Traballo Fin de Máster

Aprendizaxe profunda e a súa aplicación a imaxes mamográficas

Sara San Luís Rodríguez

Máster en Técnicas Estadísticas

Curso 2017-2018

Proposta de Traballo Fin de Máster

Título en galego: Aprendizaxe profunda e a súa aplicación a imaxes mamográficas
Título en español: Aprendizaje profundo y su aplicación a imágenes mamográficas
English title: Deep learning and its application to mammographic images
Modalidade: Modalidade B
Autora: Sara San Luís Rodríguez, Universidade de Santiago de Compostela
Director/a: Beatriz Pateiro López, Universidade de Santiago de Compostela; Manuel Rodrigo Cabello Malagón, Plain Concepts
Breve resumo do traballo: A Intelixencia Artificial (IA) ten un papel moi importante na actualidade e, dentro desta, está a despuntar -grazas aos avances no campo da computación- a aprendizaxe profunda (*deep learning*), sendo unha rama moi solicitada no mundo empresarial. Neste traballo abordaremos esta técnica de aprendizaxe automática. O traballo dividirase en dúas partes fundamentais: Na primeira parte realizarase unha revisión da literatura existente sobre a aprendizaxe profunda a cal estará centrada na área de análise de imaxes. Nesta revisión veranse os conceptos fundamentais para a comprensión da aprendizaxe profunda, así como as redes neuronais máis empregadas á hora de realizar a análise de imaxes. Na segunda parte, daremos aplicación ás técnicas explicadas enfrontándose a un experimento real onde se poñerán en práctica os coñecementos obtidos para realizar análise de imaxes mamográficas.
Recomendacións: Recoméndase ter cursado as materias de Análise Exploratorio de Datos, Modelos de Regresión e Análise Multivariante así como ter experiencia en computación con R e Python.

Dona Beatriz Pateiro López, Doutora en Estatística e Investigación Operativa da Universidade de Santiago de Compostela, e don Manuel Rodrigo Cabello Malagón, Enxeñeiro de Desenrolo de Software de Plain Concepts, informan que o Traballo Fin de Máster titulado

Aprendizaxe profunda e a súa aplicación a imaxes mamográficas

foi realizado baixo a súa dirección por dona Sara San Luís Rodríguez para o Máster en Técnicas Estadísticas. Estimando que o traballo está rematado, dan a súa conformidade para a súa presentación e defensa ante un tribunal.

En Santiago de Compostela, a 29 de Xuño de 2018.

A directora:

Dona Beatriz Pateiro López

O director:

Don Manuel Rodrigo Cabello Malagón

A autora:

Dona Sara San Luís Rodríguez

Agradecementos

Gustaríame comezar agradecendo a Plain Concepts por brindarme a oportunidade de ter unha primeira toma de contacto co mundo laboral a cal, considero, foi inmellorable. Así mesmo, agradecer aos meus compañeiros e compañeiras por integrarme dende o primeiro día no equipo facendo que a realización das prácticas fose amena e, en particular, ao meu titor Rodrigo pola súa inesgotable paciencia e a súa impecable dirección, realizada cun entusiasmo polo tema contaxioso.

No ámbito académico, gustaríame agradecer á miña titora Beariz Pateiro López tanto o traballo de corrección como as diferentes aportacións realizadas, as cales permitiron aumentar a calidade do traballo. Tampouco quero esquecerme do demais profesorado do Máster en Técnicas Estatísticas, os cales e as cales me aportaron moitos dos coñecementos precisos para a realización deste traballo.

Índice xeral

Resumo

Resumo

Nos últimos anos, o interese da industria e da comunidade investigadora polas tecnoloxías asociadas ao aprendizaxe máquina (*machine learning*) incrementouse notablemente. Diversos factores determinantes, como a dispoñibilidade de entornos computacionais cada vez máis potentes, a aparición das técnicas de aprendizaxe profunda, ou o acceso a un conxunto de datos cada vez maior e de menor coste, converteron esta tecnoloxía nun referente actual en case calquera campo do coñecemento.

En especial, a aprendizaxe profunda permite que redes neuronais compostas de múltiples capas de procesamento aprendan representacións de conxuntos de datos con múltiples niveis de abstracción. Isto, mellorou de forma drástica o estado da arte na análise de imaxes, no recoñecemento de voz e en moitos outros campos como a xenética e a farmacoloxía.

Neste traballo, revísanse distintos modelos de aprendizaxe profunda para a análise de imaxes, destinados á detección e á clasificación de obxectos. Por último, emprégase esta metodoloxía no ámbito da sanidade para axudar na detección e diagnose do cancro de mama.

Abstract

In recent years the interest of the industry and the research community for the technologies associated with Machine Learning has increased significantly. Several determining factors, such as the availability of increasingly powerful computing environments, the emergence of Deep Learning, or access to a growing and smaller cost dataset, have turned this technology into a current reference in almost any field of knowledge.

In particular, deep learning allows neural networks that are composed of multiple processing layers to learn representations of data with multiple levels of abstraction. In this way, deep learning has drastically improved the state-of-the art in image analysis, speech recognition and in many other fields such as genetics and pharmacology.

In this work, we review different models for image analysis, intended for the detection and classification of objects. Lastly, this methodology is used in the health field to help in the detection and diagnosis of breast cancer.

Capítulo 1

Introdución

A Intelixencia Artificial (IA) leva formando parte do imaxinario colectivo e cociñándose en numerosos laboratorios de investigación dende que un grupo de informáticos empregaron o termo na década dos 50. Dende aquela, a IA foi proclamada como a chave para acadar o futuro máis brillante da civilización.

Este concepto, naceu coa idea de construír máquinas complexas que posuíran a mesma intelixencia que os seres humanos. Sen embargo, isto non se acadou, quedando estas expectativas relegadas a películas e novelas de ciencia ficción por seren inviables aínda a día de hoxe. O que si se logrou foi a creación de tecnoloxías capaces de realizar certas tarefas que, ata daquela, eran realizadas exclusivamente por humanos, chegando a realizalas tan ben como eles ou, nalgúns casos, incluso mellor. Entre estes exemplos atópanse, por exemplo, o recoñecemento facial empregado por Facebook ou os sistemas de recomendación empregados por compañías como Netflix, a cal che recomenda series en función dos teus gustos persoais. E non é só en multinacioinais deste calibre onde está a aterrar a IA, senón que cada vez máis empresas queren incorporar esta tecnoloxía.

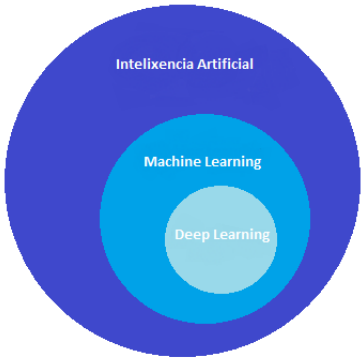

Figura 1.1: Relación entre os termos intelixencia artificial, aprendizaxe automática e aprendizaxe profunda

Estas novas tecnoloxías exhiben certas facetas da intelixencia humana, o que nos leva a preguntarnos de onde vén a mesma. Esta pregunta, lévanos ao seguinte círculo dentro do noso diagrama (ver Figura 1.1):

a aprendizaxe automática (do inglés *machine learning*) (ML). para explicar o que é ML hai dúas grandes definicións. Así, Arthur Samuel describiuna como:

"*Campo de estudo que lle dá aos ordenadores a habilidade de aprender sen ser programados explicitamente*".

Esta definición é anterior á de Tom Mitchell, quen define ML como:

"*Un programa de ordenador dise que aprende dunha experiencia E con respecto a algún tipo de tarefas T e unha medida P que avalía a realización de ditas tarefas se a realización das tarefas T medidas por P melloran coa experiencia E*"

Polo tanto, ML non é outra cousa que o emprego de algoritmos para analizar os nosos datos, aprender deles e, en base a unha nova observación, facer unha predición. A principal característica de ML consiste en "adestrar" a nosa máquina, empregando para iso algoritmos que, estudando inxentes cantidades de datos, lle dan a habilidade de aprender como realizar certa tarefa. En xeral, calquera problema de ML pode ser asignado a dúas grandes ramas: aprendizaxe supervisada e aprendizaxe non supervisada.

Estes dous termos que acabamos de introducir non son completamente alleos para alguén con certo coñecemento estatístico. De feito, non debería sorprendernos que se usen estes termos aquí, pois, realmente, a aprendizaxe máquina está estreitamente relacionada coa aprendizaxe estatística, a cal podería definirse como o campo da estatística que se emprega para realizar predicións en base a datos.

É máis, tanto a aprendizaxe estatística como a aprendizaxe máquina empréganse para resolver o mesmo tipo de problemas, se ben reciben diferente nome segundo a rama da ciencia na que se estean a empregar, así, o termo aprendizaxe máquina está ligado á rama de computación e o termo aprendizaxe estatística resérvase para a rama da estatística. Se ben é certo que algúns autores, como Breiman (2001) [1], sosteñen un enfoque moi afastado do que acabamos de expoñer, en xeral, a comunidade científica non só acepta esta equivalencia, senón que defende a necesidade de mesturar os coñecementos informáticos xunto cos estatísticos para unha mellor aplicación de ambos tipos de aprendizaxe. As persoas que dominan ambas as dúas ramas reciben o nome de *Data Scientists*.

A equivalencia entre estes termos queda aínda máis clara se enumeramos algunha das técnicas empregadas na aprendizaxe profunda, entre as que se atopan, entre outras, a regresión, a aprendizaxe por reforzo e as redes Bayesianas. Dentro destas, están comezando a xogar un papel moi importante as redes neuronais, inspiradas no funcionamento do cerebro humano, entre as que destacan as redes neuronais profundas, o que se coñece como aprendizaxe profunda (do inglés *deep learning*) (DL). Ditas redes, levan estando presentes dende a década dos 80, pero, debido a que tanto o tempo como o volume de datos preciso para acadar bos resultados eran moi elevado, non suscitaron moito interese.

Isto cambiou nos últimos anos, pois o progreso, as necesidades e as expectativas creadas polas redes neuronais son signos da era de *Big Data*, a cal trouxo consigo o almacenamento masivo de datos de todo tipo (imaxes, texto, transaccións...). Outro gran aliado foi o incremento das capacidades computacionais, nas cales destacan as GPU, que axudaron grazas ao seu procesamento en paralelo a que o proceso sexa máis rápido, máis económico e máis potente. Todos estes recursos, están hoxe ao noso alcance máis do que nunca estiveron e é grazas a estes, que as redes neuronais, en concreto as redes neuronais profundas, se perfilan como un competidor moi serio no campo da IA, tanto no procesamento de voz e de linguaxe natural, como no campo de recoñecemento visual, sendo neste último no que nos centraremos ao longo deste traballo.

Así pois, no Capítulo 2, preséntanse os fundamentos básicos das técnicas de aprendizaxe profunda para, seguidamente, explicar en que consisten as mesmas. No Capítulo 3, introdúcense as redes neuronais máis empregadas para a análise de imaxes, centrándonos especialmente na detección e clasificación de obxectos, así como o estado da arte actual das mesmas e, no Capítulo 4, presentamos unha posible aplicación delas para a detección de cancro de mama en imaxes mamográficas, reservando un apartado no que se contempla o traballo futuro a realizar. Xa por último, no Capítulo 5, preséntanse as

conclusións do traballo.

Como comentamos, o interese da industria e da comunidade investigadora polas tecnoloxías asociadas á aprendizaxe máquina, incrementouse notablemente durante os últimos anos. Diversos factores, como a dispoñibilidade de entornos computacionais cada vez máis potentes, a irrupción das tecnoloxías de *Big Data*, a mellora nos algoritmos de aprendizaxe, a aparición da aprendizaxe profunda ou o acceso a un conxunto de datos cada vez maior e a menor custo converteu esta tecnoloxía nun referente actual en case calquera campo do coñecemento.

Dende que comecei a escoitar falar desta área da Estatística interesoume moito polo que cando vin a posibilidade de realizar as prácticas sobre este tema, non o dubidei, sendo consciente de como esta área pode chegar a mellorar a nosa vida.

Así pois, non quero finalizar a introdución deste Traballo de Fin de Máster sen deixar constancia da formación recibida en Plain Concepts, empresa na que realicei as prácticas que levaron á realización deste traballo. Durante dita formación, aprendín gran parte das cousas que se expoñen a continuación pois, ademais dunha lectura detida da bibliografía referenciada, tamén asistín a conferencias e charlas impartidas maioritariamente polo meu titor na empresa e realicei diversos cursos, dos que tomei algúns dos exemplos presentes no documento, entre os que me gustaría destacar o curso Machine Learning de Andrew Ng, accesible na plataforma de cursos online Coursera[1], e o curso de Deep Learning Explained de Microsoft impartido por Sayan Pathak e Roland Fernandez na plataforma online edX[2].

Quero destacar tamén os libros de Hands-On Machine Learning with Scikit-Learn & Tensorflow de Aurélien Géron [10] e o libro de Deep Learning A Practitioner's Approach de Josh Patterson e Adam Gibson [30] nos que me baseei para elaborar gran parte do traballo.

[1] Pode accederse ao mesmo empregando a ligazón `https://www.coursera.org/learn/machine-learning`.

[2] Pode accederse ao mesmo empregando a ligazón `https://courses.edx.org/courses/course-v1:Microsoft+DAT236x+4T2017/course/`.

Capítulo 2

Fundamentos da aprendizaxe profunda

A aprendizaxe profunda permite que redes neuronais artificiais, compostas por múltiples capas de procesamento, aprendan representacións de datos con múltiples niveis de abstracción. Grazas a isto, o estado da arte no recoñecemento de voz e no recoñecemento de obxectos visuais, entre outros campos, mellorou drasticamente nos últimos anos.

Dentro destas redes neuronais destacan as catro grandes arquitecturas que se enumeran a continuación:

- Redes neuronais non supervisadas preadestradas
- Redes neuronais convolucionais
- Redes neuronais recorrentes
- Redes neuronais recursivas

Dentro destas, as redes neuronais convolucionais son as que máis avances provocaron no procesamento de vídeo e imaxes mentres que as recorrentes son das máis novidosas dentro do campo de procesamento de texto, máis literatura pode consultarse en LeCun et al. (2015) [26].

Se ben a utilidade que ten a aprendizaxe profunda está clara, definila é un reto en si mesmo pois, a día de hoxe, aínda non se conseguiu dar unha definición coa que estivera de acordo toda a comunidade científica o cal se debe, en gran parte, ao rápido que dita aprendizaxe foi evolucionando ao longo das décadas. Unha das definicións máis empregada pola súa utilidade considera como aprendizaxe profunda a todas aquelas redes neuronais que teñen máis de dúas capas. Non obstante, neste traballo consideraremos como aprendizaxe profunda todas aquelas redes neuronais cun gran número de parámetros e capas en calquera das catro arquitecturas mencionadas. Ditas redes tamén poden recibir o nome de redes neuronais profundas se ben nós empregaremos o termo de rede neuronal para referirnos a elas.

Antes de afondar no obxectivo deste capítulo preséntanse algúns conceptos sobre as redes neuronais co fin de dar unha base sobre a cal asentar os fundamentos da aprendizaxe profunda.

2.1. Inspiración biolóxica das redes neuronais

As redes neuronais son un modelo inspirado no funcionamento do cerebro humano, máis concretamente nas conexións interneuronais. As neuronas están compostas polo corpo, o cal contén un núcleo voluminoso central que recibe o nome de *soma* e no cal se atopan a maioría das compoñentes complexas

da célula; as *dendritas*, que son a ramificación e unha extensión longa do soma que recibe o nome de *axón*. A lonxitude do axón é variable podendo chegar a ser entre dez e mil veces máis longo que o soma. Preto do final do axón, este divídese en máis ramificacións chamadas *telodendrias* e, ao final destas, atópanse as *terminais sinápticas* que son as encargadas de realizar a conexión coas dendritas doutras neuronas. Na Figura 2.1[1] pódense ver cada unha das partes mencionadas, podendo consultarse a literatura existente en Kandel et al (2000) [19].

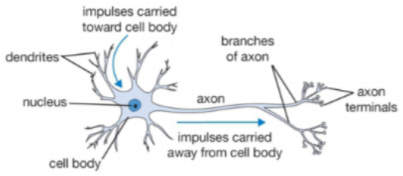

Figura 2.1: Estrutura da neurona.

Se ben, polo dito ata o de agora, as neuronas biolóxicas parecen comportarse de xeito moi sinxelo, na realidade están organizadas nunha basta rede de billóns de neuronas onde cada unha se conecta a miles doutras neuronas de forma que, pese a "sinxeleza" das mesmas, poden realizarse cómputos altamente complexos. En canto á arquitectura destas redes neuronais biolóxicas parece que as neuronas están a miúdo organizadas en capas consecutivas (ver Figura 2.2).

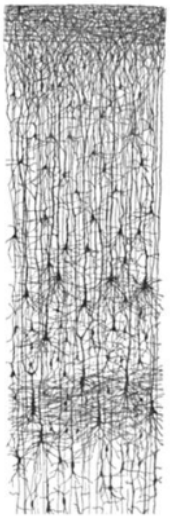

Figura 2.2: Múltiples capas dunha rede neuronal biolóxica (debuxo realizado por Ramón y Cajal).

Unha vez explicada a estrutura da neurona vexamos brevemente o funcionamento da mesma para logo comprender mellor como estas características son empregadas nas redes neuronais. Así pois, as neuronas teñen tres funcións principais as cales poden ser consultadas en Stephen M. Stahl (2013) [39]:

- A primeira delas, recibir os impulsos, sexan eléctricos ou químicos, a través das dendritas que pasan ao corpo da neurona.
- A segunda lévase a cabo no soma da neurona, onde se leva a cabo o procesamento da información que lles chega.

[1] Tomada de Li (2017) [25].

- A última delas é o envío da saída a outras neuronas a través das terminais sinápticas. Cando unha neurona recibe suficiente número de sinais doutras neuronas, comeza a enviar as súas propias sinais en cuestión de milisegundos.

As principais características que se aplican ás redes neuronais son principalmente dúas. A primeira delas é que a unidade máis básica da rede neuronal é a neurona artificial, á que chamaremos indistintamente *nodo* a partir de agora. Estes nodos son modelados como as neuronas biolóxicas e, ao igual que elas, son estimuladas mediante impulsos.

A segunda das características é que, do mesmo xeito que as neuronas son as células encargadas da comunicación química no cerebro, nas redes neuronais son os nodos os encargados de pasar parte da información que reciben a outros nodos realizando para iso as transformacións precisas.

Finalmente, do mesmo xeito que as neuronas no cerebro poden ser adestradas para transmitir só aqueles sinais que poden ser útiles para realizar certa actividade cerebral, os nodos da rede neuronal tamén poden ser adestrados para transmitir só aquela información que resulte útil para o fin para a cal está a ser adestrada.

Ao longo deste capítulo tratarase de explicar tanto as transformacións que se levan a cabo nos nodos como de ver como as redes neuronais son capaces de exercer un papel similar ás neuronas no cerebro mediante bits e funcións. Vexamos cal foi a necesidade que motivou as redes neuronais así como o primeiro achegamento ás mesmas.

2.2. O comezo das redes neuronais artificiais

Como xa dixemos, a aprendizaxe profunda e, en especial as redes neuronais, é unha idea bastante antiga que se volveu retomar na actualidade para ser empregada en moitos algoritmos de ML. Deste xeito, as redes neuronais convertéronse en fortes competidoras doutros algoritmos clásicos dentro deste campo, como poden ser a regresión lineal ou a regresión loxística.

Pero se xa tiñamos eses algoritmos, que fixo que fose necesario un novo?

2.2.1. Motivación

Para motivar a discusión sobre a necesidade das redes neuronais comecemos propoñendo algúns exemplos de ML onde precisamos de hipóteses complexas non lineais para resolvelos. Consideremos, sen ir máis lonxe, un problema de clasificación onde o gráfico de dispersión dos nosos datos aparece recollido na Figura 2.3[2].

Se quixésemos aplicar regresión a este problema, poderíamos intentalo aplicando regresión loxística, xa que temos unha variable dicotómica e na regresión loxística a variable resposta é discreta.

Para isto lembremos que a hipótese da regresión loxística está definida como:

$$h_\theta(x) = g(\theta^T x),$$

onde a función g recibe o nome de *función de activación*[3], a cal neste caso será a función sigmoide definida como:

$$g(z) = \frac{1}{1 + e^{-z}}.$$

[2]Tomada do curso Machine Learning `https://www.coursera.org/learn/machine-learning`.

[3]Máis adiante explicarase con máis detalle o papel da función de activación así como os diferentes tipos que hai.

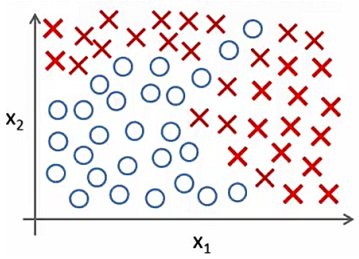

Figura 2.3: Problema de clasificación non lineal.

Unha forma de intentar resolver este problema de clasificación, sería empregar como variable explicativa coeficientes polinómicos de forma que a hipótese fose:

$$h_\theta(x) = g(\theta_0 + \theta_1 x + \theta_2 x_2 + \theta_3 x_1 x_2 + \theta_4 x_1^2 x_2 + \theta 5 x_1^3 x_2 + \theta_6 x_1 x_2^2 + ...)$$

Deste xeito, poderíamos chegar a obter unha hipótese o suficientemente boa separando os dous conxuntos[4], se ben isto non garantiría que fose a dar bos resultados ante novas observacións. De feito, isto funciona ben con dúas variables, pero moitos dos problemas cos que se traballan na industria ou en campos de investigación teñen moitas máis de dúas variables, o que pode complicar sumamente o que acabamos de ver.

