

$$f(x) = \frac{\sum_F x \cdot \mu_F(x) \cdot dx}{\sum_F \mu_F(x)} \quad (16b)$$

Existen diferentes algoritmos para la transformación de un conjunto difuso (expresado a partir de su función de pertenencia μ_F) a un valor *crisp*. Suelen ser comunes en conjuntos de salida continuos el COA (*centroid of area*, ecuación 16a) y en conjuntos de tipo singleton su versión *media ponderada* (ecuación 16b)³⁶.

³⁶ En el trabajo [Defuzzification: criteria and classification] se realiza una clasificación muy completa de una gran cantidad de algoritmos de defuzzificación.

Tipos de Sistemas de Control Difuso

Los sistemas de control difuso se han usado en muchos dominios diferentes. Son tantos los autores que han trabajado sobre ello que los sistemas tienden a variar en su funcionamiento. Existen dos tipos de controladores que son los que dominan la literatura, los cuales de describen brevemente a continuación.

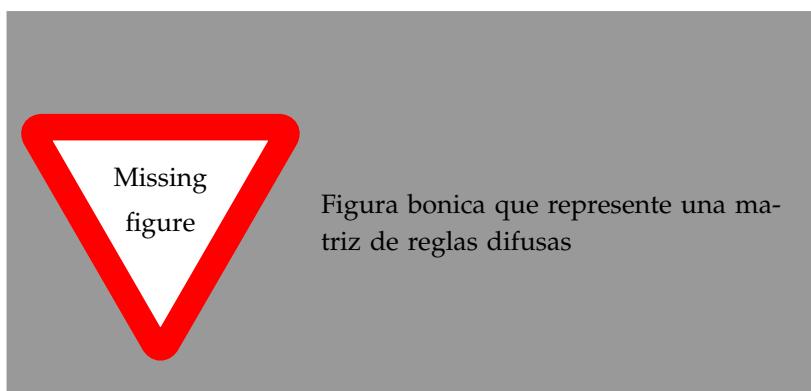
Controlador difuso tipo Mamdani Este controlador fue el primer intento de control destinado automatizar un motor de vapor y su caldera utilizando para ello un conjunto de reglas difusas del tipo *IF ... THEN ...* obtenidas directamente de la experiencia de los operarios.

Su estructura es la básica presentada previamente (ver Figura 21).

Es consecuente de un sistema de tipo *Mamdani* es siempre un conjunto difuso y por tanto requiere de un proceso de defuzzificación que es costoso. De ahí el significado el siguiente tipo de controlador.

Controlador difuso tipo Takagi-Sugeno-Kang Este tipo de controlador difuso fue propuesto por Takagi, Sugeno y Kang en los trabajos [Fuzzy identification of systems and its applications to modeling and control, Structure identification of fuzzy model], aunque se suele abbreviar a tipo Takagi-Sugeno o, directamente, Sugeno.

En este tipo de controlador, el bloque de defuzzificación desaparece y los consecuentes de las reglas de inferencia son reemplazados por una función (normalmente un polinomio basado en los parámetros del antecedente) la cual nos devuelve directamente un valor *crisp* (ver Figura 23).



En función del orden del polinomio de la función de salida, el controlador recibe también el mismo orden. De esta manera, por ejemplo, si tenemos dos valores de entrada x e y , cuando $f(x, y) = c$ (constante) se dice que el controlador es de orden 0. Si $f(x, y) = ax + by + c$, se dice que es de orden 1, y así sucesivamente³⁷.

Figura 23: En un controlador de tipo Takagi-Sugeno-Kang se prescinde del bloque de defuzzificación y los consecuentes de las reglas de inferencia son reemplazados por funciones. Por término general, estas funciones son polinomios.

³⁷ Los controladores de tipo Takagi-Sugeno-Kang de orden 0 son equivalentes a controladores de tipo Mamdani cuyos conjuntos difusos de salida son de tipo singleton.

El consecuente en un controlador de tipo Sugeno es un valor directamente, y por tanto no necesita proceso de defuzzificación. Sin embargo, la respuesta pierde significado semántico, a diferencia de un controlador de tipo Mamdani.

El paradigma de los Agentes Inteligentes

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 entidades (e.g. sistemas, software, ...) actúen de la manera más inteligente posible. El concepto de **agente** pertenece a todo el área de la computación, y no únicamente a la **AI**.

Es difícil encontrar un consenso en la definición de agente, y más todavía cuando entra en juego el adjetivo *inteligente*. Sin embargo, sí existen una serie de características que coínciden dentro de la literatura que exponemos a continuación (ver Figura 24):

- 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.

Nosotros hablaremos de estas entidades desde el punto de vista de la **AI**, y las denominaremos indistintamente *agentes*, *agentes inteligentes* o *agentes racionales*³⁸. Por tanto usaremos la siguiente definición:

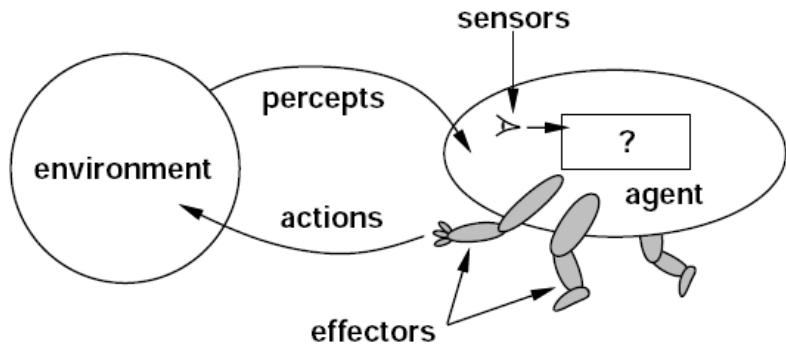
“Un agente es una entidad física o virtual que realiza una acción ³⁹ de manera total o parcialmente autónoma dada una secuencia de percepciones del entorno en el que se ubica.”

Según ésta, 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 a cumplir y comprensión del entorno en el que se desarrolla la acción, que puede ser o no correcta.