Imaxinemos agora que temos 100 variables diferentes con información sobre dunha casa

$$x_1, x_2, x_3, ..., x_{100}$$

onde x_1 poderían ser os metros cadrados, x_2 o número de habitacións... E supoñamos que queremos predicir a *odds* de que a casa sexa vendida dentro dos seis meses seguintes, estamos polo tanto ante un problema de clasificación. Seguindo coa idea comentada anteriormente, se quixésemos aplicar regresión loxística con termos polinómicos incluíndo tan só os termos cadráticos, isto é, aqueles da forma:

$$x_1^2, x_1 x_2, x_1 x_3, ..., x_1 x_{100}, x_2^2, x_2 x_3, ..., x_2 x_{100}, ...$$

de incluír todos os termos cadráticos para o caso de $n = 100$ remataríamos tendo aproximadamente unhas 5000 variables. Ademais, o número de características medra na orde de

$$O(n^2) \approx \frac{n^2}{2}$$

de xeito asintótico, onde n é o número de variables orixinais. Polo tanto, é obvio que a estratexia seguida no exemplo anterior, onde só tiñamos dúas variables, non é tan boa idea agora posto que, debido á gran cantidade de características que temos, non sería difícil rematar sobreaxustando o conxunto de adestramento e, ademais, sería moi custoso computacionalmente.

Unha alternativa podería ser traballar tan só cun subconxunto destas variables. Un exemplo disto podería ser incluír tan só aquelas variables da forma $x_i^2, i \in \{1, ..., 100\}$. Deste xeito, teríamos reducido en gran medida a cantidade de variables pero non serían suficientes e non nos permitirían axustar o noso conxunto de datos[5] (ver Figura 2.3).

[4]Lema: Para calquera conxunto de puntos $n+1$ da forma (x_i, y_i) con $x_i \neq x_j$ para $i \neq j$ hai un único polinomio f de grado ao sumo n tal que $f(x_i) = y_i$ para todo i.

[5]Notemos que, se ben incluír estas variables non nos permiten axustar o noso conxunto de datos, se incluísemos tamén $x_i, i \in \{1, ..., 100\}$ poderíamos axustar hipóteses como elipses.

Ata o de agora só falamos de incluír os termos cadráticos, pero, que ocorrería se quixésemos incluír tamén os termos cúbicos? A resposta é sinxela: o número de características aumentaría moito máis, de feito, teríamos da orde de 170000 características. Polo tanto, incluír estas características polinómicas das que vimos falando aumentaría drasticamente o espazo de características o que non parece un bo xeito de atopar características adicionais coas que construír clasificadores para un valor de n grande e, en moitos dos problemas tanto de ML como de DL, n será grande.

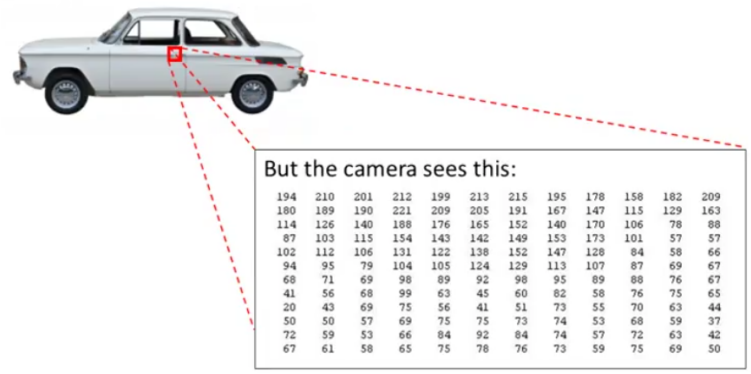

Figura 2.4: Diferenza entre a visión humana e a visión por computadora.

Para ver isto último, consideremos un problema de visión por computador e supoñamos que queremos adestrar un clasificador para examinar se unha imaxe é ou non un coche. Así, supoñamos que temos o coche da Figura 2.4[6], nós vemos claramente que se trata dun coche pero o ordenador non está a ver o coche do mesmo xeito que nós. Así, se facemos zoom nunha pequena parte da imaxe (rectángulo vermello na Figura 2.4), podemos observar que a información que o ordenador ten non é outra cousa que unha matriz onde cada entrada representa os valores de intensidade dos diferentes píxeles que conforman a imaxe. O problema que nos ocupa consiste entón en observar esta matriz de intensidades de píxeles e, a partir dos mesmos, decidir se estes números representan o tirador dun coche ou non.

Figura 2.5: Imaxe para aclarar a diferenza entre a visión humana e a visión por computadora.

Máis concretamente, cando empregamos ML para construír un detector de coches o que facemos é obter un conxunto de datos de fotografías de coches e doutros obxectos que non o sexan xunto coas súas respectivas etiquetas, é dicir, "coche" e "non coche". O clasificador é adestrado sobre estas imaxes e, unha vez finalizado o adestramento, pasaríamoslle a Figura 2.4 como preguntándolle: Que é isto? E esperaríamos que identificase un coche.

Pese ao explicado ata o de agora, parece non estar claro aínda o porqué precisamos hipóteses non lineais. Para responder esta pregunta, vexamos algunha das imaxes que lle pasaríamos ao noso algoritmo e seleccionemos un par de localizacións de píxeles nestas imaxes, para simplificalo consideraremos só

[6]Tomada do curso Machine Learning `https://www.coursera.org/learn/machine-learning`.

dúas (ver Figura 2.5)[7], e representemos o punto obtido en $\mathbb{R}^2$, como podemos ver na Figura 2.6[8] (esquerda). Supoñamos que se seguísemos facendo isto con tódalas imaxes obteríamos un gráfico de dispersión como o da Figura 2.6 (dereita). Así vemos que aquelas fotografías nas que aparece un coche aparecen en certas rexións do gráfico, mentres que aquelas que non conteñen coches aparecen noutras.

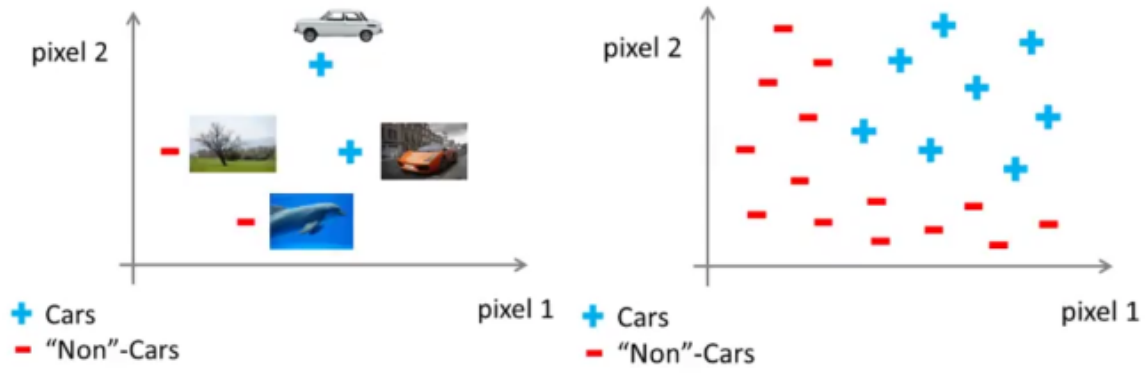

Figura 2.6: Gráficos de dispersión: en construción (esquerda) e final (dereita).

Á vista deste último gráfico, ver de novo a Figura 2.6 (dereita) faise evidente a necesidade de hipóteses non lineais xa que non podemos separar os dous conxuntos cunha hipótese lineal. Ata o de aquí, isto non difire moito do problema que vimos de ver, vexamos entón o outro punto do que falamos: o número de características empregado.

Supoñamos para isto que a nosa imaxe é de 50×50 píxeles[9], polo que teríamos un total de 2500 píxeles, o que equivale a que o noso conxunto de características teña de dimensión $n = 2500$ onde a nosa variable $\mathbf{x}$ sería unha vector con todas as intensidades dos píxeles, as cales nas representacións clásicas do ordenador adoitan tomar valorescomprendidos entre 0 e 255:

$$\mathbf{x} = \begin{pmatrix} x_1 = \text{Intensidade do primeiro píxel} \\ x_2 = \text{Intensidade do segundo píxel} \\ \dots \\ x_m = \text{Intensidade do n-ésimo píxel} \end{pmatrix},$$

se estamos traballando coa escala de grises teríamos, para o exemplo que nos ocupa, $n = 2500$ como acabamos de dicir. Sen embargo, notemos que se estivésemos empregando imaxes RGB o tamaño de n ascendería ata $n = 7500$ posto que teríamos tres canles, un correspondente a cada cor: vermello, verde e azul. Así pois, repetindo os cálculos anteriores, se agora quixésemos incluír todas as características cadráticas teríamos aproximadamente tres millóns[10] de características. Esta cantidade é demasiado grande para chegar a ser incluso razoable, facendo as esixencias computacionais altísimas.

[7] Tomada do curso Machine Learning `https://www.coursera.org/learn/machine-learning`.
[8] Tomada do curso Machine Learning `https://www.coursera.org/learn/machine-learning`.
[9] Notemos que esta cantidade é de por si reducida xa que mesmo as fotografías sacadas coa cámara do teléfono móbil ascenden aos 4608×3456 píxeles.
[10] Se a imaxe fose de 100×100 píxeles a cantidade ascendería ata 50 millóns.

É por todo isto que métodos como a regresión loxística non resultan una boa opción para estes problemas, facéndose evidente a necesidade dun novo método que permita aprender hipóteses non lineais complexas para n grandes. É aquí onde entran en xogo as redes neuronais, as cales nos ocuparan no resto do presente documento, comecemos.

2.2.2. A neurona artificial

En 1943, Warren McCulloh e Walter Pitts propuxeron un modelo moi sinxelo da neurona biolóxica, bautizado como *neurona artificial*, que foi a precursora do perceptrón. Esta tiña simplemente unha ou máis entradas binarias as cales poderíamos pensalas como entradas de aceso ou apagado. O que facía esta neurona artificial era activar a súa saída cando un certo número das entradas estaban activas (acesas). Deste xeito, McCulloh e Pitts demostraron que, incluso cun modelo tan simple como este, era posible construír unha rede de neuronas artificiais tal que fosen capaces de calcular simples proposicións lóxicas como pode consultarse McCulloh e Pitts (1943) [29]. Parémonos un pouco nisto vendo a Figura 2.7[11].

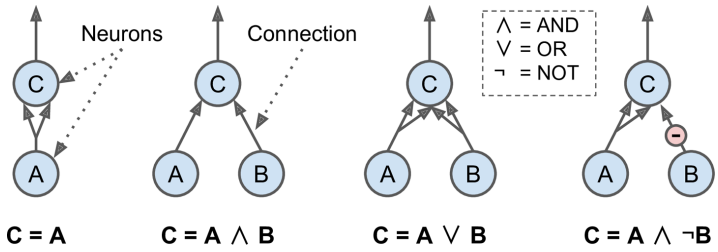

Figura 2.7: A neurona artificial realizando cálculos lóxicos.

A primeira das redes amosadas xoga o papel da función identidade: cando a neurona A está activada a neurona C tamén se activa xa que recibe os dous sinais da neurona A. Do mesmo xeito, cando a neurona A estea desactivada tamén o estará a neurona C.

A segunda rede realiza o operador lóxico AND: a neurona C activarase só cando o estean as neuronas A e B de forma simultánea.

Na terceira, estase a realizar o operador lóxico OR, é dicir, a neurona C actívase sempre e cando se active polo menos unha das neuronas A ou B.

Xa por último, a cuarta rede que aparece representada baséase na suposición de que unha entrada pode inhibir a actividade dunha neurona como ocorre coas neuronas biolóxicas. Isto aparece reflectido na cuarta rede onde C só se activaría se a neurona A estivese activada e a neurona B non o estivese. Notemos tamén que se a neurona A estivese sempre activa sería como ter un operador lóxico NOT, isto é, a neurona C estaría activa sempre que a neurona B estivese apagada e viceversa.

2.2.3. O perceptrón

O *perceptrón*, unha das arquitecturas máis simples das redes neuronais artificiais, é un modelo lineal empregado para a clasificación binaria o cal se basea nunha relación de entrada-saída. Foi inventado polo psicólogo Frank Rosenblatt en 1957, consultar Rosenblatt (1957) [35], como un modelo simplificado de como operan as neuronas no cerebro humano tomando algúns dos conceptos empregados por McCulloh e Pitts (1943) [29] como os limiares que activaban, ou non, ás neuronas artificiais.

[11] Tomada de Géron (2017) [10].

De feito, o perceptrón vén a ser case como a neurona artificial de McCulloh e Pitts pero cunha pequena modificación: agora tanto a entrada como a saída son números, en vez de valores binarios como acontecía antes, e cada unha das conexións ten asociado un peso.

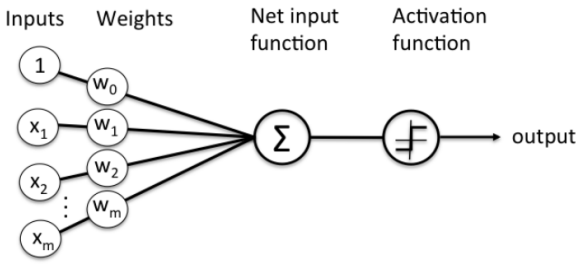

Figura 2.8: Representación esquemática do perceptrón de Rosenblatt.

Na Figura 2.8[12] podemos ver unha representación esquemática do perceptrón, na cal podemos ver que estamos considerando m entradas así como os seus pesos asociados aos que denotaremos por $w_i, i \in \{1, ..., m\}$. Estes pesos son coeficientes que escalan, amplifican ou minimizan, o sinal de entrada dunha determinada neurona da rede.

Notemos que a entrada x_0 ten asociado un peso igual a 1, esta recibe o nome de *unidade de sesgo* e o seu papel é importante posto que permite a translación da función de activación e asegura que polo menos algún nodo de cada capa é activado sen importar a forza do sinal.

Con esta información calcúlase a suma ponderada:

$$z = w_0x_0 + w_1x_1 + w_2x_2 + ... + w_mx_m = \mathbf{W}^T\mathbf{x},$$

a cal se lle pasa á función de activación $h_{\mathbf{W}}(\mathbf{x})$. Estas funcións son as que se encargan de modelar o comportamento da rede neuronal mediante a transformación das entradas, os pesos e as unidades de sesgo. Os resultado destas transformacións actúan como entrada para a seguinte capa de nodos ou como saída da rede neuronal na última capa. Cando un nodo transmítelle un valor non nulo a outro nodo dise que este novo nodo está *activado*.

Poden empregarse distintas funcións de activación, se ben, para ilustrar como funciona, empregaremos a función Heaviside. Na sección 2.3.2 definiremos algunha das funcións de activación máis empregadas nas redes neuronais e discutiremos as vantaxes dunhas fronte a outras. A función Heaviside, á cal denotaremos por H, vén definida por:

$$H(z) = \begin{cases} 0 & z < 0 \\ 1 & z \geq 0 \end{cases}.$$

O limiar desta función sitúase no 0.5, é dicir, a saída será un único valor real binario dependendo do valor de entrada, así será 1 cando a suma ponderada sexa positiva ou nula e 0 no caso de que a suma ponderada sexa negativa. Esta saída é a saída do perceptrón e danos unha posible clasificación, neste caso clasificación binaria, dos valores de entrada. Esta é tamén unha das principais diferencias coa regresión loxística pois, mentres que con esta obtiñamos as probabilidades das clases, cos perceptróns realizamos predicións baseándonos no limiar.

Parémonos agora a ver o papel da unidade de sesgo, así, supoñamos que $x_0 = -1$, neste caso isto conlevaría que a suma dos pesos aprendidos tivese que ser bastante maior que 1 para obter un valor

[12]Tomada de `https://sebastianraschka.com/Articles/2015_singlelayer_neurons.html`.

de 1 na saída. Polo tanto, o que fai este termo, ademais do xa dito, é mover a rexión de decisión. A unidade de sesgo, ao igual que os pesos, é modificada a través do algoritmo de aprendizaxe pero non se ve afectada polos valores de entrada.

É xusto neste algoritmo onde radica a principal diferenza entre a neurona artificial de McCulloch e Pitts e o perceptrón de Rosenblatt xa que que a primeira non tiña ningún mecanismo de aprendizaxe, o que é crucial para poder ser empregada no campo da IA. Aquí é onde destacou o perceptrón posto que Rosenblatt, inspirado polo traballo de Hebb (1949) [15], o cal suxería que cando unha neurona biolóxica activa a outra neurona a conexión entre estas dúas medra máis forte, atopou un xeito para que os perceptróns aprendesen por si mesmos.

Así, os perceptróns adéstranse baseándose nesta idea e tendo en conta o erro cometido pola rede de xeito que non se reforcen aquelas conexións que conduzan a un resultado erróneo. É dicir, dado un conxunto de adestramento, formado tanto polos datos como polas súas correspondentes etiquetas, o perceptrón debería aprender unha función para que, para cada exemplo, se reforzasen aquelas conexións que contribuíron a realizar unha predición correcta. Para isto o que se fai é, dado o xa mencionado conxunto de adestramento:

1. Inicializar o perceptrón con pesos aleatorios pequenos ou nulos.
2. Para cada unha das entradas do noso conxunto de adestramento calcular a saída do perceptrón.
3. Comprobar se a saída do perceptrón coincide coa saída que se suporía correcta para a entrada. En caso contrario actualízanse os pesos que terían contribuído a unha mellor predición.
4. Pasar ao seguinte exemplo do noso conxunto de adestramento e repetir os pasos 2 e 3 ata que o perceptrón non cometa erros.

Este algoritmo simple e intuitivo aparece recollido na Ecuación 2.1 :

$$w_{i,j}^{k+1} = w_{i,j}^{k} + \lambda(y_j - \hat{y}_j) \cdot x_i \tag{2.1}$$

- $w_{i,j}^{k}$ é o peso que ten a conexión entre a neurona de entrada i-ésima e a neurona de saída j-ésima, na iteración k .
- x_i é a i-ésima entrada da entrada actual do noso conxunto de datos.
- $\hat{y}_j$ é a saída calculada pola neurona j-ésima para a entrada actual.
- y_j é o valor que se esperaría obter como saída j-ésima neurona para a entrada actual.
- λ é a taxa de aprendizaxe[13].

Unha vez explicado brevemente o funcionamento do perceptrón, na Figura 2.9[14] podemos ver a semellanza entre a neurona biolóxica e o perceptrón do que estamos a falar.

Previamente, xa dixemos que as redes neuronais son modelos computacionais que comparten algunhas propiedades co cerebro humano, no sentido de que moitas unidades simples traballan de forma paralela sen unha unidade centralizada de control. Tras os novos termos introducidos, podemos ver agora os pesos entre as distintas unidades como a principal forma de almacenamento de información na rede neuronal e, a súa actualización, como a principal forma de aprender a nova información que a rede neuronal recibe.

[13] A taxa de aprendizaxe será explicada na Sección 2.3.5.

[14] Tomada do curso Machine Learning `https://www.coursera.org/learn/machine-learning`.

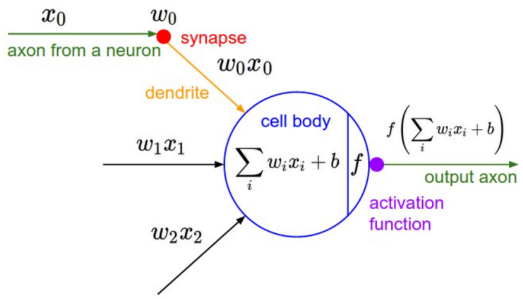

Figura 2.9: Semellanza entre a neurona biolóxica e o perceptrón de Rosenblatt.

Ademais, Rosenblatt demostrou que se o conxunto de adestramento é linealmente separable[15] o algoritmo converxerá a unha solución, se ben esta non ten porque ser única xa que cando temos un conxunto de datos linealmente separables hai unha infinidade de hiperplanos que os poden separar. Pola contra, unha pega deste algoritmo é que non converxerá se as entradas non son linealmente separables. Ben é certo que isto acontecía - e segue a acontecer- en tódolos problemas de clasificación como, por exemplo, na regresión loxística, pero os investigadores tiñan tantas esperanzas postas no perceptrón que a súa decepción foi maiúscula.

Pese ás grandes expectativas creadas polo perceptrón, as limitacións que vimos de comentar levaron a unha diminución no interese pola intelixencia artificial dando paso ao que se coñece como o inverno da IA que duraría ata 1980 coa chegada do algoritmo de propagación cara atrás (*backpropagation*).

Así, en 1969, Marvin Minsky e Seymour Papert escribiron o libro *Perceptróns* (1969) onde deixaban evidencia das debilidades atopadas nos perceptróns facendo especial fincapé no problema de clasificación XOR dicindo que para poder resolvelo habería que empregar múltiples capas de perceptróns, o que hoxe en día recibe o nome de *redes neuronais multicapa*, pero o algoritmo deseñado por Rosenblatt non servía para traballar con multicapas. Intuitivamente é sinxelo darse conta do porqué este algoritmo non funcionaba e é que neste só se especifica a saída correcta para a última das capas, polo que non se coñecía ningún xeito de saber como axustar os pesos das capas anteriores a esta. Afondemos entón nestas redes neuronais multicapa.

2.2.4. O perceptrón multicapa

Unha vez introducido o perceptrón e, xunto con el, algunhas ideas de conceptos básicos, podemos dicir que o comportamento das redes neuronais vén dado pola súa arquitectura a cal pode ser definida empregando:

- O número de neuronas ou nodos artificiais.
- O número de capas.
- O tipo de conexións entre as capas.

Ocuparémonos a continuación do perceptrón multicapa. Este non é máis que a combinación de diferentes capas formadas por nodos. O perceptrón multicapa está composto por tres tipos de capas:

[15] Un conxunto linealmente separable é aquel para o cal podemos atopar os valores dun hiperplano que divida limpamente as dúas clases do conxunto de datos.

- Capa de entrada: o perceptrón multicapa ten unha única capa de entrada formada por aqueles nodos que introducen os datos de entrada á rede neuronal. Nos nodos que conforman a capa de entrada non se produce ningún tipo de procesamento e, xeralmente, o número de nodos da capa de entrada coincide co número de entradas da rede neuronal.
- *Capas ocultas*: as capas ocultas reciben este nome por estar entre a capa de entrada e a capa de saída. Polo tanto, as súas entradas proveñen de capas anteriores e as súas saídas pasan a nodos de capas posteriores. Este tipo de capas son moi importantes pois son as que permiten ás redes neuronais modelar funcións non lineais, unha das principais limitacións do perceptrón.
- Capa de saída: esta capa está formada por aqueles nodos que dan as saídas que, dependendo da función de activación da nosa rede, poden ser valores reais (regresión) ou un conxunto de probabilidades (clasificación).

Na Figura 2.10 podemos ver a representación esquemática dun perceptrón multicapa con dúas capas ocultas nas que as unidades de sesgo non aparecen representadas.

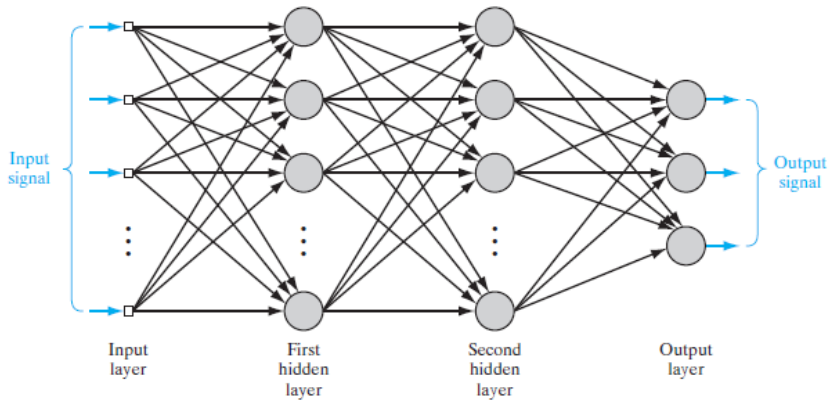

Figura 2.10: Representación esquemática dun perceptrón multicapa.

Igual que hai distintos tipos de capas e que o número das mesmas pode ser variable, estas tamén poden estar conectadas entre elas de xeito distinto dando lugar a dous tipos de perceptrón multicapa:

- Perceptrón multicapa *totalmente conectado*: este tipo de perceptrón multicapa é aquel no que cada saída dun nodo da capa k é entrada de todos os nodos da capa $k+1$. O perceptrón multicapa aparece representado na Figura 2.10
- Perceptrón multicapa *localmente conectado*: ao contrario que o anterior, diremos que un perceptrón multicapa é localmente conectado cando cada nodo da capa k é entrada dunha serie de nodos da capa $k+1$.

Unha vez explicada a arquitectura básica deste perceptrón multicapa sinalemos unha principal diferenza co perceptrón, máis alá da de estar traballando agora con varias capas. Para iso notemos que agora xa non estamos falando de perceptróns senón de nodos xa que, igual que evolucionou a arquitectura da rede neuronal, tamén o fixo a súa unidade fundamental. A diferenza da que estamos a falar radica unicamente en que agora a función de activación destes nodos será diferente en función do propósito específico de capa da rede [16]sendo a única condición que ditas funcións sexan diferenciables. Permitindo diferentes funcións de activación o que se consegue é obter diferentes tipos de valores de saída.

[16]Lembremos que a función de activación do perceptrón era a función Heaviside.

Introduzamos previamente algo máis de notación do perceptrón multicapa:

- Denotaremos por $\mathbf{W}^{(k)}$ á matriz formada por aqueles pesos que van dende a capa k ata a capa $k+1$. Notemos que se unha rede ten n_k nodos na capa k, engadindo a unidade des sesgo serían n_k+1, e n_{k+1} nodos na capa $k+1$, entón $\mathbf{W}^{(k)} \in \mathbb{R}^{n_{k+1}\times(n_k+1)}$.

- Denotaremos por $a_i^{(k)}$ ao valor calculado, e que é emitido como saída, polo nodo i na capa k, ao que chamaremos *activación*. Por exemplo, $a_3^{(2)}$ é a activación do terceiro nodo da segunda capa. E, do mesmo xeito, $a^{(2)}$ é o vector formado por todas as activacións que conforman a segunda capa.

- Denotaremos por $z_i^{(k)}$ ao valor de entrada do nodo i na capa k. Na capa de entrada este valor coincide co dato de entrada e, en capas posteriores, calcúlase como o produto das activacións da capa $k-1$, $a^{(k-1)}$, e a matriz de pesos $\mathbf{W}^{(k-1)}$.

A efectos de ilustrar as operacións que se levan a cabo nas neuronas artificiais denotemos por n_k o número de nodos na capa k da rede e por m o número total de capas da nosa rede neuronal.

As n_1 entradas da nosa rede neuronal pasan a través da capa de entrada á capa oculta, onde se calcula a súa activación. Esta activación sae cara a seguinte capa oculta ou, en caso de non haber, cara a capa de saída onde se calcula o resultado final. O cálculo realizado en cada neurona para o cómputo desta activación aparece recollida na Ecuación 2.2 onde g denota a función de activación e os c_j a saída calculada pola neurona j da capa anterior.

$$a_i^{(k)} = g\left(\sum_{j=0}^{n_{k-1}} w_{ij}^{(k-1)} c_j^{(k-1)}\right), \quad j \in \{1,...,\#n_{k-1}\}, i \in \{1,...,\#n_k\}, k \in \{2,...,m\} \tag{2.2}$$

Aclaremos isto cun exemplo. Consideremos unha rede neuronal con unha capa de entrada, unha capa oculta e unha de saída ($m=3$). Cada unha delas con $n_1=3$, $n_2=3$ e $n_3=1$ nodos respectivamente. Como temos 3 nodos de entrada, teremos 3 variables de entrada (unha por cada nodo), ás que denotaremos por x_1, x_2, x_3 xunto coa unidade de sesgo, á cal denotaremos por x_0. Así, se quixésemos calcular a activación do primeiro nodo da segunda capa (no noso exemplo a capa oculta) teríamos que resolver a seguinte ecuación:

$$a_1^{(2)} = g\left(z_1^{(2)}\right) = g\left(w_{10}^{(1)}x_0 + w_{11}^{(1)}x_1 + w_{12}^{(1)}x_2 + w_{13}^{(1)}x_3\right)$$

Do mesmo xeito, se quixésemos calcular a activación da capa de saída teríamos agora:

$$h_{\mathbf{W}}(\mathbf{x}) = a_1^{(3)} = g\left(w_{10}^{(2)}a_0^{(2)} + w_{11}^{(2)}a_1^{(2)} + w_{12}^{(2)}a_2^{(2)} + w_{13}^{(2)}a_3^{(2)}\right)$$

Notemos que esta hipótese está parametrizada en función de $\mathbf{W}$ polo que, segundo varíen os pesos, obteremos diferentes hipóteses e, por conseguinte, diferentes solucións. As operacións aquí recollidas reciben o nome de *propagación cara adiante* (*forward propagation*).

Ata o de agora temos introducido e explicado o funcionamento do perceptrón multicapa, pero non temos falado de como pode ser adestrado. A resposta a isto, se ben tardou tempo en atoparse, baseouse nunha vella regra de cálculo: a regra da cadea. Antes de discutir en profundidade este punto (ver Sección 2.3.4), introduciremos outros conceptos fundamentais para a completa comprensión das redes neuronais.

2.3. Definicións básicas

Tras esta pequena introdución histórica ao mundo das redes neuronais e aproveitando que xa se introduciron algúns dos termos, imos proceder agora a dar algunhas definicións básicas que nos permitan entender mellor tanto as redes neuronais como os algoritmos empregados polas mesmas.

2.3.1. Conxuntos de datos

Cómpre dicir que tanto en ML como en DL unha das tarefas máis comúns é o estudo e construción de algoritmos que poden aprender a partir de datos e realizar predicións ou ben tomar decisións en base a esta aprendizaxe. Estas predicións e decisións son realizadas construíndo modelos matemáticos a partir dos datos de entrada.

Estes datos, empregados para construír o modelo final, proceden normalmente de diferentes conxuntos de datos destacando principalmente tres, os cales son empregados en diferentes etapas de creación do modelo: o conxunto de adestramento, o conxunto de test e o conxunto de validación.

Vexamos entón, antes de introducir conceptos máis relacionados coa aprendizaxe profunda en si, como dividiremos o conxunto de datos co que traballaremos e o papel xoga cada un destes conxuntos no noso modelo.

Conxunto de adestramento

O *conxunto de adestramento*, que adoita estar formado por entre un 75 % e un 80 % dos datos orixinais, é o conxunto de datos sobre o cal se adestra o modelo empregando un método de aprendizaxe supervisada, Máis informalmente é o conxunto de datos que se emprega para que o algoritmo aprenda. É dicir, o conxunto de adestramento é o que se emprega para axustar o modelo.

Na práctica este conxunto de datos adoita estar formado por pares onde un dos elementos é a entrada e, o outro, a correspondente "resposta" que adoita recibir o nome de *etiqueta*. Volvendo ao problema de clasificación de casas, a entrada serían as variables que se teñen e a etiqueta sería unha variable dicotómica que nos indicaría que ocorreu coa casa: se foi vendida ou non.

Como xa dixemos, o modelo adéstrase con este conxunto producindo un resultado que se comparará coa etiqueta para ver como de ben ou mal predixo o noso algoritmo e, en función disto, actualizaranse os pesos do modelo ata que sexa axustado. Afondaremos nisto con máis detalle un pouco máis adiante.

Conxunto de validación

Xeralmente o termo de *conxunto de validación* emprégase indistintamente xunto co termo *conxunto de test* e refírese ao conxunto de datos formado polo 25 % ou 20 % dos datos que non foi empregado no conxunto de adestramento.

O conxunto de validación emprégase, como se pode ler en Gareth et al (2013) [8], para obter unha estimación da taxa de erro que está a producir o noso modelo. Máis informalmente, o conxunto de validación emprégase para ver como de ben ou como de mal está a predicir o noso modelo.

Para isto, unha vez que o modelo foi adestrado e axustado sobre o conxunto de adestramento, este é empregado para realizar predicións. Ditas predicións teñen que ser realizadas sobre un conxunto de datos que non foi empregado para adestrar o modelo nin que foi visto polo mesmo, como pode consultarse en Kuhn et al (2013) [24].

Así pois, as predicións realizaranse sobre o conxunto de validación que xoga polo tanto o papel do conxunto de datos non vistos polo modelo. Unha vez obtidas ditas predicións estas son comparadas coas etiquetas obtendo deste xeito unha medida do erro que se está a cometer[17].

Se ben estas son as definicións nas que nos basearemos ao longo deste traballo cómpre mencionar que algúns autores, como Russel et al (2009) [37], propoñen a división do noso conxunto de datos en tres conxuntos en vez de dous. Así, estes autores propoñen dividir o conxunto de datos que se emprega para axustar o modelo, que ata o de agora recibía o nome de conxunto de adestramento, en dous conxuntos: o conxunto de adestramento e o conxunto de validación. Do mesmo xeito, o que antes recibía o nome de conxunto de validación pasaría a chamarse conxunto de test. Vexamos a función que ten agora cada un dos distintos conxuntos:

- O conxunto de adestramento: o conxunto de adestramento segue a empregarse para realizar o axuste dos parámetros do noso modelo.
- O conxunto de validación: este conxunto recibe o mesmo nome que o conxunto que se empregaba para avaliar o modelo. Sen embargo agora, o conxunto de validación ten outro papel, empregándose nunha etapa intermedia do modelo e non na final. Así pois, o conxunto de validación emprégase agora para, unha vez axustada a rede sobre o conxunto de adestramento, realizar o *fine-tuning* do modelo. O *fine-tuning* pode ser, entre outras cousas, a elección do número de capas ocultas na rede neuronal, se ben veremos isto con máis detemento na sección dedicada aos hiperparámetros (ver Sección 2.3.3).
- O conxunto de test: este conxunto pasaría a xogar o papel que vimos de explicar anteriormente no conxunto de validación.

2.3.2. Funcións de activación

Como xa dixemos ao introducir o perceptrón (ver Sección 2.2.3), as funcións encargadas de modelar o comportamento das redes neuronais reciben o nome de funcións de activación. Do mesmo xeito, a saída da función de activación de cada un dos nodos da nosa rede eran os valores que lle pasábamos á seguinte capa da rede neuronal e recibían o nome de activación.

Se volvemos á inspiración biolóxica, a función de activación pódese pensar como unha abstracción representando unha taxa de potencial de activación que activa, ou non, á neurona segundo o valor obtido, estando polo tanto a xogar o papel do axón na neurona biolóxica.

As funcións de activación atópanse no corazón de cada rede neuronal, onde xace unha transformación lineal seguida dunha función de activación $f(\cdot) : \mathbb{R} \longrightarrow \mathbb{R}$. Esta función, dada a entrada ou conxunto de entradas , define a saída dun nodo permitindo propagar a saída da capa actual cara a seguinte capa ata, finalmente, chegar á capa de saída. Ademais, nas capas ocultas, estas funcións permítennos introducir a capacidade para modelar hipóteses non lineais e moitas delas permítennos transformar os nosos datos para que todos eles estean definidos sobre un mesmo rango, sendo os máis comúns o $[0, 1]$ e $[-1, 1]$.

Moitas destas funcións pertencen a unha clase loxística que recibe a nome de *sigmoidal*, graficamente estas funcións caracterízanse por ter unha forma moi semellante a unha "S". A familia sigmoidal contén varias variacións sendo a máis común a función sigmoide.

Despois desta pequena introdución vexamos algunha das funcións de activación máis empregadas en redes neuronais.

[17] Máis detalles sobre a forma de medir estes erros veranse máis adiante na Sección 2.3.2.

Lineal

Esta función de activación é a máis sinxela pois non é outra cousa que unha transformación lineal. Polo tanto, trátase dunha función da forma $f(x) = Ax$ onde a función dependente é directamente proporcional á variable resposta. Podemos pensala como a función identidade, é dicir, estaríamos a pasar o sinal á seguinte capa de nodos sen que este sufrira ningunha modificación.

Notemos que se ben esta función non xoga un papel moi importante á hora de realizar unha transformación dos nosos datos, incluímola aquí xa que adoita empregarse nos nodos de entrada posto que, se lembramos, as entradas non son modificadas ata entrar nas capas ocultas ou, de non habelas, na capa de saída.

Sigmoide

A función sigmoide, como xa vimos, vén dada por:

$$f(z) = \frac{1}{1 + e^{-z}},$$

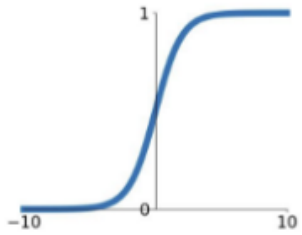

Figura 2.11: Función sigmoide.

e, como todas as funcións loxísticas, permite reducir os posibles valores extremos ou atípicos dos nosos datos sen ter que eliminalos. Esta función, como xa adiantábamos un pouco máis arriba, leva calquera elemento do intervalo $(-\infty, \infty)$ no intervalo $(0, 1)$ como podemos ver na Figura 2.11.

Historicamente, a función sigmoide foi moi popular xa que imita o comportamento do cerebro relativo á activación das neuronas biolóxicas ademais de ser especialmente vantaxosa cando se usa en redes neuronais adestradas co algoritmo de propagación cara atrás, como se establece en Karlic e Olgac (2011) [20]. Sen embargo, nos últimos anos a súa popularidade esta diminuíndo en gran medida debido a problemas relacionados co esvaecemento do gradiente debido ás dúas asíntotas horizontais da nosa función ($y = 0$ e $y = 1$).

Ademais, a función sigmoide non está centrada no cero, para velo non fai falta máis que ver que $f(0) = 0.5$, se ben este problema non é tan grave como o esvaecemento do gradiente que vimos de comentar.

Softmax

A función *softmax* definida por

$$f(z)_j = \frac{e^{z_j}}{\sum_{k=1}^{K} e^{z_k}},\ j = 1, ..., K,$$

é unha xeneralización da función de regresión loxística no senso de que pode ser aplicada a variables continuas, en vez de a clasificadores binarias, e pode conter múltiples rexións de decisión. Esta función de activación adoita atoparse na capa de saída das redes neuronais que actúan como clasificadores, sendo especialmente útil nos problemas de clasificación multiclase.

Así, nos problemas de clasificación multiclase a capa de saída terá tantos nodos como clases esteamos considerando pois, en cada un destes nodos, calcularase a probabilidade de que a entrada pertenza á devandita clase. Para isto, aplicariámoslle a cada entrada da última capa esta función obtendo a distribución da probabilidade sobre todas as clases.

Tanxente hiperbólica

A *tanxente hiperbólica* definida como

$$f(z) = \frac{e^z - e^{-z}}{e^z + e^{-z}},$$

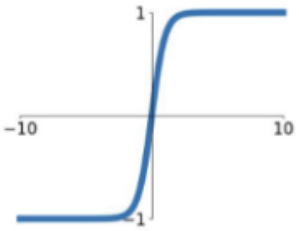

Figura 2.12: Función tanxente hiperbólica.

aparece representada na Figura 2.12. Esta función leva calquera número real ao intervalo $(-1, 1)$ e non é outra cousa que un reescalado da función sigmoide[18]. Se nos fixamos na gráfica, é sinxelo darse conta que segue a ter o problema de esvaecemento do gradiente que presentaba a sigmoide pero a tanxente hiperbólica, a diferenza da sigmoide, si que está centrada no cero o que a fai preferible á función sigmoide pois isto axuda a acelerar a converxencia.

Rectificador lineal

O *rectificador lineal*, máis coñecido como ReLU, é unha función de activación definida como

$$f(z) = \text{máx}(0, z).$$

Esta función actúa como a identidade para cada valor positivo e devolve un cero para calquera valor nulo ou negativo, é dicir, leva calquera valor real ao intervalo $[0, \infty)$ (ver Figura 2.13).

Algunha das vantaxes que presenta a ReLU son a súa rápida converxencia. Por exemplo, o método do descenso do gradiente estocástico converxe moito máis rápido con ReLU que coa función sigmoide ou a tanxente hiperbólica. Destaca tamén a súa sinxela implementación xunto con que non presenta esvaecemento do gradiente como si acontecía coas outras dúas que vimos de mencionar no punto anterior, pode consultarse máis literatura en Maas et al (2014) [27].

[18] Sexa g a función sigmoide e f a tanxente hiperbólica, entón: $f(z) = 2 \cdot g(2z) - 1$.

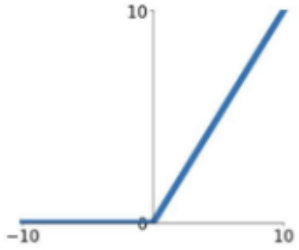

Figura 2.13: Función ReLU.

Sen embargo, tamén presenta as súas desvantaxes como pode ser a súa fraxilidade. Así o nodo pode chegar a "morrer, referíndonos por "morrer" a que ese nodo quedará inactivo de xeito irreversible cando hai grandes cambios no gradiente, máis literatura pode consultarse en Lil et al (2017) [25]. Pese a isto, a función ReLU tense erixido nos últimos anos como unha das funcións de activación máis populares por, entre outras cousas, permitir o adestramento das redes neuronais sen requirir un pre-adestramento supervisado, máis literatura pode consultarse en LeCun et al (2015) [26].

Leaky ReLU

A función de activación *leaky ReLU* creouse coa intención de arranxar o problema de morte de ReLU. Así, en vez de que a función tome o valor cero para valores nulos ou negativos, a función leaky ReLU presenta unha pequena pendente negativa (ver Figura 2.14). Esta función defínese como

$$f(z) = \begin{cases} \alpha x & z < 0 \\ x & z \geq 0 \end{cases}.$$

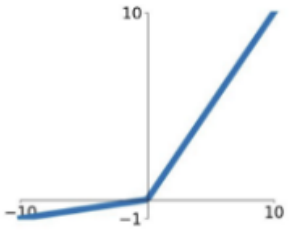

Figura 2.14: Función Leaky ReLU.

onde a pendente da recta, α, é unha constante[19] pequena (0.01, por exemplo). Esta pendente tamén pode ser parametrizada en cada nodo como propuxeron He et al (2015) [14], neste caso a función recibe o nome de *ReLU paramétrica*.

[19]Notemos que para $\alpha = 1$ teríamos a función ReLU.

Maxout

MaxOut é unha das técnicas máis avanzadas. A súa principal característica é que consiste en aplicar unha non linearidade ao produto escalar entre os pesos e os datos. Unha das eleccións máis comúns para esta función de activación é a *neurona MaxOut* a cal foi proposta por Goodfellow et al (2013) [13], esta calcula a función

$$f(z) = \text{máx}(a_1^T z + c_1, a_2^T z + c_2).$$

onde $a_i, c_i, i = 1, 2$ son parámetros que poden ser aprendidos como acontecía coa pendente en leaky ReLU. Non é difícil ver que esta función xeneraliza tanto a función ReLU como á leaky ReLU polo que beneficiase das súas vantaxes, salvo polo número de parámetros que aquí dóbrase.

Avances e conclusións

Incluímos ata o de aquí as funcións de activación máis coñecidas polo de agora se ben o campo das funcións de activacións está sendo amplamente investigado e novas funcións de activacións están sendo propostas entre as que destaca a Swish, sobre a cal se pode ler en Ramachandram et al (2017) [**?**].