³⁸ El término preferido en esta tesis es precisamente este último, **agentes racionales**, dado que capture la esencia de lo que es un comportamiento inteligente. El problema de este punto de vista es que, como bromea en [Russell et al., 2003], hasta un elemento tan rudimentario como un termostato puede ser considerado un elemento inteligente, ya que realiza siempre la mejor acción para cumplir sus objetivos por muy simples que puedan parecer. 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 (e.g. en [Russell et al., 2003]). Según otros, es necesario que el comportamiento sea flexible, esto es, que sea reactivo (reacciona ante el entorno que percibe), proactivo (iniciativa para tratar de cumplir sus objetivos) y social (capaz de interactuar con otros agentes para cumplir sus objetivos) (e.g. [Wooldridge et al., 1995]). Y otros directamente exigen, además, un comportamiento racional a la hora de cumplir los objetivos para calificarlo de inteligente (e.g. [Shoham, 1993]). Como dijimos anteriormente, dónde está el límite entre qué es y qué no es un inteligente cae dentro de los dominios de la filosofía.

³⁹ 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 impri me 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 24: 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 identifican. Es autónomo, opera realiza 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. **TODO!**CAMBIAR LA IMAGEN Y VER CÓMO SE PUEDE INCLUIR EL CONCEPTO DE COLABORACIÓN



Las nociones de agentes inteligentes y la de **IC** van de la mano. Esto es debido a que su definición funciona a la perfección para las técnicas de la **IC**, 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**, **Lógica Difusa**, ...). Por ello desde mediados de los años 1990 el concepto de agente inteligente ha ganado tanta popularidad⁴⁰.

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 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**⁴¹.
- **Episódico o. secuencial.** Un entorno en el que las acciones se dividen atómicamente donde cada una de ellas conlleva un ciclo

⁴⁰ 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.

⁴¹ En general, los entornos del mundo real tienden a ser tan complejos que es imposible para un agente abarcar todos los aspectos medibles de éste. Por lo tanto, sea o no la naturaleza del entorno determinista, en general se suele suponer éste como no determinista.

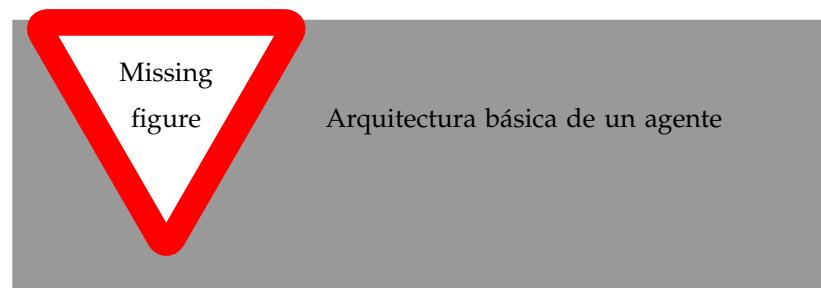
de (percepción, decisión, acción) y sin relación una con otra se denomina episódico. Si en lugar de ello la acción del agente puede afectar a las decisiones futuras se dice que el entorno es **no episódico o secuencial**.

- **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**⁴².

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.

Figura 25: 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.



La figura 25 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

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 26):



Figura 26: 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.

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 27):

- **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.

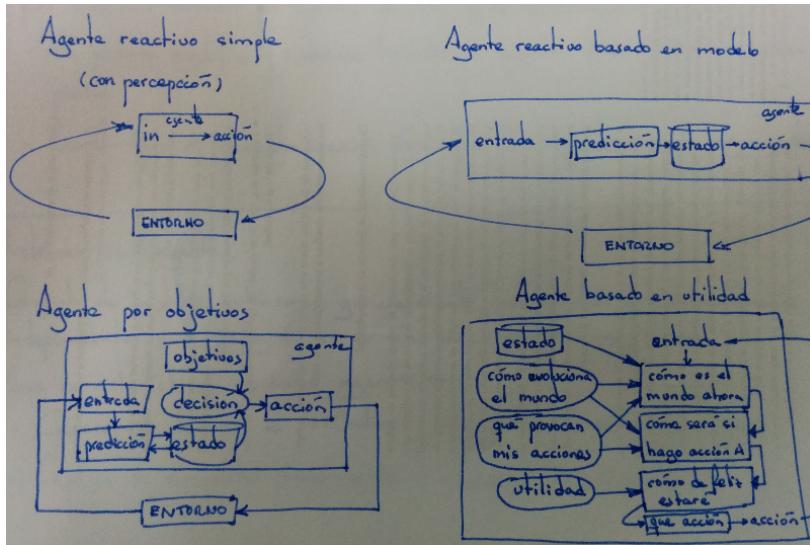


Figura 27: Distintas arquitecturas de agentes en función del comportamiento. Dependiendo de las acciones a realizar, se identifican tres tipos, los reactivos que aplican una acción sin proceso deductivo y los basados en modelo y utilidad (en algunos contextos denominados deliberativos) que basan su comportamiento en alguna forma de deducción.

- **Basados en objetivos.** Plantean una deducción de forma que determinan cuál sería el estado del entorno tras 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 25, 26 y 27).

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](#), 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 [Sistema Multiagente](#), 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 quizás 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 **Sistema Multiagente** 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 **Sistema Multiagente**, ya que para denominarse de esta manera dos o más agentes deben interactuar (i.e. comunicarse) entre sí. Esta interacción puede implementarse de diversas maneras⁴³ y siempre toman una o las dos formas siguientes (figura 28):

- **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 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.

⁴³ 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.

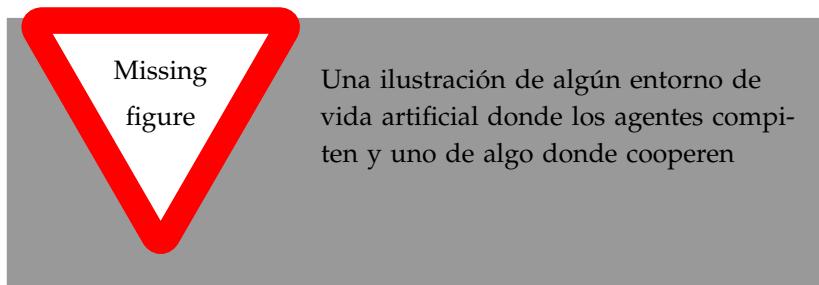


Figura 28: La comunicación entre agentes puede ser de dos tipos: *colaborativa*, donde los agentes tratan de llegar a una solución intercambiándose información y *competitiva*, donde los agentes compiten unos contra otros en un entorno.