Tras ver algunhas das diferentes opcións que poden ser empregadas como funcións de activación, fagamos un pequeno inciso para facernos á idea de como se pode escoller cal empregar. Con isto en mente limitémonos ás funcións lineal, sigmoide e softmax e a dous dos problemas máis clásicos da estatística: regresión e clasificación, considerando tanto a binaria como a clasificación multiclase.

- Regresión: se estamos ante unha capa de saída para un problema de regresión teremos que ter en conta o tipo de saída que queremos para o noso modelo para, así, poder escoller unha función de activación que se axuste ás nosas necesidades. Neste caso, por tratarse dunha regresión, esperaríamos que a saída fose un único valor real para cada entrada, polo tanto, a función de activación lineal sería unha boa opción.
- Clasificación binaria: nos problemas de clasificación esperamos obter como saída un número real no intervalo $(0, 1)$, o cal se interpreta como probabilidade de pertenza a un dos dous grupos. Unha boa opción para este tipo de problemas sería usar como capa de saída un único nodo coa función sigmoide como función de activación para obter estas probabilidades.
- Clasificación multiclase: no caso de ter un problema de multi-clasificación onde as clases son mutuamente exclusivas o obxectivo é basicamente o mesmo que o comentado no problema de clasificación binaria, obter as probabilidades de pertenza a cada unha das clases. Sen embargo, se agora empregásemos a función sigmoide non obteríamos probabilidades[20], unha solución amplamente recorrida pasa por empregar a función softmax.

Como consellos máis xerais recoméndase probar sempre coa función ReLU tendo especial coidado coas taxas de aprendizaxe e monitoreando a fracción de nodos mortos da nosa rede. Outra posibilidade sería empregar calquera das súas variantes ou a función MaxOut para intentar evitar o problema dos nodos mortos. Ademais, recoméndase non empregar a función sigmoide en favor da tanxente hiperbólica se ben con esta cabe esperar peores resultados que coa ReLU.

Dependendo da función obxectivo escollida veremos que diferentes funcións de activación serán máis ou menos apropiadas en función do tipo de datos cos que esteamos a traballar. Estas decisións do deseño da arquitectura da rede divídense en dúas áreas principais: funcións de activación para capas ocultas e para capas de saída.

[20] Isto é obvio pois se sumásemos a saída de cada clase o total sumaría 1.

As capas ocultas están fortemente relacionadas coa extracción progresiva de características de orde superior a partir dos datos de entrada. Dependendo da arquitectura coa que esteamos traballando adoitamos usar só certos subconxuntos das funcións de activación, como pode consultarse en Patterson et al (2017) [30].

En canto á capa de entrada xa vimos que non se emprega realmente ningunha función de activación xa que o que nos interesa é pasar os datos orixinais sen realizar ningunha transformación sobre eles.

2.3.3. Funcións de perda

As *funcións de perda* cuantifican como de próxima está a rede neuronal de acadar o obxectivo para o que está a ser adestrada. Para isto o que se fai é calcular unha métrica baseada no erro observado nas predicións da rede e, unha vez obtidos todos os posibles erros, calcúlase a súa media de forma que se obten un valor real representativo de como de cerca estivo a rede neuronal de alcanzar dito obxectivo.

Supoñamos o problema de predicir se unha determinada casa será vendida nos seis próximos meses. Como xa sabemos, para isto precisaríamos un conxunto de datos formado polas variables de cada unha das casas xunto cunha etiqueta que nos dixese se a casa foi vendida neses seis meses ou non. Unha vez que a rede neuronal realizou as predicións para cada unha das casas, a función de perda podería calcular como de ben predixo a nosa rede neuronal comparando a saída obtida coa etiqueta correspondente.

Estas funcións de perda divídense en dous grandes grupos segundo o tipo de problema co que esteamos a tratar, destacando principalmente dous tipos: os problemas de regresión e os problemas de clasificación. Ao longo desta sección centrarémonos nestas últimas pois o problema que imos tratar, a detección e clasificación de obxectos en imaxes, é un problema de clasificación.

A continuación introduciremos polo tanto as funcións de perda máis empregadas para os problemas de clasificación así como a lóxica detrás das mesmas. Antes de proceder a describir algunhas das diferentes funcións que nos podamos atopar introduzamos a notación que empregaremos ao longo desta sección:

- Denotaremos por n o tamaño mostral do noso conxunto de datos.
- Tendo en conta a natureza do conxunto de datos denotaremos por p o número de variables de entrada e por m o número de saídas observadas. Así, no exemplo das casas onde tiñamos 100 variables, $p = 100$, e os datos de saída sería a observación sobre si a casa se vendeu ou non, $m = 1$.
- O conxunto de datos será denotado polo par

  $$(\boldsymbol{X}, \boldsymbol{Y}) = \left\{ \left(x^{(1)}, y^{(1)} \right), ..., \left(x^{(n)}, y^{(n)} \right) \right\}$$

  onde o i-ésimo par do conxunto de datos denotarase por $\left(x^{(i)}, y^{(i)} \right)$. Notemos ademais que, polo dito no punto anterior, haberá n pares onde a entrada $x^{(i)}$ é unha colección de p valores e a saída $y^{(i)}$ é unha colección de m valores. Por comodidade supoñeremos de aquí en diante que tanto os $x^{(i)}$ como os $y^{(i)}$ son vectores, é dicir, $p \geq 2$ e $m \geq 2$.
- A saída da rede neuronal denotarase por $\hat{\boldsymbol{Y}}$, a cal notemos coincide coa activación calculada na última capa. Notemos tamén que, por cada $\boldsymbol{y}^{(i)}$ do noso conxunto de datos, a rede neuronal fará unha predición polo que teremos un total de m predicións por cada entrada.
- Igual que antes teremos que

  $$h_{\mathbf{W}} \left(\boldsymbol{x}^{(i)} \right) = \hat{\boldsymbol{y}}^{(i)}$$

  o cal denota as transformacións que sofre a entrada $\boldsymbol{x}^{(i)}$ para obter a saída $\hat{\boldsymbol{y}}^{(i)}$.

- Cando nos refiramos á j-ésima característica das p ou m existentes denotarémolo como $x_j^{(i)}$ no caso de datos de entrada e como $y_j^{(i)}$ no caso das etiquetas. Así, por exemplo, $y_2^{(1)}$ refírese á segunda característica de saída observada na primeira mostra recollida $\left(x^{(1)}, y^{(1)}\right)$.

- Denotaremos a función de perda como $L(\mathbf{W}, b)$, indicando deste xeito que o valor da función de perda depende tanto da matriz de pesos ,$\mathbf{W}$, como das unidades de sesgo, b. É dicir, o valor da función de perda depende de xeito exclusivo do estado no que se atopa a nosa rede neuronal, o cal vén definido tanto polos pesos como polas unidades de sesgo.

Unha vez introducida a notación que imos a empregar vexamos as distintas funcións de perdas que se poden empregar cando temos problemas de clasificación.

Funcións de perda para problemas de clasificación

Xa vimos en que consisten os problemas de clasificación co exemplo que vimos empregando todo este tempo no cal tiñamos un conxunto de datos de casas e queríamos predicir se ían ser vendidas ou non.

Se ben é certo que podemos crear redes neuronais que nos permitan realizar esta clasificación en clases, normalmente, á hora de crear estas redes a atención céntrase en asociar probabilidades a estas clasificacións. Así, dada unha entrada do noso conxunto de datos o que fai a rede neuronal é asociarlle probabilidades á clasificación. Por exemplo, podería asociarlle unha probabilidade do 18 % de que a casa non sexa vendida fronte a un 82 % de que si o sexa, deste xeito -por ter unha probabilidade superior ao 50 %- clasificaríamos a casa no grupo de vivendas que si serán vendidas nos próximos seis meses.

Estes diferentes escenarios que vimos de comentar fan que se precisen diferentes funcións de perda que imos ver a continuación.

Hinge loss

En primeiro lugar, destaca a función de perda *hinge loss*, que en español significa perda de bisagra. Esta función é unha das funcións de perda máis empregadas cando a rede debe ser optimizada para realizar unha clasificación directa en clases sen importar tanto a probabilidade. Por exemplo, $0 =$ "non se vendeu a casa" e $1 =$"vendeuse a casa", este tipo de clasificadores reciben por convención o nome de clasificadores $0-1$. A elección de $0, 1$ é unha elección completamente arbitraria podendo empregar calquera outros dous números, por interese consideremos o -1 e 1.

Deste xeito, cando os datos deben ser categorizados nas clases -1 e 1, a función de perda represéntase por:

$$L\left(\mathbf{W}, b\right) = \frac{1}{n}\sum_{i=1}^{n} \text{máx}\left(0, 1 - y_j^{(i)} \cdot y_j^{(i)}\right).$$

A *hinge loss* é maioritariamente empregada cando se trata de problemas de clasificación binaria aínda que tamén existen extensións para problemas de clasificación multiclase se ben non entraremos nestes agora.

Función de perda loxística

A función de perda loxística, pola contra á *hinge loss*, emprégase cando as probabilidades de pertenza a cada unha das diferentes clases son máis interesantes que a pertenza á clase en si. A predición de probabilidades válidas significa xerar números entre 0 e 1 asegurándose de que as probabilidades de pertenza a clases mutuamente exclusivas sumen 1. É por esta e non outra razón o porqué a función

de activación da última capa dunha rede neuronal empregada para realizar clasificación debe de ser a función softmax[21].

Unha vez explicado como deben ser as saídas da nosa rede neuronal para que sexan probabilidades válidas afondemos algo máis na función de perda loxística. O que queremos optimizar é a máxima verosimilitude, isto é, queremos maximizar a probabilidade predita para a clase correcta e queremos facelo para cada unha das mostras que temos.

Para entender mellor esta función de perda supoñamos, sen perda de xeneralidade, que queremos predicir a probabilidade de dúas clases como no caso do clasificador $0-1$. Deste xeito, dado un conxunto de pesos ($\mathbf{W}$), unidades de sesgo ($\boldsymbol{b}$) e unha entrada ($\boldsymbol{x}^{(i)}$) podemos expresar a saída como $h_{\mathbf{W}}\left(\boldsymbol{x}^{(i)}\right)$ e $1-h_{\mathbf{W}}\left(\boldsymbol{x}^{(i)}\right)$ expresando a probabilidade de 1 como

$$\mathbb{P}\left(y^{(i)}=1|\boldsymbol{x}^{(i)};\mathbf{W},\boldsymbol{b}\right)=h_{\mathbf{W}}\left(\boldsymbol{x}^{(i)}\right)$$

e expresando a probabilidade de 0 como

$$\mathbb{P}\left(y^{(i)}=0|\boldsymbol{x}^{(i)};\mathbf{W},\boldsymbol{b}\right)=1-h_{\mathbf{W}}\left(\boldsymbol{x}^{(i)}\right).$$

Estas dúas ecuacións podemos expresalas de forma combinada como

$$\mathbb{P}\left(y^{(i)}|\boldsymbol{x}^{(i)};\mathbf{W},\boldsymbol{b}\right)=\left(h_{\mathbf{W}}\left(\boldsymbol{x}^{(i)}\right)\right)^{y^{(i)}}\cdot\left(1-h_{\mathbf{W}}\left(\boldsymbol{x}^{(i)}\right)\right)^{1-y^{(i)}}.$$

Definir a función de perda resulta inmediato se lembramos que queremos maximizar a probabilidade predita para a clase correcta para cada unha das mostras das que dispoñemos, deste xeito definimos a función de perda como

$$L(\mathbf{W},b)=\prod_{i=1}^{n}\hat{y}^{(i)y^{(i)}}\cdot\left(1-\hat{y}^{(i)}\right)^{1-y^{(i)}}.$$

Negative log likelihood

Cando estamos a tratar con produtos de probabilidades adoitamos transformalos en logaritmos de probabilidades por conveniencia. Ademais, o logaritmo é unha función monótona crecente polo que maximizar a probabilidade, que é o que queremos, equivale a minimizar o logaritmo da máxima verosimilitude.

Polo tanto, aplicando as transformacións que vimos de comentar á función de perda loxística, teríamos a seguinte función de perda:

$$L\left(\mathbf{W},b\right)=-\sum_{i=1}^{n}y^{(i)}\times\log\left(\hat{y}^{(i)}\right)+\left(1-y^{(i)}\right)\times\log\left(1-\hat{y}^{(i)}\right)$$

Se quixésemos estender esta función de perda ao caso onde temos m clases obteríamos a seguinte función de perda:

$$L(\mathbf{W},b)=-\sum_{i=1}^{n}\sum_{j=1}^{m}y_j^{(i)}\times\log\left(\hat{y}_j^{(i)}\right).$$

Así, nesta sección explicamos as funcións de perda que cuantifican o erro cometido polas redes neuronais en problemas de clasificación. Ademais, o obxectivo para o que está a ser adestrada a rede

[21] Notemos que a función sigmoide tamén ten como saída valores válidos entre 0 e 1 pero esta non pode ser empregada cando hai máis de dúas clases mutuamente exclusivas posto que non modela as dependencias entre os valores de saída.

neuronal acadarase cando o erro cuantificado polas redes neuronais sexa nulo ou mínimo no conxunto de validación[22]. Intentar reducir o erro cuantificado polas funcións de perda non vén ser outra cousa que atopar os parámetros, pesos e unidades de sesgo, que minimizan a perda na que se incorre cos erros, para isto emprégase o algoritmo de propagación cara atrás xunto cun algoritmo de optimización como veremos a continuación (ver Sección 2.3.4).

2.3.4. Algoritmo de propagación cara atrás

Vimos que as funcións de perda calculan o erro na predición da rede neuronal comparando para iso o valor obtido como saída da rede neuronal co valor que se esperaría obter. Para que a rede neuronal fose mellor o ideal sería que este erro se reducise e para isto, bastaría con atopar os parámetros, pesos e unidades de sesgo, que minimizasen a función de perda. É para isto para o que se adestra a rede neuronal.

Durante moitos anos os investigadores esforzáronse por atopar un método para realizar este adestramento e non foi ata 1974, ver Webos (1974) [42], que se obtivo finalmente a resposta co algoritmo de *propagación cara atrás*, o cal se basea nunha vella regra do cálculo: a regra da cadea. Se facemos memoria, cando introducimos o perceptrón multicapa, explicamos o funcionamento do mesmo, pero non explicamos como se realiza o adestramento; de feito, vimos que o algoritmo proposto por Rosenblatt non servía para adestrar este tipo de perceptrón multicapa debido ás capas ocultas. Así pois, agora o noso algoritmo necesita decidir baixo que circunstancias as capas ocultas deberían ser activadas para que se reduza este erro, pode consultarse máis literatura en Rumelhart et al (1986) [36].

O adestramento é un proceso de aprendizaxe no que o algoritmo reaxusta os pesos e as unidades de sesgo facéndoas máis grandes ou pequenas para intentar minimizar a función de perda. Estes reaxustes axudan ao noso modelo a aprender que nodos predín que cousas e axustar os pesos e as unidades de sesgo de acordo a isto. Deste xeito, os pesos poden ser entendidos matematicamente como un vector de parámetros empregado para optimizar a función de perda ou, o que vén a ser o mesmo, minimizar o erro. O algoritmo máis coñecido e amplamente empregado nas redes neuronais é o algoritmo de propagación cara atrás, no cal nos centraremos a continuación.

Este algoritmo de propagación cara atrás emprégase para realizar o cálculo do gradiente da función de perda. O gradiente é logo empregado para actualizar os pesos de acordo a algún algoritmo de optimización cuxa función é minimizar o valor da función de perda. Entre estes algoritmos de optimización o máis empregado é o descenso de gradiente estocástico (SGD), como pode consultarse en LeCun et al (2015) [26].

Antes de proceder a explicar o algoritmo de propagación cara atrás de xeito formal vexamos de forma intuitiva que é o que fai o mesmo. Así, en primeiro lugar, realiza unha primeira predición para cada instancia de adestramento, a cal non vén a ser outra cousa que a saída da rede neuronal. En segundo lugar, mídese o erro cometido empregando unha función de perda. E, finalmente, retrocede por cada unha das capas, é dicir, vai avanzando dende a capa de saída cara atrás ata chegar de novo á capa de entrada da rede neuronal, neste proceso vaise medindo a contribución de cada unha das conexións ao erro para logo, coa información obtida, calcular o gradiente e actualizar os pesos.

Deste xeito, a rede neuronal, grazas ao algoritmo de propagación cara atrás, pode ver canto cambia o erro cando se modifica un dos seus pesos, incluíndo aqueles das capas ocultas. Empregando esta información xunto cun algoritmo de optimización a rede neuronal modifica lixeiramente os seus pesos para intentar minimizar a función de perda.

Así pois, as ideas claves do algoritmo de propagación cara atrás son tres:

[22]Notemos que se durante a fase de adestramento o erro fose nulo ou moi pequeno, estaríamos ante un problema de sobreaxuste.

- Cuantificar como de boas ou malas son as predicións realizadas pola rede neuronal baseándose na función de perda.

- Calcular os gradientes para cada parámetro na rede aplicando as regras de cálculo multivariante tanto á función de perda como á estrutura da rede neuronal.

- Empregar os gradientes calculados para, de forma iterativa, axustar os parámetros da rede na dirección que minimiza a función de perda.

Agora que xa temos a intuición vexamos máis en profundidade o funcionamento deste algoritmo. Como vimos de dicir, o obxectivo é minimizar a función de perda, á cal denotaremos por J, suxeita á matriz de pesos $\mathbf{W}$:

$$\min_{\mathbf{W}} J(\mathbf{W}).$$

Polo tanto, o obxectivo é atopar os parámetros óptimos que minimicen esta función. Para isto basearémonos como xa dixemos na regra da cadea facendo uso das derivadas parciais:

Para minimizar a función empregamos o algoritmo de propagación cara atrás que veremos a continuación se ben antes introducimos algo de notación. Así:

- $\delta(k)$ representa os erros da capa k da nosa rede neuronal. Polo tanto, o elemento $\delta_{ij}^{(k)}$ representa o erro do nodo j na capa k. Lembremos tamén que $a_j^{(k)}$ era a activación do nodo j na capa k polo que, en certo xeito, o termo $\delta_j^{(k)}$ está a capturar o erro que se produce na activación $a_j^{(k)}$.

- $\Delta^{(k)}$ é unha matriz acumuladora. É dicir, en cada entrada $\Delta_{ij}^{(l)}$ da matriz vanse acumulando as derivadas parciais de $J(\mathbf{W})$ con respecto a $\mathbf{W_{ij}^{(k)}}$ como veremos no Algoritmo 1. Ao final do algoritmo estes acumuladores serán empregados para o cálculo da derivada parcial

$$\frac{\partial}{\partial \mathbf{W_{ij}^{(k)}}} J(\mathbf{W}),$$

como pode consultarse en Rumelhart et al (1986) [36]. Ademais pode demostrarse, se ben a demostración é moi laboriosa e omítese aquí, que de ignorar o termo de regularización λ ou se $\lambda = 0$,

$$\frac{\partial}{\partial \mathbf{W_{ij}^{(k)}}} J(\mathbf{W}) = a_j^{(k)} \cdot \delta_i^{(k+1)}.$$

Algoritmo 1 Propagación cara atrás

Dado un conxunto de adestramento

$$\left\{\left(x^{(1)}, y^{(1)}\right), ..., \left(x^{(n)}, y^{(n)}\right)\right\}$$

O algoritmo de propagación consiste en:

1: Fixar $\Delta_{ij}^{(k)} = 0, \forall k, i, j$.

2: Para $i = 1, ..., n$

- Fixar $\boldsymbol{a}^{(1)} = \boldsymbol{x}^{(i)}$.
- Realizar propagación cara adiante para calcular a activación

  $$\boldsymbol{a}^{(k)}, k = 2, 3, ..., K$$

  onde K é o número total de capas da nosa rede.
- Empregando $\boldsymbol{y}^{(i)}$ calcular $\boldsymbol{\delta}^{(K)} = \boldsymbol{a}^{(k)} - \boldsymbol{y}^{(i)}$
- Calcular os erros das capas anteriores, para iso:

  $$\boldsymbol{\delta}^{(k-1)} = \left(\mathbf{W}^{(k-1)T} \cdot \boldsymbol{\delta}^{(k)}\right) . * g'\left(\mathbf{z}^{(k-1)}\right)$$

  onde $.*$ denota a multiplicación elemento a elemento.
- Actualización do valor de $\Delta_{ij}^{(k)}$:

  $$\Delta_{ij}^{(k)} = \Delta_{ij}^{(k)} + a_j^{(k)} \delta_i^{(k+1)},$$

  o cal se pode vectorizar como segue:

  $$\Delta^{(k)} = \Delta^{(k)} + \boldsymbol{\delta}^{(k+1)} \cdot \boldsymbol{a}^{(k)T}.$$

3: Cálculo de

$$D_{ij}^{(k)} = \begin{cases} \frac{1}{m}\Delta_{ij}^{(k)} + \lambda \mathbf{W}_{ij}^{(k)} & j \neq 0 \\ \frac{1}{m}\Delta_{ij}^{(k)} & j = 0 \end{cases},$$

onde λ é o termo de regularización.

Tras a introdución da notación expliquemos agora máis formalmente en que consiste o algoritmo de propagación cara atrás, o cal pode consultarse en Algoritmo 1.

Así, a nosa mostra de adestramento

$$\left\{\left(x^{(1)}, y^{(1)}\right), ..., \left(x^{(n)}, y^{(n)}\right)\right\}$$

está formada por pares onde os x_i con $i = 1, ..., n$ serían os datos de entrada da nosa rede neuronal e onde os y_i con $i = 1, ..., n$ son o valor real que a nosa rede está a intentar predicir.

Outro paso a detallar é o cálculo dos erros, así, o erro cometido na última capa,

$$\boldsymbol{\delta}^{(K)} = \boldsymbol{a}^{(k)} - \boldsymbol{Y}$$

é simplemente a diferenza entre o valor da activación obtida na capa de saída e os valores que se esperarían obter. Afondemos tamén un pouco no cálculo dos erros nas capas anteriores o cal vén dado por

$$\boldsymbol{\delta}^{(k-1)} = \left(\mathbf{W}^{(k-1)T} \cdot \boldsymbol{\delta}^{(k)}\right) .* g'\left(\boldsymbol{z}^{(k-1}\right)$$

onde a función g' é a derivada da función de activación g avaliada nos valores de entrada dados por $\mathbf{z}^{(k-1)}$, por simplicidade tomemos g como a función sigmoide, de xeito que

$$g'\left(\mathbf{z}^{(k-1)}\right) = \boldsymbol{a}^{(k-1)} .* \left(\mathbf{1} - \boldsymbol{a}^{(k-1)}\right).$$

É dicir, o vector $\boldsymbol{\delta}$ da capa $k-1$ calcúlase multiplicando o vector $\boldsymbol{\delta}$ da capa k coa matriz de pesos que van dende a capa $k-1$ ata a capa k. Unha vez feita esta operación este resultado multiplícase elemento a elemento coa función $g'\left(\mathbf{z}^{(k-1}\right)$.

Notemos ademais que se ben os $\boldsymbol{\delta}$ de capas anteriores si son calculados, nunca se calculará $\boldsymbol{\delta}^{(1)}$ pois a primeira capa correspóndese coa capa de entrada que, como dixemos, non soe ter función de activación e, de tela, soe ser a función lineal. Polo tanto a activación coincidiría co valor esperado non tendo ningún erro asociado.

Finalmente, se ben omitiremos a demostración por ser bastante laboriosa e afastarse dos obxectivos deste documento, pódese demostrar que

$$\frac{\partial}{\partial \mathbf{W}_{\mathbf{ij}}^{(\mathbf{k})}} J(\mathbf{W}) = D_{ij}^{(k)}.$$

Polo tanto, o algoritmo de propagación cara atrás proporciónanos o que queríamos: un método para o cálculo das derivadas parciais da función de perda.

Ademais, unha das grandes vantaxes de empregar o algoritmo de propagación cara atrás é que, na práctica, os mínimos locais rara vez supoñen un problema en redes neuronais grandes. De feito, independentemente das condicións iniciais, o sistema case sempre acada solucións de calidade semellante. Hai estudos empíricos e teóricos que suxiren que, como acabamos de dicir, os mínimos locais non constitúen un grave problema, máis literatura pode consultarse en LeCun et al (2015) [26].

Unha vez que temos un método para calcular as derivadas parciais da función de perda vexamos brevemente o algoritmo de optimización empregado para minimizar o valor da función de perda: o descenso de gradiente estocástico.

Descenso de gradiente estocástico

Por ir máis aló dos obxectivos deste traballo non afondaremos neste algoritmo, omitindo a explicación formal do mesmo, se ben si resulta convinte facer certos comentarios acerca del que nos permitan entender porqué se adoita empregar este algoritmo de optimización, e non outro, neste tipo de problemas.

Comecemos vendo a que se refire o termo estocástico, este débese a que no algoritmo de descenso de gradiente estocástico o cálculo do gradiente non é exacto, senón que se calcula empregando tan só unha mostra do noso conxunto de datos -ao contrario doutros algoritmos como o de descenso do gradiente no que se emprega todo o conxunto de datos- a cal é seleccionada de xeito aleatorio en cada caso, isto permite unha converxencia máis rápida.

Ademais, o descenso de gradiente estocástico é un algoritmo de optimización que destaca por poder realizar o adestramento en ordes de magnitude superiores a outros métodos, como pode ser o descenso do gradiente, sen que isto resulte nunha perda da exactitude do modelo. Outras das vantaxes deste

método de optimización son a súa sinxela implementación xunto co feito de que é un modelo robusto, o que axuda a construír modelos que xeneralicen ben.

Se lembramos, o gradiente en cada punto do espazo é un vector que sinala na dirección na cal a función de perda medra máis abruptamente. En cada unha das iteracións, o gradiente de descenso estocástico o que fai é dar un paso na dirección contraria ao gradiente, reducindo deste xeito a función de perda.

O emprego do gradiente de descenso estocástico é útil posto que, cando a función de perda é moi irregular, a aleatoriedade deste algoritmo pode axudarnos a saír de posibles mínimos locais, sendo máis probable que se chegue a acadar o óptimo global. Isto é bo por unha parte e mala por outra pois, debido á súa natureza estocástica, este algoritmo é moito menos regular que outros. Así, a función de perda en vez de ir decrecendo lentamente ata acadar un mínimo irá crecendo e decrecendo, reducíndose só en media. Co tempo, chegará a estar moi próxima do mínimo se ben unha vez aí non chegará a asentarse polo que, unha vez que o algoritmo pare, os valores finais serán suficientemente bos pero non óptimos.

Unha forma de evitar isto é ir reducindo de xeito gradual a taxa de aprendizaxe, de forma que o paso sexa grande ao principio, axudándonos a axilizar o proceso e escapar dos mínimos locais, e se vaia reducindo, a medida que o algoritmo vai converxendo, para que poida establecerse no mínimo global. A función que determina a taxa de aprendizaxe en cada unha das iteracións recibe o nome de *learning schedule*. Isto explícase con máis detalle na seguinte sección.

2.3.5. Hiperparámetros

Vimos de ver que para mellorar os resultados que se obteñen coa rede neuronal podemos modificar os pesos da mesma. Sen embargo, estes non son os únicos parámetros que podemos axustar para obter mellores resultados dada a flexibilidade das redes neuronais, os parámetros da rede neuronal que poden ser refinados reciben o nome de hiperparámetros.

Estes hiperparámetros caen en diferentes categorías entre as que destacan:

- Tamaño das capas
- Magnitude (*momentum* , taxa de aprendizaxe...)
- Regularización
- Funcións de activación
- Funcións de perda
- Época e tamaño do *minibatch*

Deste xeito, non só podemos empregar calquera topoloxía[23] imaxinable para as redes neuronais senón que tamén podemos trocar o número de capas, o número de nodos por capa, o tipo de función de activación empregado en cada unha das capas, etc.

Os hiperparámetros son os encargados de controlar tanto as funcións de optimización como a selección do modelo durante o adestramento, sendo un dos seus principais obxectivos o asegurarse de que o modelo non sobreaxusta nin infraaxusta o conxunto de adestramento, mentres que aprende a estrutura subxacente nos datos tan rápido como sexa posible. Vexamos logo algúns destes hiperparámetros.

[23] Entendendo por topoloxía a forma en que os nodos están interconectados.

Tamaño das capas

O tamaño da capa é definido polo número de neuronas que hai na mesma. As capas de entrada e saída son relativamente sinxelas pois correspóndense directamente a como os nosos modelos manexan os datos de entrada e saída. Así, o número de neuronas da capa de entrada coincidirá co número de variables explicativas que teña o noso conxunto de datos e a capa de saída estará formada por unha única neurona ou polo mesmo número de neuronas que clases esteamos a intentar predicir.

O reto comeza cando se intentan axustar o número de nodos das capas ocultas. Así pois, poderíamos escoller un número aleatorio de nodos para definir a capa, de feito, non hai regras sobre como de grande ou pequeno pode ser este número. Sen embargo, os modelos más difíciles de modelar adoitan precisar de un maior número de nodos nas capas ocultas o que podería levar a comezar cun gran número de neuronas dende o principio.

Tralo comentado faise evidente porqué o axuste dos nodos presentes en cada unha das capas da nosa rede neuronal xoga un papel important, se ben nós omitiremos os métodos empregados para realizar dito axuste que, normalmente, son modelos heurísticos.

Taxa de aprendizaxe

Afondemos agora un pouco máis na taxa de aprendizaxe. Concepto do cal xa falamos cando introducimos o algoritmo de adestramento do perceptrón e na sección dedicada ao algoritmo de propagación cara atrás. Así, como xa deixabamos entrever nesta última sección, este hiperparámetro afecta á cantidade na que axustamos os parámetros da nosa rede con respecto á función de perda, sendo o obxectivo final a minimización da mesma. Polo tanto, trátase dun coeficiente que escala o tamaño dos pasos que a rede neuronal da durante a actualización dos seus parámetros para optimizar a función de perda.

Durante a propagación cara atrás, para actualizar os pesos, multiplicamos o gradiente pola taxa de aprendizaxe e este valor réstaselle ao peso da iteración anterior para obter un novo peso. Vexámolo en forma de pseudocódigo para comprendelo mellor:

```
novo_peso = peso_existente - learning_rate*gradiente
```

Deste xeito a taxa de aprendizaxe determina canto influirá o gradiente na actualización dos parámetros na iteración do algoritmo na que esteamos. Canto menor sexa o valor da taxa de aprendizaxe máis lentamente nos moveremos e, pola contra, canto máis grande sexa o seu valor máis grandes serán os pasos que demos.

É por isto que antes comentábamos que unha boa estratexia sería fixar a taxa de aprendizaxe a un valor grande mentres o erro sexa grande para avanzar máis rápido e a medida que o erro se reduce, reducir tamén a taxa de aprendizaxe para facilitar a converxencia. Outra estratexia perfectamente válida sería, unha vez observemos que a nosa función de perda non mellore, cambiar a taxa de aprendizaxe en cada iteración de acordo a algunha función cíclica, como se pode ver en Hutter et al (2017) [16] onde propoñen empregar a función coseno.

Regularización

A regularización é unha técnica que realiza lixeiras modificacións no algoritmo de aprendizaxe, sendo o principal obxectivo o de reducir un posible sobreaxuste do modelo, permitindo que o mesmo xeneralice mellor obtendo mellores resultados sobre conxuntos de datos que nunca antes foran observados.

Os termos de regularizacón, os cales adoitan denotarse por λ, intentan atopar o equilibro entre un bo axuste e manter os pesos de certas características baixos, evitando así o sobreaxuste. A elección do valor

de λ é, polo tanto, fundamental porque un valor moi pequeno podería dar lugar a un infraaxuste e un valor moi grande podería non ter ningún efecto, conlevando ao sobreaxuste que tratamos de evitar. O ideal é atopar un punto intermedio que mediante a ponderación dos pesos dean lugar a hipóteses máis simples, as cales son máis sinxelas de xeneralizar xa que canto menores sexan os grados de liberdade máis complicado será sobreaxustar o modelo. Un dos exemplos máis comúns de regularización consiste en reducir o número de graos polinómicos.

A medida que o tamaño do conxunto de adestramento , o efecto da regularización diminúe e os parámetros tenden a crecer en magnitude, o cal é conveniente pois un exceso de variables relativas ao conxunto de adestramento adoita levar ao sobreaxuste. Ademais, entre estes métodos destacan a regresión Ridge e a regresión Lasso, se ben nos últimos anos a técnica do *autoencoder*, cuxa finalidade é axudar na redución da dimensionalidade, está a situarse como unha das principais técnicas en aprendizaxe profunda, como pode verse en LeCun et al (2015) [26].

Momentum

O *momentum* refírese a un conxunto de técnicas empregadas no ámbito da aprendizaxe profunda, deseñadas para acelerar a converxencia dos métodos de optimización. O termo do *momentum* axuda ao algoritmo de aprendizaxe a escapar de certos lugares do espazo de busca nos que podería quedar atrapado, axudando ao algoritmo a atopar os camiños que o levan cara o mínimo.

Esencialmente consiste en engadir á función obxectivo o que se chama termo do momentum, un coeficiente que toma valores entre 0 e 1, que permite aumentar o tamaño dos pasos que se toman para chegar ao mínimo, evitando os mínimos locais. Cando o termo de *momentum* é grande, o que acelera a converxencia, a taxa de aprendizaxe debería permanecer pequena, para evitar que unha vez atopado o mínimo global, saiamos del. Pola contra, un valor pequeno do termo do *momentum* pode reducir a velocidade de converxencia, ademais de non ser capaz de escapar dos mínimos locais. Deste xeito, o *momentum* é á taxa de aprendizaxe o que a taxa de aprendizaxe é aos pesos, e axuda a producir modelos de maior calidade.

2.3.6. Época e tamaño do *minibatch*

Nesta sección introducimos dous termos novos moi importantes á hora de realizar o adestramento das redes neuronais: as *épocas* e o tamaño do *minibatch*.

A época é o número total de veces que se lle pasa completamente o noso conxunto de datos á rede durante o adestramento.

O tamaño do minibatch define o número de datos que vai ser propagado a través da rede en cada iteración. Así, se temos un total de 1000 datos e fixamos o tamaño do *minibatch* en 100, o algoritmo empezaría a adestrar coas 100 primeiras mostras do conxunto de adestramento, o *mnibatch*. Seguidamente, tomaría as seguintes 100 mostras e adestraríase novamente a rede. E así sucesivamente ata que se tivesen empregado os 1000 datos dos que dispoñemos, para o que precisaríamos un total de 10 iteracións. Notemos que as 10 iteracións constitúen unha época, se tivésemos fixado o número de épocas en 3, este proceso que vimos de describir repetiríase 3 veces en total.

A vantaxe de empregar o *minibatch* é que fai que sexa precisa menos memoria, posto que se están a empregar un menor número de mostras cada vez que realiza o adestramento. Ademais, as redes adestran máis rápido cos *minibatch*, xa que os pesos son actualizados en cada unha das iteracións, mentres que se tivésemos usado todos os datos á vez, só teríamos actualizado os pesos unha vez.

Hai que ter especial coidado, pois os tamaños pequenos de *minibatch* poden facer que a estimación do gradiente sexa menos precisa. Normalmente adóitanse empregar tamaños que van dende 32 ata

512, dependendo das características do modelo que se estea a empregar. Tamén hai que ter especial coidado co número de épocas a empregar á hora de adestrar o modelo, xa que un número moi grande equivalería a adestrar empregando moitas veces o noso conxunto de datos, o que pode resultar nun sobreaxuste do modelo.

Con isto damos por concluída a introdución ás redes neuronais artificiais. Nos seguintes capítulos explicaremos a rede neuronal máis empregada para a análise de imaxes e discutiremos as distintas arquitecturas que podemos empregar neste tipo de redes, presentando finalmente unha aplicación práctica para as mesmas.

Capítulo 3

Aprendizaxe profunda para a análise de imaxes

Tras esta introdución aos fundamentos básicos das redes neuronais e aumentar os nosos coñecementos sobre as mesmas, imos indagar un pouco nas súas aplicacións. En concreto, imos centrarnos nas aplicacións das mesmas para a análise de imaxes, a cal, xunto co procesamento de texto, é un dos campos máis estudados dentro da aprendizaxe profunda. Isto non debería sorprendernos, posto que as imaxes son un tipo de dato fácil de xerar e manipular, así como fácil de entender polo ser humano e, sen embargo, difícil de entender polos ordenadores.

Dentro das aplicacións das redes neuronais para a análise de imaxes, destacan principalmente dúas: a clasificación de imaxes e a detección de obxectos. En concreto, imos centrarnos na detección de obxectos en imaxes e a clasificación dos mesmos, pois é esta aplicación na que estamos interesados por ser necesaria para poder acadar o obxectivo último deste traballo: aplicar a aprendizaxe profunda para a análise de imaxes mamográficas.

3.1. Principais aplicacións

Comecemos explicando en que consisten cada unha das aplicacións que vimos de comentar para poñernos en contexto.

3.1.1. Clasificación de imaxes

Empecemos vendo a clasificación de imaxes. Esta consiste en, dada unha imaxe, predicir a clase á que pertence o elemento máis dominante na mesma. Para entendelo mellor vexámolo cun exemplo.

Para isto imos considerar un dos conxuntos de datos máis populares na aprendizaxe profunda: o conxunto MNIST[1], que está formado por imaxes de números manuscritos comprendidos entre o 0 e o 9. Algunhas destas imaxes poden verse representadas na Figura 3.1.

Supoñamos entón que temos este conxunto de imaxes, para realizar a clasificación faltaría identificar as distintas clases nas que queremos clasificar as nosas imaxes. Para isto, bastaría considerar os díxitos do 0 ao 9 como clases, tendo así tantas clases distintas como conceptos hai no conxunto de imaxes, onde

[1]Na seguinte sección explicarase en máis detalle este conxunto de datos.

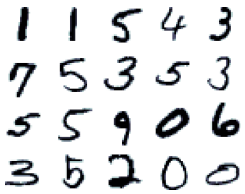

Figura 3.1: 20 imaxes do conxunto de datos MNIST, onde en cada unha das cales aparece representado un único díxito.

cada imaxe estaría etiquetada nunha e só nunha destas clases. Ao realizar a clasificación estaríamos dicindo que díxito aparece representado na imaxe.

Resumindo, o que queremos é construír un sistema que, dada unha imaxe, a clasifique na clase á que pertence, onde as posibles clases son:

$$0, 1, 2, 3, 4, 5, 6, 7, 8, 9.$$

Lembremos ademais que cada unha das imaxes pertencen a unha única clase.

Para obter esta clasificación, durante o adestramento ensínaselle á rede unha imaxe, en base a esta, a rede produce unha saída para cada categoría, predicindo a probabilidade de que a imaxe pertenza a cada unha desas clases. Deste xeito, o que se espera é que a categoría correcta sexa aquela que obtivo unha maior puntuación de entre todas as categorías posibles. Así, se quixésemos clasificar a imaxe da Figura 3.2, na cal aparece escrito o número 3, esperaríamos que a puntuación obtida para a clase 3 fose a máis elevada de todas.

Notemos ademais que isto é pouco probable que suceda antes de que se realice o adestramento pois, ata que este se realiza, a rede non pode aprender.

Figura 3.2: Imaxe do conxunto de datos MNIST.

Para realizar o adestramento calcúlase unha función obxectivo que mide o erro (ou a distancia) entre as puntuacións calculadas e os patróns de puntuación desexados. En función dos resultados obtidos coa función obxectivo a rede neuronal irá modificando progresivamente os seus pesos para ir reducindo este erro.

3.1.2. Detección e clasificación de obxectos

Despois de comentar brevemente a clasificación de imaxes, vexamos agora en que consiste a detección e clasificación de obxectos que, se ben son aplicacións distintas, están estreitamente ligadas xa que a clasificación dos obxectos lévase a cabo tras a detección dos mesmos, pode consultarse máis literatura en Chavan (2016) [2].

Comecemos vendo en que consiste a detección de obxectos para logo ver como realizar a súa clasificación. Como o seu propio nome indica, a detección de obxectos en imaxes consiste en detectar a presenza de obxectos nas mesmas, para velo maior claridade vexámolo cun exemplo.

Así, na Figura 3.3[2] podemos ver dous obxectos: un cadrado e un con forma de escaleira. A detección de obxectos consiste en identificar a presenza destes, a cal adoita indicarse mediante un *cadro delimitador* (*bounding box*) ao redor de cada un dos obxectos pertencentes ao conxunto de categorías previamente especificado, asignándolle a cada un deles a clase correspondente. Neste caso, as categorías serían cadrado ou triángulo e o cadro delimitador ten cor vermello, no caso do cadrado, e cor amarelo para o caso do obxecto con forma de escaleira. Notemos que estes obxectos poderían ser persoas, animais, vehículos, etc. Esta tarefa é moito máis complicada que a de clasificación, porque é posible que nunha mesma imaxe haxa múltiples obxectos, onde algúns dos cales podan chegar a solaparse ou ser apenas visibles.

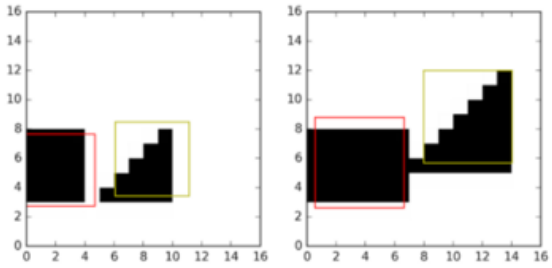

Figura 3.3: Ilustración da detección de obxectos.

Tras ver a grandes trazos en que consiste a detección de obxectos, vexamos agora a clasificación dos mesmos. A clasificación de obxectos consiste, basicamente, en facer o mesmo que facíamos no anterior apartado, pero agora en vez de coa imaxe enteira emprégase só unha parte dela. É dicir, unha vez detectados os posibles diferentes obxectos das imaxes, clasificámolos segundo a clase á que pertenzan. Vexámolo outra vez cun exemplo, na Figura 3.4[3] vemos que agora non só se detectan os obxectos, senón que xunto aos cadros delimitadores aparece o nome da clase á que pertencen: círculo (*circle*), rectángulo (*rectangle*) e triángulo (*triangle*).

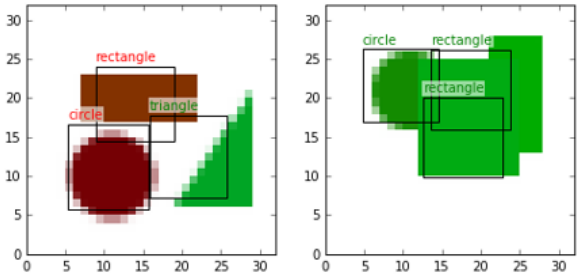

Figura 3.4: Ilustración da clasificación de obxectos.

Un claro exemplo desta aplicación serían os algoritmos de recoñecemento facial empregados por Facebook, que consigue detectar caras cando subes unha foto e dicirche de quen son.

Unha vez explicada a diferenza destas aplicacións e en que consiste cada unha delas, introduciremos as distintas técnicas de aprendizaxe profunda existentes, comezando pola máis clásica ata chegar ás redes

[2]Tomada de `https://towardsdatascience.com/object-detection-with-neural-networks-a4e2c46b4491`.
[3]Tomada de `https://towardsdatascience.com/object-detection-with-neural-networks-a4e2c46b4491`.

neuronais máis punteiras, pero antes é conveniente explicar como se manipulan os datos de entrada, imaxes neste caso, para que poidan ser procesados polas redes neuronais.