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 **Driver-Vehicle Units (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 acotada 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 41).

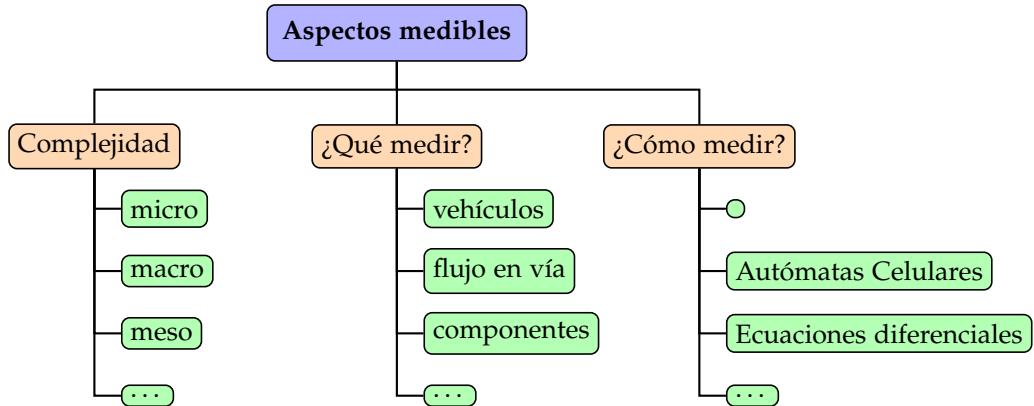


Figura 29: Los aspectos medibles del problema del tráfico son muy diversos, y dependen del nivel de granularidad (i.e. complejidad) al que se quiere llegar, de qué queremos medir y de cómo lo queremos hacer.

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 30:

- **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 inviables⁴⁴.
- **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 31.

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**.

⁴⁴ 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.) 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 30: 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.

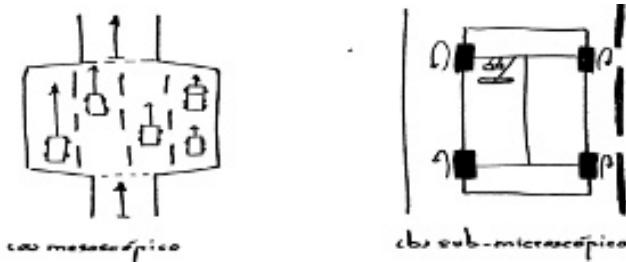
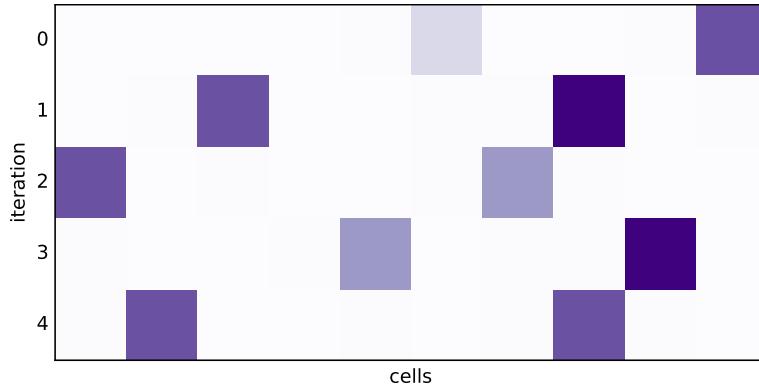


Figura 31: Otras aproximaciones alternativas de modelos en función de la complejidad. Ejemplo de mesosimulación como ventana de microsimulación dentro de un flujo de tráfico en un macrosimulador (e.g. [Munoz et al., 2001]) y ejemplo de submicrosimulación donde se modelan componentes internos de un vehículo.

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 32: 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 un modelo de *car-following* 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-schreckenberg-demo (Ver capítulo Sistemas desarrollados).

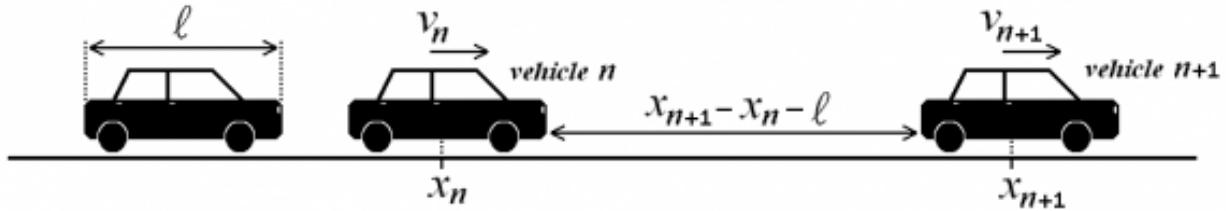


En el caso de la dimensión **espacio**, la clasificación diferencia las simulaciones que se mueven por un espacio discreto o por uno continuo:

- **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 32).
- **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 33).

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 **CAs** son también simuladores típicos discretos, ya que cada posición en el espacio se va calculando para cada intervalo discreto de tiempo (figuras 32 y 34).

- **Tiempo continuo.** En estos simuladores el tiempo es un factor más para un modelo de ecuaciones diferenciales. La figura 33 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 .

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 síncrona ⁴⁵ (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 33: 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].

⁴⁵ 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.