3.2. Vectorización da entrada

Xa vimos que á hora de traballar con técnicas de ML ou DL precisamos analizar case calquera tipo de data, incluíndo entre outros:

- Documentos de texto
- Imaxes
- Arquivos de sons
- Vídeos
- Datos secuenciais

Na introdución á aprendizaxe profunda que vimos de ver, fíxose evidente que aínda que os datos de entrada da rede neuronal sexan textos, imaxes, arquivos de sons, etc., o que está a entrar na rede non son estes formatos como tal, xa que as redes non poden traballar con estes tipos de datos directamente, senón que son valores numéricos. Por este motivo, é obvia a necesidade dunha transformación para converter os datos en valores numéricos que poidan ser procesados polas redes neuronais, para o cal se recorre á *vectorización* dos mesmos.

Estas técnicas de vectorización soen mencionarse moi brevemente en moitos dos artigos e libros sobre o tema, chegando incluso a ser omitidas moitas veces. Sen embargo, ímoslle dedicar aquí unha sección debido á súa importancia pois, como acabamos de comentar, son fundamentais para poder traballar coas redes neuronais.

Hai moitas técnicas diferentes para realizar a vectorización, tendo cada unha delas diferentes grados de efectividade no modelo de saída, pois esta dependenderá en gran medida de como de ben teñamos vectorizados os nosos datos de entrada. Se ben estas técnicas de vectorización aplícanse a calquera tipo de dato, centrarémonos naquelas específicas para imaxes por estar este traballo dedicado á aprendizaxe profunda para o procesamento de imaxes, máis especificamente para a detección de obxectos.

Para poñernos algo máis en contexto, lembremos a imaxe da Sección 2.2.1 (ver Figura 2.4) a cal o ollo humano interpreta como un coche, mentres que un ordenador só ve certos números. O que imos explicar agora é precisamente como podemos facer para que, dada unha imaxe, o ordenador consiga "ver" a matriz numérica, onde cada entrada era un número cuxo valor indica o brillo e a cor de cada un dos píxeles presentes.

Antes de proseguir expliquemos un pouco máis sobre os píxeles que, como xa dixemos antes, toman valores entre 0 e 255. Así, o número de bits por píxel determina cantas cores pode representar cada un deles. Deste xeito, un píxel de 1-bit só pode representar imaxes binarias, normalmente en branco e negro como sería a Figura 3.5 (esquerda). Os píxeles máis comúns son os de 8-bits, e poden amosar ata 256 cores por píxel, Figura 3.5 [4]. Para imaxes en escalas de grises o enteiro representando o píxel indica o brillo, sendo máis branco canto máis próximo estea a 255 e máis negro canto máis próximo estea ao 0. Na aprendizaxe profunda, para representar imaxes a cor adoitase empregar o modelo RGB[5] de representación de imaxes, nestes casos o valor do píxel pasa a estar formado por unha tupla de tres elementos, onde cada un deles representa unha das canles de cor.

[4] Pertencente ao conxunto de datos *Grocery* que introducimos na sección 3.4.5.

[5] RGB é un modelo de cor baseado na síntese aditiva, con el é posible representar unha cor mediante a mestura por adición das tres cores de luz primarias: vermello, azul e verde.

Figura 3.5: Imaxe en cor.

As imaxes son unha fonte de datos que pode ser examinada para extraer información e, tralo dito no parágrafo anterior, non é difícil darse de conta que esta información vén dada polos píxeles, pois dannos unha información numérica a cal pode ser procesada polas redes neuronais. Vexamos un exemplo cunha imaxe en branco e negro e como aplicalo logo a imaxes en cor.

Supoñamos para iso que o dato de entrada que queremos vectorizar é a imaxe da Figura 3.2. Esta imaxe pertence a un dos conxuntos de datos máis coñecidos no campo da aprendizaxe profunda: o MNIST[6]. Este, está formado por imaxes de tamaño 28×28 píxeles, onde en cada unha delas figura un número manuscrito entre 0 e 9. Na Figura 3.1 podemos ver máis imaxes pertencentes a este conxunto. Ademais, o conxunto está dividido en dous subconxuntos: un, formado por 60.000 imaxes, que se adoita empregar como mostra de adestramento; e outro, de 10.000, que se emprega como conxunto de validación.

Despois desta breve descrición sobre a imaxe que imos considerar, procedamos a explicar o proceso de vectorización que nos ocupa. Como vimos de dicir, temos unha imaxe de tamaño 28×28 píxeles. Polo tanto, unha posibilidade sería crear un vector de lonxitude 784:

$$\mathbf{x} = (x_1, ..., x_{784})$$

onde cada elemento do vector representase o valor de cada un dos píxeles da imaxe. Así pois, cada unha das nosas imaxes podería ser introducida na rede neuronal, empregando un vector análogo a este para cada unha das imaxes.

Sen embargo, esta forma de representar a imaxe non nos permite introducir interaccións entre os diferentes píxeles que conforman a imaxe. De feito, se o pensamos con detemento, non é difícil darse conta que máis alá da conveniencia para a realización de cálculos alxebraicos, representar unha imaxe de tamaño $n \times m$ píxeles como un vector de dimensión $1 \times (n \times m)$ non aporta ningunha vantaxe a maiores pois, nas imaxes, os píxeles están adxacentes uns a outros, pero co método que vimos de ver, esta adxacencia que podería revelarnos información, pérdese. Polo tanto, interésanos estudar outra posibilidade para representar os nosos datos, de tal forma que as imaxes poidan ser procesadas sen que sexa preciso transformalas nun vector, conservando así a correlación existente entre os píxeles.

Vexamos que ocorre cando lle pasamos este tipo de datos como entrada ás nosas redes neuronais. Para iso, consideremos unha rede neuronal multicapa cunha ou máis capas ocultas completamente conectadas, a cal supoñamos que queremos adestrar con imaxes de tamaño 32×32 con 3 canais RGB. Isto xeraría un total de 3072 pesos[7] por neurona na primeira capa oculta, e non é difícil decatarse de que, probablemente, vaiamos ter máis dunha neurona. Ademais, en moitos dos casos, quereremos ter múltiples capas ocultas na nosa rede multicapa, o que fará que se multiplique o número de pesos.

Unha imaxe normal pode chegar a ter perfectamente un tamaño de 300×300 con tres canais de

[6] Este pode consultarse en `http://yann.lecun.com/exdb/mnist/`.

[7] O número obtense de facer $32 \cdot 32 + 32$ e multiplicalo por cada unha das canles.

información RGB, o que crearía un total de 270000 conexións de pesos por cada unha das capas ocultas. Isto permítenos ver, como de rápido as redes multicapa crean un número masivo de conexións cando intenta modelar imaxes. Sen embargo, e pese a que agora pareza o contrario, a arquitectura da rede neuronal pode ser empregada grazas á estrutura das imaxes, de xeito que poidamos tomar vantaxe da mesma. As neuronas das redes neuronais profundas poden ser reorganizadas para formar a estrutura tridimensional típica dada polos seguintes tres atributos:

- Ancho
- Alto
- Profundidade

O ancho corresponderíase co ancho da imaxe en píxeles, analogamente para a lonxitude e a profundidade corresponderíase cos canais RGB da imaxe. De agora en diante usaremos o termo profundidade para referirnos a isto en vez de á profundidade da nosa rede neuronal, como estivemos a facer ata o de agora. Esta estrutura tridimensional pode apreciarse na Figura 3.6[8], onde podemos ver unha comparación coa estrutura das redes multicapas. Nesta Figura, podemos ver que, efectivamente, as neuronas da rede convolucional teñen tres dimensións e como cada unha das capas transforma unha entrada tridimensional nunha saída tamén tridimensional.

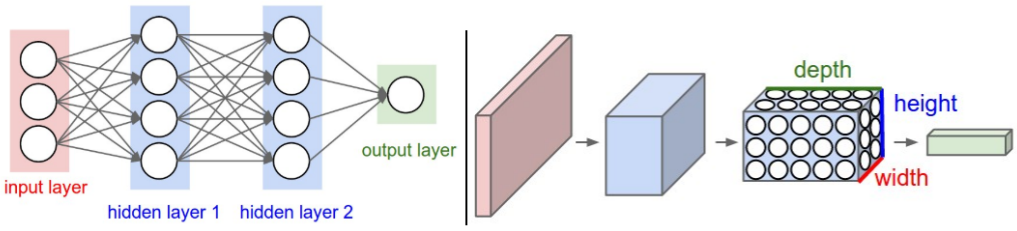

Figura 3.6: Comparación de rede multicapa (esquerda) con rede convolucional (dereita).

A profundidade será dunha ou tres canles dependendo de se a imaxe é en escala de grises ou se, pola contra, é a cor:

- Imaxes en escala de grises: represéntanse nunha matriz das mesmas dimensións que a imaxe onde cada entrada representa un píxel, que, por estar en escala de grises, toma un único valor. Polo tanto, as dimensións (D) do tensor serán:

$$D = (1, \text{anchura}, \text{altura})$$

- Imaxes a cor: xa dixemos que para a representación de imaxes en cor emprégase o modelo RGB, onde cada píxel toma tres valores, un para cada cor. Polo que agora, as imaxes serán representadas por matrices das mesmas dimensións que a imaxe, pero nas que cada elemento toma tres valores, tendo polo tanto un total de 3 canles:

$$D = (3, \text{anchura}, \text{altura}).$$

  A maioría das imaxes que se empregan en aplicacións reais son deste tipo.

Unha vez introducida polo tanto a necesidade das redes convolucionais vexámolas agora con máis detemento.

[8] Tomada de Li et al (2017) [25].

3.3. Redes convolucionais

O obxectivo das redes convolucionais é aprender características de orde superior nos datos mediante o emprego de convolucións. Estas redes xogan un papel moi importante nos campos de análise de sons e palabras, se ben o ámbito no que destacan por excelencia é a detección e clasificación de obxectos. Entre as súas aplicacións destacan os algoritmos de detección de caras, a condución autónoma e a robótica, entre outros moitos.

Antes de explicar en que consisten as redes convolucionais, vexamos de onde xurdiu a súa inspiración biolóxica, como tamén fixemos ao introducir as redes neuronais artificiais.

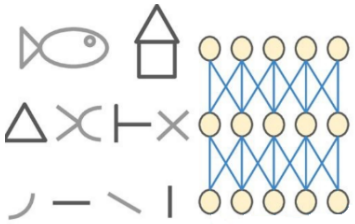

Figura 3.7: Exemplificación do funcionamento do córtex visual.

Así, as redes convolucionais están inspiradas no córtex visual dos animais. animais, no cal as células son sensibles a pequenas subrexións da imaxe de entrada, o que se coñece como campo visual. As células simples son activadas cando detectan patróns parecidos a bordos e, as células máis complexas, as cales son invariables á posición do patrón, actívanse cando reciben un campo visual máis grande. Isto aparece representado na Figura 3.7[9] onde, se nos fixamos, na esquerda da rede neuronal empézanse detectando bordos e, a medida que avanzamos a través da mesma, vanse formando imaxes máis complexas ata ver un peixe e unha casa.

3.3.1. Arquitectura

Como xa dixemos, as redes convolucionais son amplamente empregadas na detección e clasificación de obxectos. Vexamos con máis detalle a arquitectura destas redes, entendo como tal a estrutura da mesma. Existen varias variacións desta arquitectura, se ben todas se basean no mesmo patrón de capas, que se poden clasificar en tres grandes grupos: capas de entrada, capas de extracción de características e a capa de saída, na que se realiza a clasificación.

As capas de extracción de características son o máis novidoso das redes neuronais convolucionais e adoitan estar formadas por un patrón repetitivo das súas dúas compoñentes máis básicas: a capa convolucional e a capa de *pooling*, nas cales afondaremos un pouco máis adiante. Estas capas encárganse de atopar certas características nas imaxes e de ir construíndo características de orde superior progresivamente. Notemos que isto é o fundamento principal da aprendizaxe profunda, pois é a propia rede quen aprende cales son as características relevantes sen teren que ser estas indicadas por nós.

Vexamos brevemente un exemplo de arquitectura para asentar unha pequena base sobre a cal ir desenrolando logo os diferentes conceptos que se van ver. Así, un exemplo de arquitectura moi empregado nas redes convolucionais consiste nunha primeira capa de entrada, unha capa de convolución seguida dunha capa con función de activación ReLU, unha capa de *pooling* e, finalmente, unha ou varias capas

[9]Tomada de Géron (2017) [10].

completamente conectadas, que serán as encargadas de realizar a clasificación. Vexámolo máis detallado, para iso supoñamos que temos unha imaxe de entrada de tamaño 32×32 e en cor, polo que teremos 3 canais:

- Capa de entrada: a capa de entrada é onde introducimos a nosa imaxe para que esta sexa procesada pola rede. Esta capa, como xa dixemos anteriormente, vai aceptar unha entrada tridimensional que, xeralmente, ten a mesma forma espacial do tamaño da imaxe e cuxa profundidade dependerá das canles e cor, neste caso $[32 \times 32 \times 3]$.
- Capa convolucional: as capas convolucionais calculan a saída das neuronas que están conectadas ás rexións locais da imaxe de entrada. Para isto, cada unha delas calcula un produto entre os seus pesos e entre a rexión pequena á que están conectadas. O tamaño da saída dependerá do número de filtros que empreguemos, así, de empregar k filtros sería $[32 \times 32 \times k]$.
- Capa de activación ReLU: a capa ReLU aplica unha función de activación que deixa o tamaño sen cambios.
- Capa de *pooling*: a capa de *pooling* realizará unha redución ao longo das dimensións espaciais (ancho e alto).
- Capas completamente conectadas: como o seu nome indica, cada neurona nesta capa é conectada con todas as neuronas da capa seguinte. É nestas capas onde se calculan as puntuacións, obtendo unha saída de tamaño $[1 \times 1 \times n]$, onde n é o número total de categorías.

Vexamos agora con máis detemento estas capas, en especial as máis novidosas isto é: as capas de convolución e as capas de *pooling*.

Capas de convolución

Dado o seu nome, non é difícil darnos conta de que as capas convolucionais xogan un papel esencial nas redes que estamos a ver. Estas capas, o que fan é transformar as entradas, mediante a realización da convolución, obtendo unha saída que ten as mesmas dimensións espaciais que a entrada, ou máis pequenas, pero que, ás veces, aumenta o número de elementos na terceira das dimensións.

Pero, como se realiza esta transformación se, como xa dixemos, cando as entradas teñen grandes dimensións, non é práctico traballar con capas completamente conectadas? A resposta é obvia, non traballaremos con este tipo de capas.

Así, o que se fai nas redes convolucionais é conectar cada unha das neuronas da capa cunha rexión local do obxecto de entrada, onde o tamaño desta rexión é un hiperparámetro que recibe o nome de *campo receptivo*. Isto aparece exemplificado na Figura 3.8[10], onde temos unha entrada de dimensións $5 \times 5 \times 3$. Nesta imaxe, podemos ver como as neuronas conéctanse unicamente a unha pequena rexión da imaxe, na que notemos só se reduce o tamaño espacial, que pasa a ser 2×2, conservando a profundidade.

Falaremos dos demais elementos que aparecen na imaxe máis adiante, pero, polo de agora, notemos que na capa convolucional hai un total de 5 neuronas ao longo da profundidade que están mirando á mesma rexión da entrada. Estas neuronas, son exactamente iguais que as que vimos de ver nas redes multicapa, sendo o único cambio que, agora, en vez de estar completamente conectadas o están localmente.

Unha vez explicado isto, vexamos en que consiste a convolución, operación que lle dá nome a este tipo de redes e fundamental nas capas convolucionais.

[10] Tomada de Patterson et al (2017) [30].

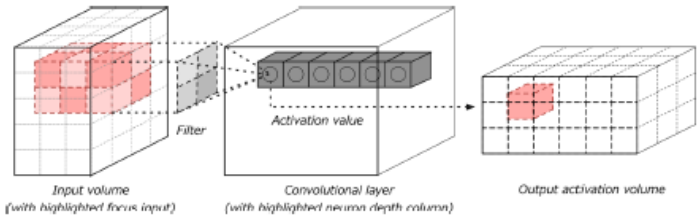

Figura 3.8: Capa convolucional.

A convolución defínese como unha operación matemática que escribe unha regra sobre como fusionar dous conxuntos de información. Esta, toma unha entrada, aplica un *filtro* de convolución (*kernel*) e devolve un mapa de activación (*feature map*), como pode apreciarse na Figura 3.9[11], que pasaría como entrada da seguinte capa da rede.

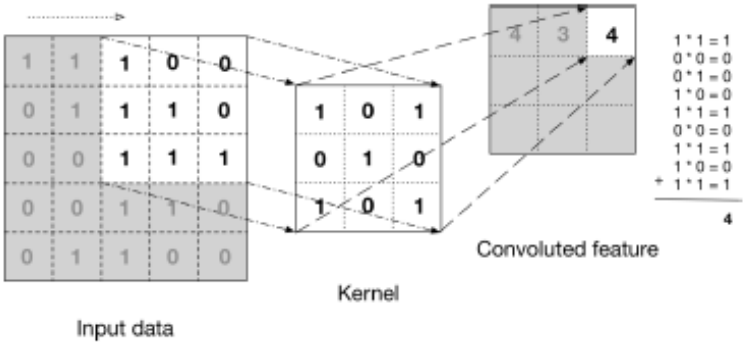

Figura 3.9: A operación de convolución.

Tras ver a idea básica que hai detrás da convolución, afondemos algo máis nestas capas, as cales, ao igual que ocorría coas redes multicapas, serán adestradas empregando o algoritmo de propagación cara atrás xunto co gradiente estocástico, de xeito que as puntuacións obtidas para as distintas clases sexan consistentes coas etiquetas do conxunto de adestramento. Unha vez dito isto, enuméranse a continuación as compoñentes principais destas capas, as cales son:

- Filtros
- Mapas de activación
- Hiperparámetros específicos

Comecemos vendo os filtros, que poden ser aprendidos. Cada un destes filtros, que non son outra cousa que os parámetros das capas convolucionais, serán tamén estruturas tridimensionais cuxas dimensións espaciais adoitan ser menores que a dos datos de entrada e cuxa profundidade adoita coincidir coa susodita. Por exemplo, un filtro típico nunha primeira capa convolucional, que recibe como entrada unha imaxe en cor, podería ter un tamaño de $5 \times 5 \times 3$, é dicir, 5 píxeles de ancho e alto e unha profundidade de 3, unha por cada canle de cor.

Os filtros, escorréganse a través do ancho e largo dos datos de entrada para producir a saída, este desprazamento vén determinado por un parámetro chamado *paso* (*stride*) que veremos máis adiante.

[11] Tomada de Patterson et al (2017) [30].

Para calcular a saída, en cada un destes deslizamentos, os valores do filtro, o cal foi previamente inicializado, son multiplicados elemento a elemento polos valores dos datos de entrada. Os elementos así obtidos son sumados, creando deste xeito unha entrada do mapa de activación, como aparece recollido na Figura 3.9. Así pois, en cada unha das capas, teremos un conxunto completo de filtros -por exemplo, 11- e cada un deles producirá un mapa de activación. Estes mapas de activación son amoreados ao longo da dimensión de profundidade producindo deste xeito a saída da capa.

A pesares de que unha capa convolucional só pode detectar características elementais, a natureza da convolución, onde a saída dunha capa convolucional é a entrada doutra, permite a extracción de características de maior orde a medida que avanzamos a través da rede neuronal. Así, nas primeiras capas, a rede aprenderá os filtros que se activan cando vén algún tipo de patrón, como un bordo dalgunha orientación ou unha mancha dalgunha determinada cor, o que permitirá que en capas superiores, se detecten patróns máis complexos e, así, sucesivamente.

Grazas a isto, a convolución pode facer que os bordos se volvan máis prominentes ou que toda a imaxe se volva máis borrosa. Isto pode ser interesante para extraer características específicas de certas imaxes, que indiquen a pertenza a unha clase en particular. Despois da convolución, será máis sinxelo que unha capa completamente conectada aprenda as características específicas e identificadoras de cada imaxe do que o sería de non terse realizado o proceso de convolución.

As arquitecturas de redes convolucionais, que se verán na Subsección 3.2.2, amosaron que o número de capas convolucionais empregado é determinante neste tipo de redes. De feito, un erro común nas redes convolucionais é empregar filtros de gran tamaño pois, a miúdo, pode obterse o mesmo resultado empregando unha capa convolucional cun filtro de tamaño 9×9 que xuntando dúas capas convolucionais con filtros de tamaño 3×3, precisando este último enfoque de menos tempo computacional e empregando menos parámetros.

Vexamos agora con máis detemento os mapas de activación. Se lembramos, cando na Sección 2.3.2 falamos das funcións de activación, dixemos que a activación era un valor numérico que decidía se unha neurona ía ser activada ou non. Do mesmo xeito, agora falaremos de se o filtro está activado ou non, isto é, se pasa información a través del ou non. Pero, quen xoga agora o papel da activación? A resposta volve ser obvia: o mapa de activación, os cales xa vimos como son calculados (ver Figura 3.9). Deste xeito, o mapa de activación para un filtro específico é a saída bidimensional resultante de realizar a convolución de dito filtro cos datos de entrada da capa convolucional.

Cada filtro, produce un único mapa de activación que, como acabamos de dicir, é bidimensional, sen embargo, a entrada ás nosas capas era unha estrutura tridimensional. Para solucionar isto e facer que, efectivamente, a entrada da seguinte capa teña tres dimensións, o que se fai é amorear todos os mapas de activación xerados a través da lonxitude de profundidade, como xa adiantáramos hai un anaco.