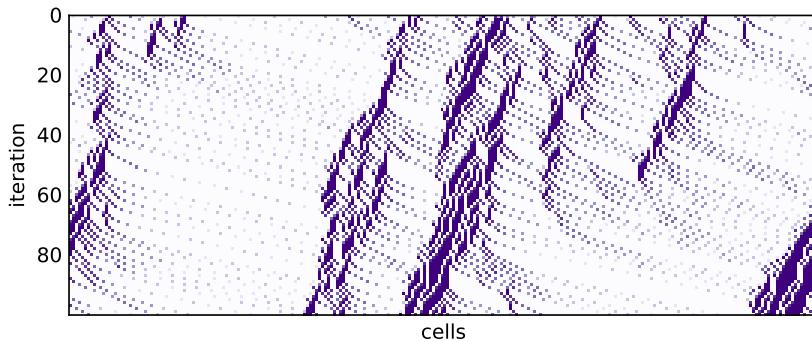


Figura 34: Aparición de retenciones en una autopista de 250 celdas usando el modelo Nagel-Scherckenberg. La densidad de ocupación es de 50 coches en la vía, la velocidad máxima es de $5c/\Delta t$ y la probabilidad de frenada es de $p = 0,5$. Se puede observar en la figura cómo se desplazan las olas 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 34 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 cada uno de los agentes tiene su propia en-

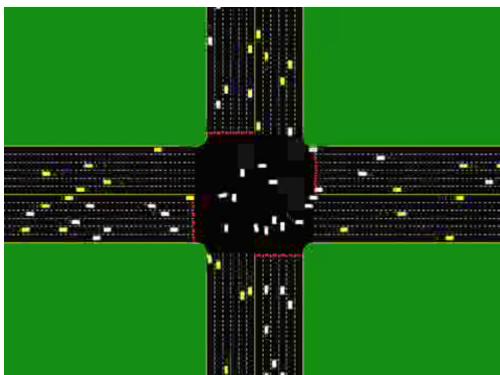


Figura 35: Simulación de comportamiento en intersección basada en un . En ésta, cada uno de los vehículos representa a un vehículo real que posee un controlador para hacerlo autónomo. Modelar este caso de estudio con una arquitectura basada en permite centrarse en el diseño del agente en concreto (i.e. 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/>).

tidad 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 **Sistemas Multiagente** 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.

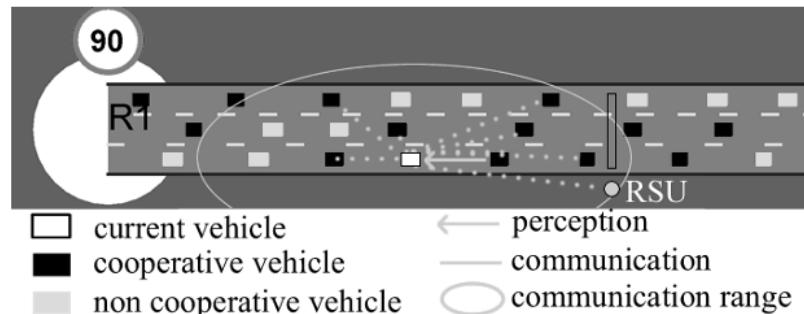
A diferencia de los **CAs**, los **Sistemas Multiagente** pueden emplazarse en un entorno virtual que represente un espacio continuo y no discreto. Esto permite modelar con mayor fidelidad magnitudes físicas asociadas a cada agente (e.g. posición y velocidades actuales, dimensiones del vehículo, masa, 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 .

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 35. 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 **Sistema Multiagente** 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** ≡ 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 36: 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 carros. Fuente: [Gu et al., 2015].



Los estudios en materia de simuladores de tráfico 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 intravehícuulares, **Vehicle-to-Vehicle (V2V)** y **Vehicle-to-Infrastructure (V2I)** respectivamente. El modelo de **Sistema Multiagente** permite la implementación rápida de diferentes políticas y protocolos de comunicación via sensores y actuadores para estudiar estos tipos de redes de comunicación (figura 36). Estudios como por ejemplo [Shiose et al., 2001] o [Galis and Rao, 2000] hacen uso de un **Sistema Multiagente** 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.

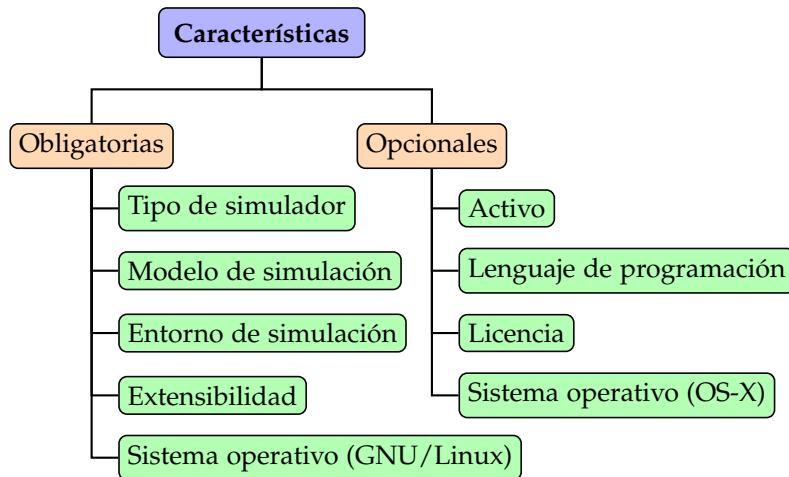


Figura 37: Características obligatorias y deseables del simulador donde implementar nuestros modelos personalizados de conductor.

Software de simulación

Para la realización de esta tesis es necesario contar con un paquete de simulación que permita modelar un 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 37):

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 **Sistema Multiagente** 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-

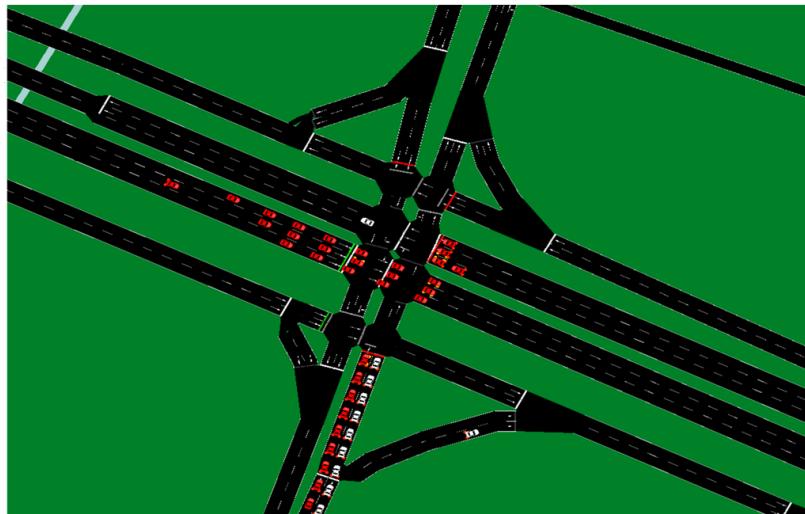


Figura 38: Captura de pantalla del simulador **SUMO**. Además de software de simulación propiamente dicho, **SUMO** provee de una interfaz gráfica que permite una visualización general, de zonas y de elementos en concreto a la vez que permite la variación de configuración de la simulación durante el 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 38) 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).

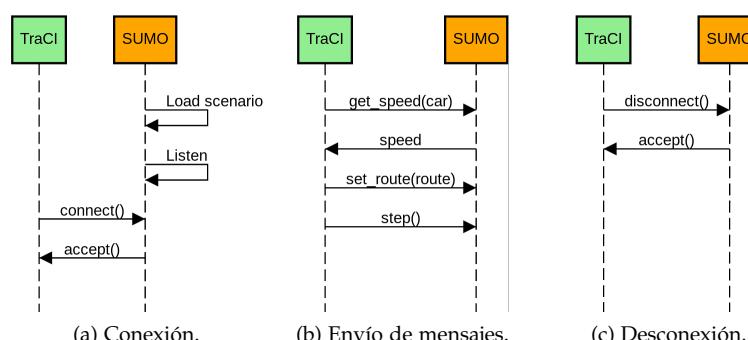
SUMO usa como modelo por defecto de *car-following* el modelo de Stefan Krauß [Jin et al., 2016], debido a su simplicidad y su velocidad de ejecución y como modelo de cambio de carril el modelo de Gipps [Krajzewicz et al., 2002]. No obstante, se encuentran paraseleccionar otros modelos como el **Intelligent 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.

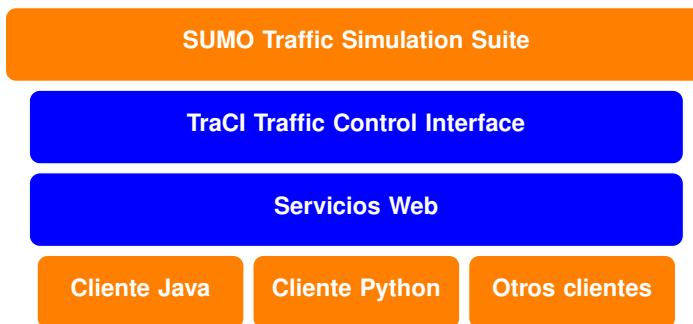
La interfaz **TraCI**

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 39: **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 39 (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 39 (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 39 (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**⁴⁶. El homólogo de la librería de abstracción de Python pero para el lenguaje Java. Está desarrollada por un tercero.
- **TrasS**⁴⁷. Una plataforma ofrecida como SaaS que proporciona una interfaz de servicios web bajo protocolo SOAP para abstraer el protocolo en mensajes HTTP (figura 40).

Figura 40: 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.

⁴⁶ <https://github.com/egueli/TraCI4J>.

⁴⁷ <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 , 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 41).

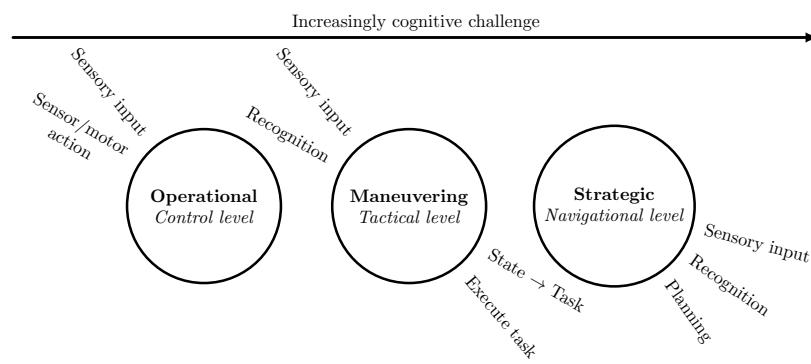


Figura 41: Los tres niveles jerárquicos que describen la tarea de conducción según [Michon, 1985]: *estrategia* (i.e. las decisiones generales), la *maniobra* (i.e. decisiones durante la conducción de más corto plazo) y *control* (i.e. automatismos).

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.

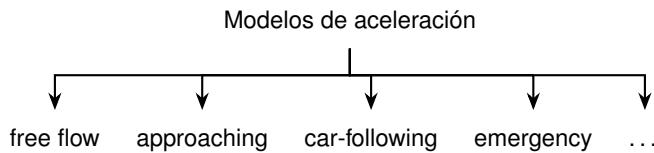
El comportamiento de un conductor al volante tiene una relación directa con el nivel de abstracción táctico. Se puede entender como el nivel encargado de planificar acciones a corto plazo para conseguir objetivos a corto plazo. Las tareas de control son automáticas e influyen poco o nada en la toma de 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 concretas⁴⁸.

⁴⁸ 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 . 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 , 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 máxima). Dependiendo del trabajo, algunos autores añaden más conocimiento.



miento 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 .

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 43 y 42): 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 de tráfico es de ligera, mejora la velocidad media del flujo de la

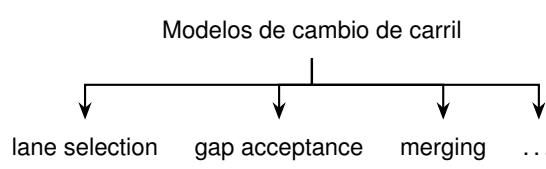
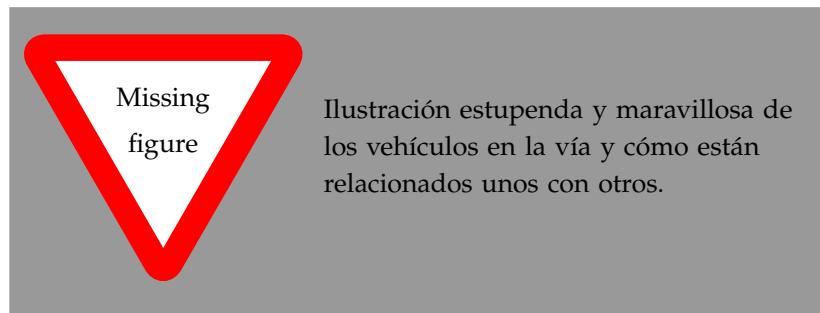


Figura 42: Tras la aparición de los modelos psico-físicos se comprobó que los umbrales en las percepciones y por tanto el comportamiento podía variar dependiendo de las situaciones. Por ello, el *car-following* no era más que una entre diferentes clases o regímenes de aceleración. Algunos de los regímenes 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 43: 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 44: Representación de los vehículos en una vía junto con la nomenclatura a usar durante el resto de la tesis.