Tras todo o que acabamos de ver, non é difícil darse conta de que as entradas desta saída foron obtidas estudando soamente unha pequena porción do dato de entrada. Posto que, como dixemos, os filtros están conectados só a unha rexión do dato de entrada, a través das conexións locais descritas anteriormente. Isto, permítenos reducir o número de parámetros que precisamos adestrar por capa, á vez que nos permite manter unha boa calidade de extracción de características. Outra técnica empregada polas capas convolucionais para reducir o número de parámetros é o que se coñece como compartimento de parámetros (*parameter sharing*), técnica na cal non afondaremos.

Explicamos xa a conectividade de cada neurona das capas convolucionais ao dato de entrada, pero, que ocorre coas neuronais na saída? Cantas neuronas hai? Como se organizan? Para responder a estas preguntas, falaremos agora dos hiperparámetros específicos das capas convolucionais, centrándonos nos seguintes:

- Profundidade da saída
- Paso (*Stride*)

- Recheo (*Padding*)

A profundidade da saída controla o número de neuronas que hai na capa convolucional conectada á rexión da entrada que se estea a empregar. Esta profundidade, correspóndese co número de filtros que nos gustaría empregar, notemos que empregar varios filtros é interesante, pois cada un deles aprenderá a buscar algo distinto a partires do dato de entrada. Por exemplo, como xa comentamos, na primeira capa certas neuronas poderían ser activadas en presenza de bordos orientados e outras ante a presenza de manchas de cor. Referirémonos ao conxunto de neuronas que están conectadas á mesma rexión de entrada como *profundidade de columna*.

Seguidamente falaremos do paso que, como dixemos no seu momento, fai referencia a canto escorregamos o filtro na imaxe. A súa principal función é que permite reducir o tamaño da saída de forma espacial mantendo a profundidade. Vexámolo con dous exemplos, así, supoñamos que temos unha imaxe de tamaño 7×7, un filtro de tamaño 3×3 e un paso de 1, ver Figura 3.10[12] (esquerda), polo que estaríamos a mover o filtro un píxel de cada vez e, en cada posición, estaríamos calculando a entrada do mapa de activación que, finalmente, sería de tamaño 5×5, como se pode ver na Figura 3.10 (dereita).

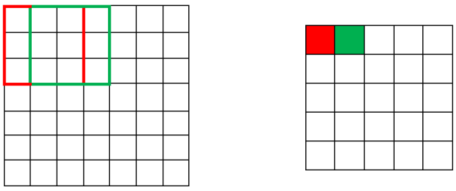

Figura 3.10: Exemplo de paso cando é igual a 1.

Na Figura 3.11 podemos ver o mesmo exemplo pero con paso 2. Deste xeito, vemos que agora o filtro desprázase de dous en dous píxeles e que a saída reduciuse a un tamaño de 3×3. Polo tanto, canto maior é o paso, menores dimensións espaciais terá a saída. Notemos ademais que, canto maior é o paso empregado, menor superposición se produce entre os filtros. Así, agora cada filtro só comparte tres píxeles cos adxacentes, mentres que no exemplo anterior compartía o dobre.

Finalmente, comentar que o paso non só reduce o tamaño da saída, senón que tamén reduce significativamente o número de convolucións realizadas en cada unha das imaxes. Isto, significa que a redución de convolucións e do tempo de procesamento nas capas seguintes será aínda maior.

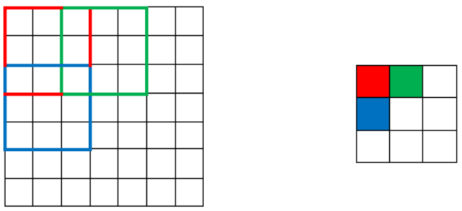

Figura 3.11: Exemplo de paso cando é igual a 2.

O último hiperparámetro do que imos falar é o recheo de ceros (*padding*) o cal serve para controlar o tamaño espacial da saída. Este hiperparámetro é especialmente útil naqueles casos nos que, por algún motivo, nos interese manter o tamaño espacial da entrada na saída. Para isto, o que se fai é reencher

[12] Tomada de `http://deeplearningthesis.com/2018/02/08/understanding_the_mathematics_of_the_convolution_layer.html`.

a entrada con ceros ao redor, como se pode ver na Figura 3.12[13], onde temos unha imaxe de tamaño espacial 4×4 á cal lle engadimos un recheo de 1 para que, á hora de escorregar o filtro, a saída manteña este tamaño.

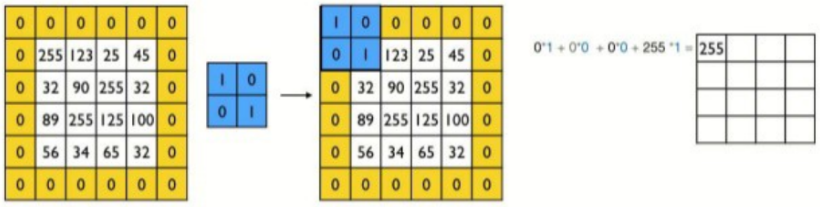

Figura 3.12: Exemplo de aplicación de recheo de ceros.

O tamaño espacial da saída pode ser calculado como unha función do volume da entrada, W, o tamaño do filtro empregado na capa convolucional, F, o paso que é aplicado, S, e en función do número de ceros empregados ao realizar o recheo, P. Deste xeito, a fórmula empregada para calcular cantas neuronas encaixan na saída vén dada por

$$\frac{W - F + 2P}{S + 1}.$$

Capa de activación ReLU

Ao introducir as redes convolucionais, xa dixemos que unha das arquitecturas máis típicas era aquela que combinaba capas de convolución, con capas de activación ReLU e capas de *pooling*. Vexamos en que consisten as segundas. A capa de activación ReLU aplica a función de activación ReLU elemento a elemento aos datos de entrada, fixando, como vimos na Sección 2.3.2, un limiar no 0. Esta capa unicamente modifica unicamente os valores dos píxeles, isto é, non produce ningunha modificación das dimensións.

A función de activación máis empregada nas redes convolucionais adoita ser a ReLU, por ser das máis rápidas, xa que, como vimos de ver, adestrar unha rede convolucional é unha tarefa computacionalmente cara polo que diminuír o tempo de adestramento constitúe unha vantaxe importante.

Capa de pooling

Entramos finalmente no último tipo de capa que imos ver das redes convolucionais: a capa de *pooling*.

Estas capas adoitan intercalarse entre capas convolucionais sucesivas co obxectivo de iren reducindo progresivamente o tamaño espacial do dato que se recibiu como entrada, reducindo tamén o tempo empregado en adestramento ademais de axudar a reducir o sobreaxuste. Estas capas operan de xeito independente en cada sector de profundidade procedente da entrada, a cal é redimensionada espacialmente mediante a operación

$$\text{máx}(),$$

á cal nos referiremos como *max pooling*. Esta técnica, que se explica con algo máis de detalle na sección 3.4.2, o que fai é tomar a saída da capa convolucional e dividila en diferentes celas, onde tan só o valor máis grande de cada unha destas será empregado na seguinte capa da rede.

Para realizar esta redución da dimensión, as capas de *pooling* empregan filtros que normalmente teñen un tamaño de 2×2 cun paso de 2. Deste xeito, o tamaño de cada porción de profundidade do dato

[13]Tomada de `https://medium.com/@pushpanjalipd/conventional-neural-network-deep-learning-a2ad1cf4487a`.

de entrada redúcese por un factor de 2 nas dimensións espaciais, o que fai que o 75 % das activacións sexan descartadas.

Este tamaño de 2×2 non debería sorprendernos pois, de ser maior, esta capa podería chegar a ser moi destrutiva ao poder reducir demasiado o tamaño orixinal da entrada. Na Figura 3.13[14] podemos ver un exemplo de como se realizaría a convolución empregando un filtro coas características que vimos de comentar.

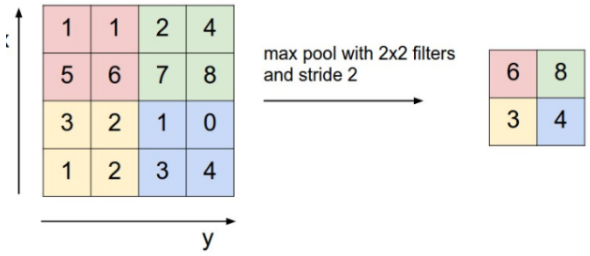

Figura 3.13: Exemplo de aplicación da técnica *max pooling* cun filtro de tamaño 2×2 e paso 2.

Outra das técnicas máis empregadas xunto co *max pooling* é o *average pooling* o cal, en vez de tomar o valor máximo de cada unha das celas, toma o valor medio. A vantaxe deste método é que tomando o valor medio, asegurámonos que non se está a producir unha perda importante da información. Sen embargo, se ben esta técnica foi moi empregada, o seu uso caeu en favor do *max pooling* por ter este obtido mellores resultados na práctica.

Máis xeralmente, a capa de *pooling* acepta unha entrada de calquera tamaño

$$w_1 \times h_1 \times d_1$$

onde w fai referencia ao ancho, h ao alto e d á profundidade e require dous parámetros: o tamaño do filtro, F e o paso, S. Producindo un volume de tamaño

$$w_2 \times h_2 \times d_2$$

onde

- $w_2 = \dfrac{w_1 - F}{S} + 1$
- $h_2 = \dfrac{w_2 - F}{S} + 1$
- $d_2 = d_1$

Antes de dar por finalizada a sección referente a redes convolucionais, vexamos a arquitectura dalgunhas das máis populares.

3.3.2. Redes convolucionais populares

Tipicamente, a arquitectura das redes convolucionais xunta unhas cantas capas convolucionais, cada unha delas seguida dunha capa con función de activación ReLU, seguidas dunha capa de *pooling*.

[14] Tomada de Li et al (2017) [25].

Despois desta, viría outro conxunto de capas convolucionais, xunto coa súa correspondente capa ReLU, e así sucesivamente.

Por outra banda, a imaxe de entrada vai diminuíndo o seu tamaño a medida que avanza a través da rede neuronal, pero tamén vai volvéndose cada vez máis e máis profunda, é dicir, con máis mapas de activación, como se pode apreciar na Figura 3.14[15]. Notemos que cara ao final, engádese tamén unha rede neuronal máis sinxela composta por capas completamente conectadas, se ben na Figura 3.14 só aparece unha (*hidden*) xunto coa a capa final, cuxa saída é a predición.

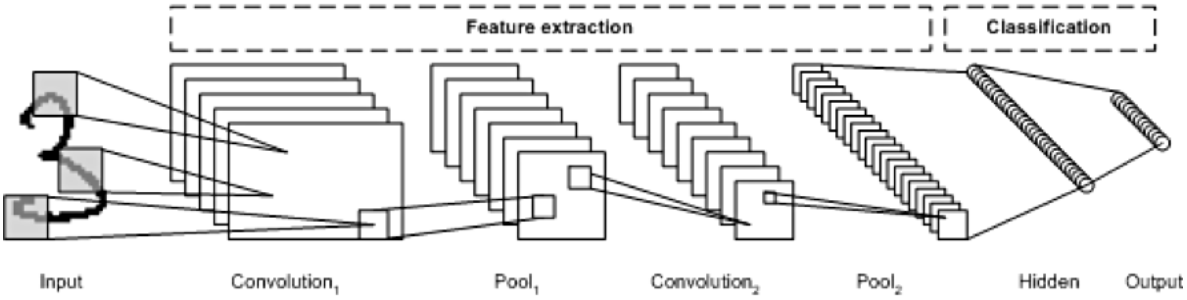

Figura 3.14: Arquitectura típica dunha rede convolucional.

Co paso dos anos, desenroláronse múltiples variantes desta estrutura que acabamos de comentar brevemente, o que deu lugar a grandes avances no campo. Unha boa medida deste progreso, é a taxa de erro en competicións como o desafío ILSVRC Imagenet, do cal se pode consultar máis información en `http://image-net.org`. Nesta competición, a taxa de erro das cinco primeiras predicións para clasificación de imaxes, caeu do 26 % ao 3 % en tan só seis anos. Esta taxa, non é outra cousa que o número de imaxes cuxa correcta etiqueta non foi incluída nas cinco clases con maior probabilidade de pertenza preditas polo modelo en cuestión. Vexamos algunhas das redes gañadoras desta competición, o que nos dará tamén outra percepción do funcionamento das redes convolucionais.

Así pois comezaremos falando da arquitectura clásica LeNet-5 (1998) para falar despois de tres dos gañadores da competición de ILSVRC: AlexNet (2012), GoogleNet (2014) e ResNet (2015).

LeNet-5

LeNet-5, creada en 1998 por Yan LeCun, é unha das primeiras arquitecturas de redes convolucionais exitosas, e unha das máis coñecidas. Esta foi amplamente empregada para o recoñecemento de díxitos escritos a man, sendo adestrada co conxunto de datos MNIST do que falamos anteriormente.

AlexNet

AlexNet é de lonxe unha das arquitecturas de redes convolucionais que máis revolocionou o panorama no campo da intelixencia artificial. Esta, foi desenrolada en (2012) por Alex Krizhevsky, Ilya Sutskever e Geoffrey Hinton e conseguiu bater a taxa de erro acadando un 17 % que, se ben é un erro bastante grande, mellorou en gran medida o erro máis pequeno obtido anteriormente, 26 %. AlexNet, é moi semellante a LeNet-5 se ben novidosa pois é moito máis grande e profunda, ademais de ser a primeira en amorear capas convolucionais directamente unha despois da outra en lugar de ilas alternando con capas de *pooling*. Máis literatura pode atoparse en Krizhevsky et al (2012) [22].

[15] Tomada de `https://www.ibm.com/developerworks/ssa/library/cc-machine-learning-deep-learning-architectures/index.html`.

GoogleNet

A arquitectura de GoogleNet foi desenrolada por Szegedy e outros en 2014 e gañou a competición de ISVRC, diminuíndo a taxa de erro por debaixo do 7 %. Esta mellora dos resultados foi, en gran parte, debida a que a rede era máis profunda que as anteriores, o cal foi posible polo que se coñece como *inception module* o que permite a GoogleNet empregar os parámetros de forma moito máis eficiente que arquitecturas anteriores. Así, GoogleNet ten dez veces menos parámetros que a súa predecesora, AlexNet.

VGGNet

VGGNet foi desenrolada por Simonyan e Zisserman no ano 2015 e é unha das arquitecturas máis populares, demostrando que a profundidade da rede é clave para o seu rendemento. Esta trátase dunha estrutura extremadamente homoxénea. Unha das súas principais desvantaxes é que é máis custosa de avaliar e que require unha gran cantidade de memoria. Ademais, contén aproximadamente 140 millóns de parámetros, máis do dobre que AlexNet, se ben a maioría deles atópanse nas capas completamente conectadas e non causan unha gran diminución do rendemento se son eliminados.

ResNet

Finalmente, presentamos a arquitectura ResNet, desenrolada en 2015 por He entre outros, que obtivo un erro excepcionalmente bo, inferior a 3.6 %. Para isto empregouse unha rede convolucional extremadamente profunda, composta dun total de 152 capas. Ademais, esta arquitectura introduciu o concepto de aprendizaxe residual, se ben este concepto é bastante avanzado polo que non afondaremos no mesmo.

3.4. Arquitecturas para a detección de obxectos

Tras ver algunha das arquitecturas máis clásicas das redes neuronais convolucionais, imos centrarnos agora en arquitecturas específicas para o problema de detección e clasificación de obxectos. Con isto en mente, falaremos brevemente de R-CNN para introducir logo dúas novas arquitecturas modificacións desta última: Fast R-CNN e Faster R-CNN.

3.4.1. R-CNN

Comecemos vendo en que consiste R-CNN, se ben non entraremos en moito detalle pois simplemente nos servirá de base para introducir despois o Fast R-CNN e o Faster R-CNN.

Como xa dixemos, agora o problema que nos ocupa e no cal nos imos centrar é a detección e a clasificación de obxectos. Notemos que, para isto, ao contrario do que sucedía na clasificación de imaxes, onde só precisábamos como dato de entrada imaxes, agora será necesario tamén localizar os obxectos que se atopan dentro de cada unha destas imaxes.

A idea básica de R-CNN é tomar unha rede neuronal, que orixinalmente foi adestrada para clasificación de imaxes empregando millóns de imaxes etiquetadas e, modificala co obxectivo de realizar detección de obxectos. Na Figura 3.15[16] aparece representada unha idea esquemática de como funcionan estas redes.

[16] Tomada de Girshick (2015) [11].

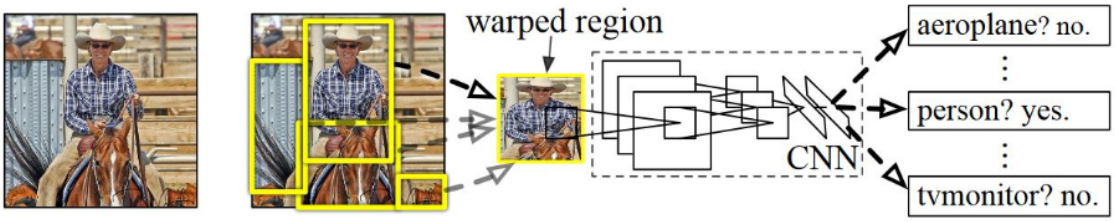

Figura 3.15: Idea esquemática de como funciona Fast R-CNN.

Vexamos a grandes trazos o funcionamento de R-CNN. Deste xeito, vemos que lle temos que proporcionar á rede dúas entradas: unha primeira, a imaxe; e unha segunda, formada polo conxunto de cadros delimitadores das rexións de interese. Vexamos como obter estes cadros delimitadores que son independentes da categoría, para iso, farase uso do método de busca selectiva.

Busca selectiva

A busca selectiva é un método empregado para atopar un conxunto grande de posibles localizacións na imaxe, independentemente da clase á que pertenza o obxecto en cuestión. Referirémonos a estas posibles localizacións como rexións de interese ou *RoI*[17], que non son outra cousa que zonas da imaxe nas que se atopa un obxecto.

Figura 3.16: Exemplificación do proceso de busca selectiva.

Para completar as rexións de interese xeradas pola busca selectiva, engádense a estas outras rexións de diferentes escalas e ratios que cobren, de forma uniforme, toda a imaxe. Así, na Figura 3.16[18] (esquerda) amósase un exemplo da saída da busca selectiva, onde cada unha das posibles rexións, as cales teñen diferentes tamaños e ratios, aparece representada cun cadro delimitador verde. Seguidamente, as rexións de interese máis pequenas son descartadas (medio) e, finalmente, engádense novas rexións que cobren uniformemente toda a imaxe, ver Figura 3.16 (dereita). O obxectivo da busca selectiva, é atopar un conxunto pequeno de rexións de interese, que cubran correctamente tantos obxectos como sexa posible, para logo seren empregados en R-CNN.

A miúdo, as saídas deste método coinciden sobre o mesmo obxecto nunha imaxe ou cubríndoo de forma parcial. Estas rexións de interese, precisan fusionarse para obter as localizacións exactas dos

[17] Polas súas sigles en inglés *Region Of Interest*.
[18] Tomada de Kranen et al (2017) [21].

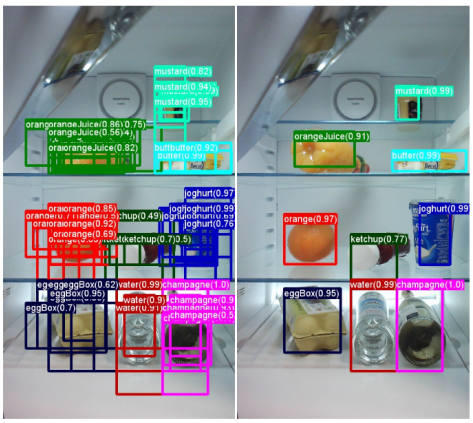

Figura 3.17: Exemplificación do proceso de supresión non máxima.

obxectos na imaxe. Para facer isto, emprégase a técnica de supresión non máxima, a cal identifica a rexión de interese que cubre mellor as localizacións reais dos obxectos e descarta todas as demais. Isto, impleméntase seleccionando de forma iterativa a rexión con maior confianza e eliminando todas as outras que, pertencendo á mesma clase, solapen significativamente a que foi escollida. O limiar para esta superposición pode ser establecido. Na Figura 3.17[19] vemos un exemplo desta técnica antes de ser aplicada (esquerda) e despois (dereita).

Unha vez que temos as rexións de interese, estas son convertidas a un tamaño fixo e, finalmente, pásanselle á rede, ao final da cal hai un clasificador que calcula a probabilidade de pertenza ás diferentes clases consideras para cada unha das rexións de interese, emitindo a etiqueta da clase na que se obtivo unha maior probabilidade. Así, no exemplo da Figura 3.15 temos un señor montado a cabalo e a saída que obtemos, é a de que nesa rexión de interese aparece unha persoa.

Se ben os resultados obtidos coa R-CNN acadan unha precisión moi boa en detección de obxectos facendo emprego dunha rede neuronal para clasificar as rexións de interese (consultar máis literatura en Girshick et al (2014) [12]), R-CNN ten tamén unha serie de desvantaxes que enumeramos a continuación:

- O adestramento lévase a cabo en varias etapas diferentes. Así, como vimos, R-CNN primeiro refina a rede convolucional para as rexións de interese e, unha vez extraídas as características, axusta un clasificador. Estes clasificadores actúan como detectores de obxectos substituíndo ao clasificador *softmax* empregado durante o axuste. Finalmente, apréndense os regresores dos cadros delimitadores.
- O adestramento é caro tanto en espazo como en tempo. Isto é así pois as características son extraídas de cada proposta de obxectos e escritas no disco. Con redes neuronais moi profundas, como VGG16, este proceso pode levar ata 2.5 días de GPU para un conxunto de adestramento de 5000 imaxes. Estas características requiren centos de gigabytes de almacenamento.
- A detección de obxectos é lenta. Durante o tempo de validación, as características son extraídas de cada proposta de obxecto en cada imaxe do test. Coa rede VGG16, a detección pode levar ata 47 segundos por imaxe empregando GPU.