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 44 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:

$$a(t) = cv^m(t) \frac{\Delta v(t - \tau)}{\Delta x^l(t - \tau)} \quad (17)$$

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 17), 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 45: 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].

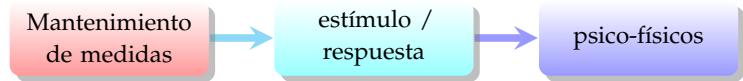
Los métodos de estas dos clases tienen un problema principal: suponen que el conductor es capaz de percibir incluso el más ínfimo cambio en las variables observadas, cuando la realidad no es así. Por ello, a mediados de los años 70 apareció una nueva clase de modelos de *car-following*, denominados posteriormente 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 lentos⁴⁹. 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 18 se describe el modelo típico de un modelo de *gap acceptance*.



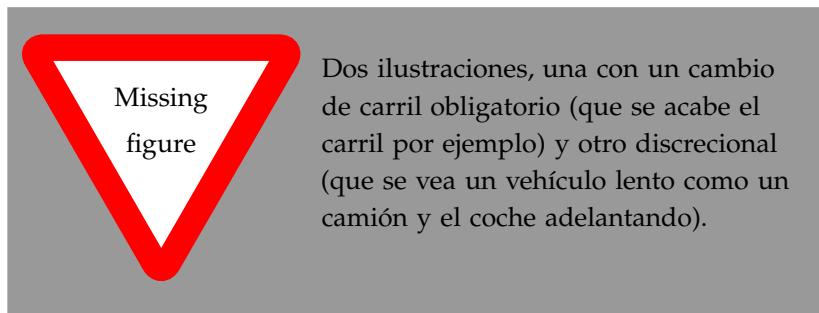
⁴⁹ En [Sparmann, 1978] el autor entiende de la derecha como el carril lento y la izquierda como el carril rápido, y es como se entiende dentro del contexto de esta tesis. Sin embargo en otros países esta correspondencia es al revés.

El modelo típico del **gap acceptance** responde a la ecuación 18, 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) $g_l^{crit}(t)$.

$$f_{gl}(t) = \begin{cases} 0 & \text{si } g_l(t) < g_l^{crit}(t) \\ 1 & \text{si } g_l(t) \geq g_l^{crit}(t) \end{cases} \quad (18)$$

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 46: 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 42).

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 46).

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

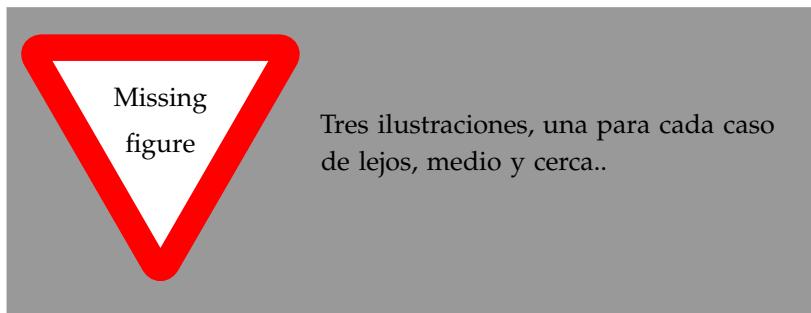


Figura 47: En el modelo de [Gipps, 1986], la distancia al punto (i.e. cerca, media distancia y lejos) determina el grado de obligatoriedad del cambio.

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]).

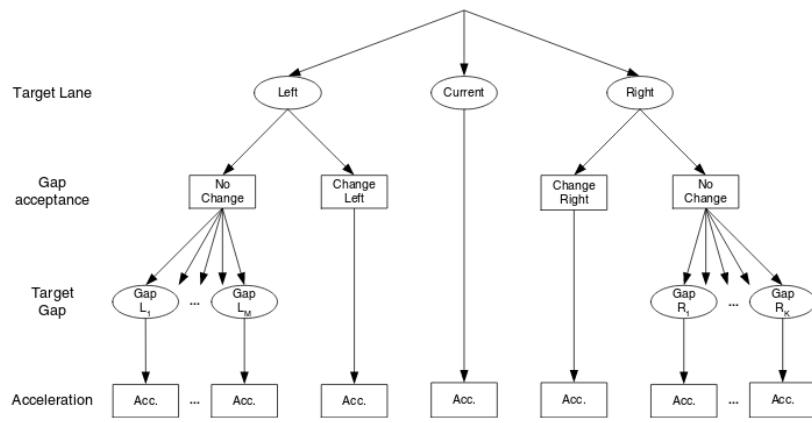


Figura 48: Estructura del modelo de comportamiento de los vehículos propuesto por [Toledo et al., 2007]. Este modelo se basa en el concepto de “objetivo a corto plazo” para elaborar un “plan a corto plazo” apoyándose, para ello, en un árbol de decisión. Aunque mantiene la clasificación, es probabilístico 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 48).

Modelado basado en

Desde mediados de los años 90 empezó a crecer el interés por aplicar la a los **ITS**. Hay dos razones por las que esto es así: la primera, los éxitos cosechados por la , los cuales atrajeron a investigadores de multitud de áreas incluida ésta. La segunda, el rápido desarrollo de la tecnología ⁵⁰, 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 **IC**, 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 **IC** 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 49). Por ello, aunque nuestro interés pueda centrarse en el modelado de conductores, es necesario conocer el estado de las demás áreas.

- En el **reconocimiento de patrones** las técnicas suelen trabajar en

⁵⁰ 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 49: Las principales áreas de aplicación de la **IC** 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.



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).

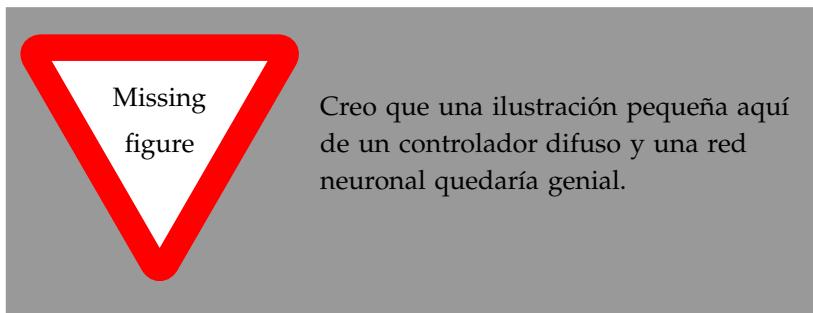
- La **caracterización de conductores** es interesante debido a que permite la identificación de perfiles de conducción y su clasificación de acuerdo a indicadores extraídos de su manera de conducir.
- El **modelado de conductores** nos permite la reproducción de comportamientos en simulación.

Las técnicas de **IC** más utilizadas en el modelado de conductores son las Redes Neuronales Artificiales y la **ANN**. 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 **IC** 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] o la predicción ([Dougherty et al., 1993, Chan et al., 2012] entre muchos otros.



Creo que una ilustración pequeña aquí de un controlador difuso y una red neuronal quedaría genial.

Figura 50: Al trabajar con métodos como las Redes Neuronales Artificiales o la , la inexactitud y la incertidumbre son ciudadanos de primera clase y forman parte de los modelos.

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 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 17) 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 [?, 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 ⁵¹.

LAS REDES NEURONALES ARTIFICIALES Y LA 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 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

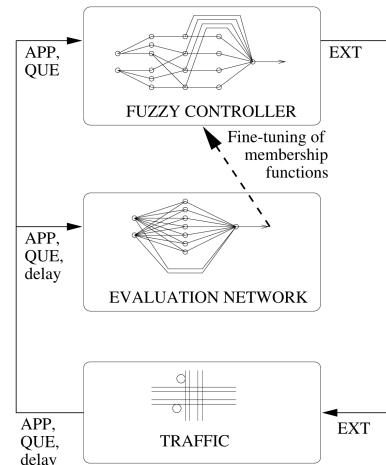


Figura 51: Un ejemplo de aproximación *neuro-fuzzy*, en este caso aplicado al control de señales de tráfico. El se implementa como una Red Neuronal Artificial de tipo *feed-forward* en lugar de con una representación tradicional. Además, el sistema completo lleva integrado un subsistema basado en también en Redes Neuronales Artificiales *feed-forward* que ajusta las funciones de pertenencia a través de entrenamiento por refuerzo. Fuente: [Bingham, 2001]

⁵¹ 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 **IC**.

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 LLocal 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 52). Éste ha sido instrumentado con los siguientes dispositivos:



Figura 52: 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 53.

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 53: 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].

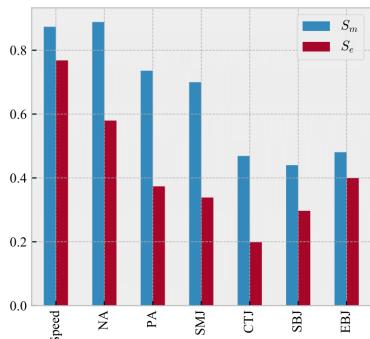
Los valores brutos se representan en la Tabla 3. Sus perfiles se han extraído de los datos de GPS y CAN Bus como se describe en [1]. La figura muestra el perfil normalizado para los datos de los dos subconjuntos de controladores (llamados Sm y Se).

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)

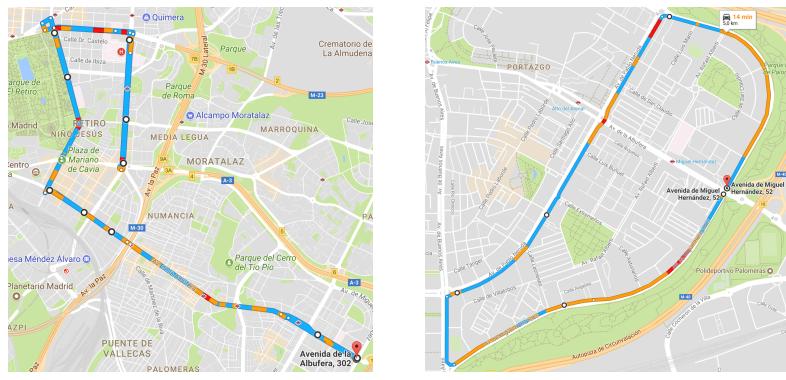


(b)

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 55 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 55: 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 54: 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 56 shows a simplified schematic outlining this concept.

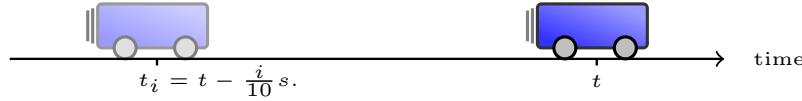


Figura 56: Given the row of data in time t , we define the moment T_i as the row of data that occurred in time $t - \frac{i}{10}$ seconds.

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_{it} and ds_{iv} , 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 45CNNs 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

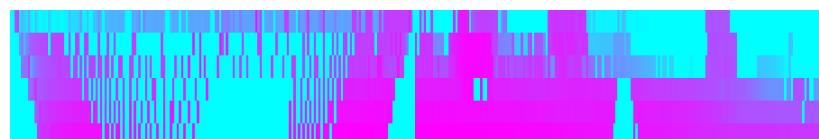
and, for each of them, we go through each point and apply it a new random shift of . After this process, a new sequence is created with the same lane-change intention and action, and similar but slightly different environment to the original one.

(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 57).

Figura 57: 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 rotondas?

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 EC

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 lo 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.

Módulos de ROS

Hablar que el paquete anterior quedó discontinuado y empezamos a desarrollar paquetes ara ROS, en el cual incluimos un desarrollo de CAN y un launcher.

Nagel-Schreckenberg

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

Glosario

ACL Agent Communications Language. [31](#)

GA Genetic Algorithm. [11](#)

CNN Convolutional Neural Network. [15](#)

Hard Computing Forma de denominar a la computación clásica en contraposición al término *Soft Computing*. [12](#)

AI Inteligencia Artificial. [5, 9–13, 26, 27, 31](#)

IC Inteligencia Computacional. [3, 9, 11–13, 15, 17, 19, 21, 23, 25, 27, 29, 31](#)

Inteligencia Artificial Distribuida Rama de la *AI* donde se estudian las técnicas de aplicación, coordinación y distribución de conocimiento en un entorno multiagente.. [31](#)

KQML Knowledge Query and Manipulation Language. [31](#)

Lógica Difusa Extensión de la lógica tradicional donde los valores no son únicamente verdaderos (1) o falsos (0) sino que pueden tomar un grado de verdad o falsedad en todo el intervalo [0, 1].. [9, 11, 22, 23, 27](#)

ML Machine Learning. [11](#)

Perceptrón Multicapa Topología de red neuronal *feed-forward* donde las redes están organizadas en capas de tal manera que todas las neuronas de una capa están conectadas con todas las siguientes..
[15](#)

NLP Natural Language Processing. [11](#)

ReLU Rectified Linear Unit. [5, 17, 18](#)

ANN Artificial Neural Network. [9, 10, 14, 18, 22, 27](#)

RS Recommender System. [11](#)

Sistema de Control Difuso Sistema para realizar tareas que basa su funcionamiento en inferencia difusa. En general es sinónimo de [Sistema de Inferencia Difusa](#), sólo que remarcando el objeto de su funcionamiento (tareas de control).. [24](#)

Sistema de Inferencia Difusa Sistema que basa su funcionamiento en la aplicación de reglas lógicas con un motor de inferencia basado en la teoría de la lógica difusa.. [5](#), [24](#), [25](#)

Sistema Experto Sistema que emula a un humano en la toma de decisiones de un dominio en el que es experto.. [11](#)

Sistema Multiagente Sistema que emplea al menos dos agentes que interactúan entre sí para solucionar el problema para el que está diseñado.. [31](#)

Soft Computing Paradigma de computación por el que se hace uso de técnicas de resolución de problemas que manejan información incompleta, inexacta o con ruido.. [12](#)

Bibliografía

- [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 Taniguchi, 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, 1949] Hebb, D. O. (1949). The organization of behavior: A neurophysiological approach.
- [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.
- [Hochreiter and Schmidhuber, 1997] Hochreiter, S. and Schmidhuber, J. (1997). Long short-term memory. *Neural computation*, 9(8):1735–1780.
- [Hopfield, 1982] Hopfield, J. J. (1982). Neural networks and physical systems with emergent collective computational abilities. *Proceedings of the national academy of sciences*, 79(8):2554–2558.
- [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.
- [Howard et al., 2017] Howard, A. G., Zhu, M., Chen, B., Kalenichenko, D., Wang, W., Weyand, T., Andreetto, M., and Adam, H. (2017). Mobilenets: Efficient convolutional neural networks for mobile vision applications. *arXiv preprint arXiv:1704.04861*.
- [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.
- [Ioffe and Szegedy, 2015] Ioffe, S. and Szegedy, C. (2015). Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *International conference on machine learning*, pages 448–456.
- [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).
- [Kingma and Ba, 2014] Kingma, D. P. and Ba, J. (2014). Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.
- [Klir and Yuan-Yu, 1997] Klir, G. J. and Yuan-Yu, H. (1997). *Fuzzy set theory: theory & applications*. na.
- [Kohonen, 1998] Kohonen, T. (1998). The self-organizing map. *Neurocomputing*, 21(1-3):1–6.

- [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(Idm):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.
- [Lofti A., 1965] Lofti A., Z. (1965). Fuzzy sets. *Journal of Information and Control*, 8(3):338–353.
- [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.
- [Ng, 2004] Ng, A. Y. (2004). Feature selection, l 1 vs. l 2 regularization, and rotational invariance. In *Proceedings of the twenty-first international conference on Machine learning*, page 78. ACM.
- [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.

- [Qian, 1999] Qian, N. (1999). On the momentum term in gradient descent learning algorithms. *Neural networks*, 12(1):145–151.
- [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.
- [Robinson et al., 2007] Robinson, A. E., Hammon, P. S., and de Sa, V. R. (2007). Explaining brightness illusions using spatial filtering and local response normalization. *Vision research*, 47(12):1631–1644.
- [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., Malik, J. M., and Edwards, 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.
- [Shoham, 1993] Shoham, Y. (1993). Agent-oriented programming. *Artificial intelligence*, 60(1):51–92.
- [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*.
- [Srivastava et al., 2014] Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., and Salakhutdinov, R. (2014). Dropout: A simple way to prevent neural networks from overfitting. *The Journal of Machine Learning Research*, 15(1):1929–1958.
- [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.
- [Widrow and Hoff, 1960] Widrow, B. and Hoff, M. E. (1960). Adaptive switching circuits. Technical report, STANFORD UNIV CA STANFORD ELECTRONICS LABS.
- [Wiedemann, 1974] Wiedemann, R. (1974). Simulation des Straßenverkehrsflusses. Technical report.
- [Wolpert and Macready, 1997] Wolpert, D. H. and Macready, W. G. (1997). No free lunch theorems for optimization. *IEEE transactions on evolutionary computation*, 1(1):67–82.
- [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., Pischedel, 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.
- [Zadeh, 1975] Zadeh, L. A. (1975). The concept of a linguistic variable and its application to approximate reasoning—i. *Information sciences*, 8(3):199–249.
- [Zeiler, 2012] Zeiler, M. D. (2012). Adadelta: an adaptive learning rate method. *arXiv preprint arXiv:1212.5701*.
- [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, 83
- back propagation, 67
- back-propagation, 91
- car-following, 71, 78, 83–87, 90–92
- courtesy yielding function, 87
- critical gap, 85
- critical-gap, 87
- emergency, 83
- fast-forward, 90
- free flow, 83
- free-flow, 86, 89
- gap acceptance, 85
- gap acceptance, 83, 85
- gap-acceptance, 87
- lane selection, 83
- lane-changing, 86
- lane-selection, 85
- merging, 83, 85
- neuro-fuzzy, 91, 92
- recurrente, 90
- umbral perceptual, 85

Cómo citar esta tesis

Si deseas citar esta tesis, lo primero gracias. Me alegra 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 = {22 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.