Co obxectivo de mellorar R-CNN propúxose Fast R-CNN que, ademais de arranxar as desvantaxes que vimos de comentar, tamén mellorou o tempo de execución e a precisión.

[19] Tomada de Kranen et al (2017) [21].

3.4.2. Fast R-CNN

Tras ver en que consiste R-CNN, así como as súas desvantaxes, presentamos agora o Fast R-CNN, proposto por Girshick (2015) [11], que, como vimos de dicir, é unha alternativa para a detección de obxectos mellorando tanto o tempo que se emprega no adestramento, como a precisión dos resultados obtidos coa mesma.

Así, entre as súas vantaxes atópanse:

- Mellor calidade de detección (mAP) que R-CNN.
- O adestramento realízase nunha única etapa.
- No adestramento actualízanse todas as capas das redes.
- Non é preciso almacenar as características.

A idea de Fast R-CNN é basicamente a mesma que a de R-CNN: tomar unha rede neuronal que orixinalmente foi adestrada para clasificación de imaxes e, posteriormente, modificala. De feito, a idea esquemática vén a ser a mesma que a que presentamos na Figura 3.15 para R-CNN se ben agora, unha das diferenzas, ademais das outras que vimos de comentar, é que se empréga un clasificador softmax.

Vexamos a grandes trazos o funcionamento de Fast R-CNN. Neste, temos que proporcionarlle á rede dúas entradas; unha primeira, a imaxe; e unha segunda, formada polo conxunto dos cadros delimitadores das rexións de interese. Estas son convertidas a un tamaño fixo e, finalmente, pásanselle á rede, ao final da cal hai un clasificador softmax que produce tanto unha etiqueta como unha probabilidade para cada unha das *RoI*.

Vexamos agora cun pouco máis de detalle o funcionamento de Fast R-CNN. Na Figura 3.15 vemos que Fast R-CNN toma como entrada tanto a imaxe como as *RoI*, que veñen dadas por un cadro delimitador (de cor vermello na imaxe). Cada unha das imaxes é procesada pola rede empregando unha rede convolucional (*Deep ConvNet* na Figura 3.18[20]) a partir da cal se obtén un mapa de activación (*Conv feature map*).

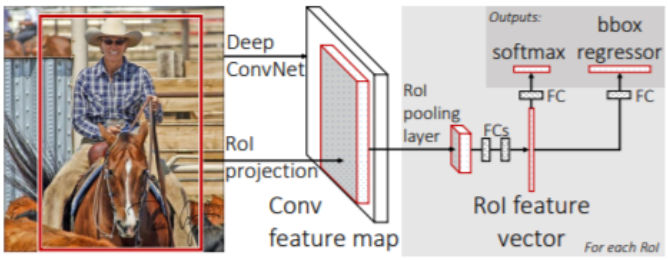

Figura 3.18: Arquitectura do Fast R-CNN.

Unha vez temos creado este mapa, é cando entran en xogo as *RoI*, que foron obtidas facendo uso de novo da busca selectiva e da supresión non máxima. Lembremos que estas rexións de interese podían ser de diferente tamaño, polo que, agora, empregando *max pooling*, convertiranse todas elas ao tamaño de entrada que espera a seguinte capa.

Cada unha destas *RoI* reescaladas é introducida como entrada da rede neuronal, a cal comeza cunha secuencia de capas completamente conectadas que, finalmente, se bifurcan en dúas capas de saída irmás:

[20]Tomada de Ren et al (2016) [33].

a primeira produce a probabilidade softmax de pertenza a cada unha das $K+1$ posibles clases[21]; a segunda, ten como saída catro números reais para cada unha das K clases, os cales son empregados para refinar os cadros delimitadores de cada unha das clases.

Capa de RoI *max pooling*

Vexamos agora cun pouco máis de detemento como funciona a capa de *RoI pooling*, a cal foi proposta por Girshick (2015)[11]. Esta capa toma dous valores de entrada:

- O mapa de activación de tamaño fixo, obtido a través dunha rede convolucional con varias capas convolucionais e de *max pooling*.
- Unha matriz de tamaño $n \times 5$ onde n é o número de rexións de interese por imaxe. Na primeira columna da matriz, atópase o índice da imaxe á que pertence a rexión de interese e nas outras catro atópanse os valores

  $$(r, c, h, w)$$

  que especifican as coordenadas da esquina superior esquerda (r, c) do cadro delimitador así como a súa altura e o seu ancho, h e w respectivamente.

Ata agora temos visto que toma a capa de *max pooling* como entrada, pero que é o que realmente fai esta capa? Para cada rexión de interese da lista de entrada, que como xa dixemos pode ser de calquera tamaño, o que fai esta capa é tomar a sección do mapa de activación que lle corresponde e reescalala a un tamaño predefinido. Para isto séguense os seguintes pasos:

1. Dividir a rexión de interese nun grid de tamaño fixo $H \times W$ en tantas seccións como tamaño que queiramos que teña a nosa saída, o tamaño destas seccións será aproximadamente de

   $$\frac{h}{H} \times \frac{w}{W},$$

   onde tanto H como W son hiperparámetros independentes de cada unha das rexións de interese consideradas.
2. Atopar o valor máximo en cada unha das seccións calculadas no punto anterior.
3. Copiar estes valores máximos á saída.

Deste xeito, dada unha lista de rectángulos de diferentes tamaños, podemos obter rapidamente unha lista de mapas de activación cun tamaño fixo. Notemos ademais que o tamaño deste mapa depende unicamente do número de seccións na que dividimos a rexión de interese.

Unha das grandes vantaxe do *RoI pooling* é a velocidade de procesamento, posto que aínda que houbese múltiples propostas de obxectos na imaxe, poderíamos seguir empregando o mesmo mapa de activación para cada unha delas e, dado que o cálculo das convolucións en etapas temperáns é moi custoso, isto pode aforrarnos moito tempo de execución.

Se ben agora está un pouco máis claro cal é o proceso levado a cabo por esta capa, vexámolo cun pequeno exemplo[22] para rematar de aclarar todas as posibles dúbidas. Se ben normalmente haberá múltiples rexións de interese, nós simplificaremos un pouco as cousas e ilustraremos os cálculos que se realizan nesta capa considerando un único mapa de activación de tamaño 8×8, unha única rexión

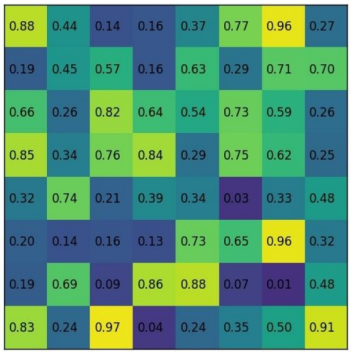

Figura 3.19: Exemplo de mapa de activación de tamaño 8×8.

de interese e un tamaño de saída 2×2. Así, un posible exemplo de mapa de activación sería o que se recolle na Figura 3.19.

Supoñamos tamén que temos unha rexión de interese que vén dada pola tupla de catro elementos

$$(0, 3, 5, 7).$$

É dicir, a esquina superior esquerda localízase nas coordenadas $(0, 3)$ e terá unha altura de 3 unidades e un ancho de 5, na Figura 3.20 podemos ver o cadro delimitador desta rexión de interese en negro.

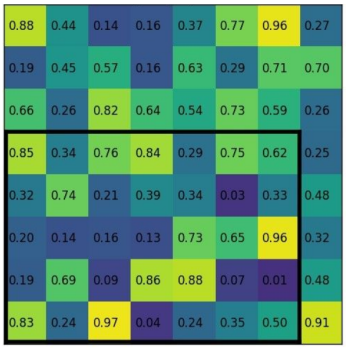

Figura 3.20: Exemplo de rexión de interese con valores $(0, 3, 5, 7)$.

Unha vez obtido isto, o primeiro paso consistía en dividir a nosa rexión de interese en tantas seccións como tamaño quixésemos que tivese a nosa saída, que neste caso era 2×2. Podemos ver a división realizada na Figura 3.21. Notemos que o tamaño da rexión de interese non ten que ser perfectamente divisible polo número de seccións, así, neste caso a nosa rexión de interese é de tamaño 7×5 pero divídese en 4 seccións.

O segundo paso consiste en seleccionar o valor máximo de cada unha destas seccións, estes valores aparecen recollidos na Figura 3.22, que se corresponde coa saída que obteríamos ao aplicarlle esta capa ao mapa de activación da Figura 3.19 e a entrada que se lle pasa á secuencia de capas completamente conectadas.

Tras ver estas capas lembremos que, ao principio, dixemos que a idea básica de Fast R-CNN era modificar unha rede neuronal, que orixinalmente fora adestrada para clasificación de imaxes, co obxectivo

[21] As K clases consieradas máis outra clase para clasificar posibles partes da imaxe que formen parte do fondo da mesma e non dunha clase específica.

[22] O exemplo foi sacado de `https://deepsense.ai/region-of-interest-pooling-explained/`.

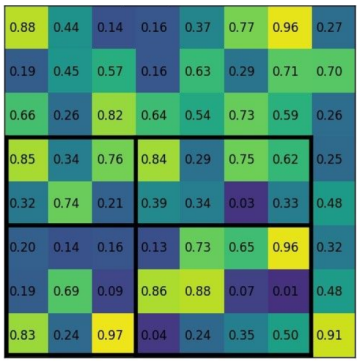

Figura 3.21: Divisón do cadro delimitador en 4 subrexións.

Figura 3.22: Aplicación de *max pooling* a cada subrexión.

de realizar a detección de obxectos. Así pois, unha vez visto o procesamento da entrada, vexamos en que consisten estas modificacións.

Inicialización empregando redes preadestradas

En Girshick (2015) [11] empregáronse tres redes preadestradas de ImageNet: CaffeNet de R-CNN, VGG-CNN-M-1024 e VGG16 as cales poden consultarse respectivamente en Darrel et al (2014) [4], Chatfield et al (2014) [3] e Simonyan et al (2015) [38]. Cada unha destas redes conta con cinco capas de *max pooling* e entre cinco e trece capas convolucionais. Pero, cales son as transformacións que experimentan estas redes preadestradas ao inicializar o Fast R-CNN? Vexámolo.

- En primeiro lugar, a última capa de *max pooling* é substituída pola capa de *RoI pooling* que é configurada establecendo H e W de forma que sexan compatibles coa primeira capa completamente conectada da rede.
- En segundo lugar, a última capa completamente conectada da rede e a capa de softmax son substituídas polas dúas capas de saída descritas anteriormente.
- En terceiro e último lugar, a rede é modificada para recibir dúas entradas: o conxunto de imaxes e o conxunto de rexións de interese para cada unha delas.

Para realizar o adestramento empregouse a propagación cara atrás xunto co descenso de gradiente estocástico, presentemos agora a función de perda empregada.

Función de perda multitarefa

Como xa dixemos, Fast R-CNN ten dúas capas de saída irmás.

A saída da primeira era unha probabilidade discreta por cada rexión de interese,

$$p = (p_0, ..., p_K)$$

calculada sobre as $K+1$ categorías, as K clases consideradas máis unha para o fondo da imaxe. Como sempre, p calcúlase empregando a función softmax sobre as $K+1$ saídas da última capa completamente conectada da rede neuronal.

Mentres tanto, a saída da outra capa consistía nun refinamento dos cadros delimitadores polo que, para cada unha das K posibles clases, obterase unha tupla formada por catro elementos da forma

$$t^k = (t^k_x, t^k_y, t^k_w, t^k_h)$$

onde a clase aparece indexada por k.

Ademais, cada rexión de interese que se emprega no adestramento é etiquetada tanto coa clase á que pertence, u, como cun cadro delimitador v.

A función de perda, L, é empregada en cada unha das rexións de interese etiquetadas para adestrar conxuntamente tanto a clasificación como o refinamento dos cadros delimitadores. Dita función vén definida por

$$L(p, u, t^u, v) = L_{cls}(p, u) + \lambda[u \geq 1]L_{loc}(t^u, v),$$

onde $L_{cls}(p, u) = -\log p_u$ é a función de perda logarítmica para a verdadeira clase u. A segunda función de perda, L_{loc}, defínese sobre unha tupla dos verdadeiros cadros delimitadores da clase u,

$$v = (v_x, v_y, v_w, v_h)$$

e sobre a tupla predita (t^u) para a mesma clase. Esta función de perda defínese como

$$L_{loc} = \sum_{i \in \{x,y,w,h\}} smooth_{L_1}(t^u_i - v_i),$$

onde

$$smooth_{L1}(x) = \begin{cases} 0.5x^2 & |x| < 1 \\ |x| - 0.5 & \text{noutro caso} , \end{cases}$$

e λ é un hiperparámetro que controla a balanza entre as dúas funcións de perda.

Por convención, a clase que se refire ao fondo da imaxe denótase por $u = 0$. Ademais, para as rexións de interese que se corresponden ao fondo da imaxe, non hai unha noción de cadros delimitadores e, polo tanto, a función de perda L_{loc} é ignorada neses casos.

Non presentamos aquí ningún exemplo de aplicación de Fast R-CNN que si se verá na Sección 3.4.5. Máis detalles sobre o adestramento desta rede poden consultarse en Girshick (2015) [11] se ben nós non afondaremos máis nisto pasando a falar dunha mellora de Fast R-CNN: o Faster R-CNN.

3.4.3. Faster R-CNN

Como acabamos de ver, as redes neuronais para a detección de obxectos dependen en gran medida dos algoritmos de propostas de rexións, como o de busca selectiva no caso de Fast R-CNN. Ademais, no caso de Fast R-CNN, a precisión do mesmo depende en gran medida das rexións de interese obtidas.

Intentando mellorar estas debilidades, xurdiu o Faster R-CNN o cal non só mellora os resultados obtidos con Fast R-CNN senón que tamén reduce considerablemente o tempo preciso para adestrar a rede neuronal, de aí o nome de Faster R-CNN pois, en inglés, "faster" significa "máis rápido".

O Faster R-CNN consiste en realizar unha pequena modificación a Fast R-CNN, a cal consiste en cambiar o proceso de proposta de rexións de interese. Así, notemos que Fast R-CNN non realizaba ningunha predición dos cadros delimitadores, exceptuando os refinamentos que xa comentamos. Pois ben, a novidade de Faster R-CNN é que introduce unha rede de proposta de rexións de interese (*Region Proposal Network*), á cal nos referiremos por RNP.

Esta, toma como entrada unha imaxe de calquera tamaño e produce como saída un conxunto de cadros delimitadores, onde cada un deles leva asociada unha puntuación de obxecto. Esta puntuación non é outra cousa que a probabilidade de pertenza ben ao conxunto de clases considerado ou ben ao fondo da imaxe.

A RPN é construída xusto antes das capas convolucionais da rede que se emprega para a detección en Fast R-CNN. Para isto, o que se fai é engadir varias capas convolucionais que son adestradas para calcular, simultaneamente, tanto os cadros delimitadores como as puntuación para cada clase. Para obter estes cálculos emprégase un método cunha fiestra corrediza (*sliding window*), a cal vai tomando cada unha das partes da imaxe e vai creando diferentes cadros delimitadores, de forma que o cunxunto deles conteña toda a información relativa á imaxe. Con estes cadros, exemplos dos cales podemos atopar na Figura 3.23[23], constrúese a proposta de rexión de xeito automatizado que logo se pasará á rede neuronal de detección, a cal é exactamente igual á que se describiu en Fast-RCNN.

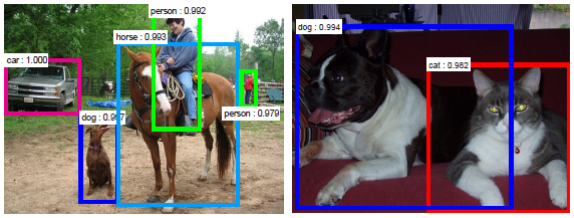

Figura 3.23: Exemplo de saídas obtidas empregando RPN.

Unha vez visto isto, xorde unha nova pregunta: como podemos unificar a RNP coa rede específica para realizar a detección, que é a mesma que se empregaba en Fast-RCNN?

Para realizar a unificación destas redes nunha única, ámbalas dúas comparten un con conxunto de capas convolucionais con mecanismos de atención, de forma que as compoñentes da RPN dinlle á rede de detección cara onde mirar. Máis información sobre como se realiza a unificación así como o adestramento que se realiza en Faster R-CNN pode consultarse en Ren et al (2016) [33].

3.4.4. Avances: detección de obxectos en tempo real

A detección de imaxes segue a ser un dos campos máis estudados a día de hoxe dentro da IA. Un dos últimos avances vén da man de YOLO: Detección de obxectos en tempo real. Este modelo, cuxa última versión acaba de saír, permite realizar a detección de obxectos en tempo real.

YOLO presenta un enfoque moi distinto aos que mencionamos ata o de agora, se ben moi sinxelo, pois trátase dunha única rede convolucional, a cal, tras procesar a imaxe que se lle pasa como entrada, realiza a súa clasificación na clase correspondente. YOLO (*You Only Look Once*) recibe este nome

[23]Tomada de Ren et al (2016) [33].

porque outras das súas peculiaridades é que se ben nos métodos anteriores había que realizar varias iteracións aquí a imaxe procésase unha soa vez obtendo directamente a probabilidade de pertenza a unha determinada clase, pode consultarse máis literatura en Redmon et al (2016) [32].

3.5. Exemplo práctico

Tras ver ambas as dúas arquitecturas, poñamos un exemplo práctico e de xoguete, tomado de Kranen et al (2017) [21], para ver os resultados que se obterían con Fast R-CNN. Para isto, empregaremos como conxunto de datos un conxunto de 25 imaxes dunha neveira de tamaño 850×850 píxeles, referirémonos a este conxunto de datos como *Grocery*.

Nestas imaxes, algunhas das cales poden verse na Figura 3.24, poden verse diferentes alimentos e bebidas no interior dunha neveira: cogombros envinagrados, zume, cartóns de leite, un bote de salsa de tomate, cebola, tabasco, pemento, tomate, aguacate, auga, laranxas, unha botella de champaña, mostaza, unha caixa de ovos, manteiga e iogur. Polo tanto, o problema consistirá en identificar os obxectos presentes nestas imaxes e clasificalos nunha das 16 clases que vimos de ver.

Figura 3.24: Exemplos do conxunto de datos *Grocery*.

Para a implementación deste método empregouse a linguaxe de programación de Python facendo uso dunha das librerías máis empregadas no que respecta á aprendizaxe profunda, `CNTK` (*Microsoft Cognitive Toolkit*), empregando como guía o material dispoñible en Kranen et al (2017) [21].

En primeiro lugar, dividimos o noso conxunto en dúas mostras: a de adestramento, formada por un 75 % das imaxes, e a de validación, formado polo 25 % restante.

Para este exemplo consideramos 200 propostas de rexión de interese, establecendo o limiar para a superposición en 0.5. Notemos que canto máis baixo sexa esta limiar, maior será o número de cadros delimitadores propostos. Ademais, o número máximo de rexións de interese asociadas ao fondo da imaxe foi fixado a 50. Estas *RoI* foron calculadas na primeira das iteracións empregando a implementación de busca selectiva do paquete `dlib`, eliminando aquelas rexións cunha área inferior a 9 píxeles.

Unha vez xeradas as rexións de interese, e antes de comezar o adestramento, normalizamos as imaxes de entrada, restando para iso en cada dimensión unha constante que foi fixada a 114. O obxectivo desta normalización é evitar que as correccións realizadas pola taxa de aprendizaxe en cada unha das dimensións sexan desproporcionais.

Unha vez feita a normalización, comezamos o adestramento do noso modelo sobre a mostra reservada para tal fin. Para o adestramento empregamos como arquitectura base a rede AlexNet coas modificacións pertinentes que xa comentamos na Sección 3.4.2. Durante o mesmo, as capas convolucionais de AlexNet foron adestradas se ben poderían ter quedado fixas. Fixar os pesos das capas convolucionais

quere dicir que os pesos son tomados do modelo base preadestrado e non son modificados durante o adestramento. Vexamos algúns dos parámetros empregados durante o mesmo:

- Taxa de aprendizaxe: a taxa de aprendizaxe foi inicializada a un valor grande, 0.001, e foi diminuíndo por un factor de 10 sendo o valor máis pequeno empregado 0.00001.
- *momentum* : o *momentum* foi fixado a 0.9.
- Regularización: para a regularización empregouse a regularización L_2 fixando $\lambda = 0.0005$
- Épocas e tamaño de *minibatch*: para o adestramento empregáronse un total de 20 épocas e un tamaño de *minibatch* igual a 4, debido ao reducido volume de datos dos que dispomos.

Unha vez finalizado o adestramento do modelo, o cal se realizou empregando o algoritmo de propagación cara atrás xunto co descenso de gradiente estocástico, procedeuse á avaliación do mesmo sobre o conxunto de validación.

A calidade deste pode ser medida empregando diferentes criterios como a precisión, *recall*, *accuracy*, ROC, etc. Unha métrica comunmente empregada no recoñecemento de obxectos é a medición da precisión media (AP) obtida para cada unha das clases. A media da precisión media (mAP), valga a redundancia, é calculada facendo a media da AP de tódalas clases, a cal se define como a precisión media nun conxunto de once niveis igualmente espaciados $[0, 0.1, ..., 1]$:

$$AP = \frac{1}{11} \cdot \sum_{r \in \{0, 0.1, ..., 1\}} p_{interp}(r)$$

con

$$p_{interp}(r) = \max_{\tilde{r}:\tilde{r} \geq r} p(\tilde{r})$$

onde $p(\tilde{r})$ é a precisión medida en $\tilde{r}$.

Os resultados obtidos tras adestrar con 200 rexións de interese o conxunto de datos *Grocery* empregando a arquitectura de AlexNet como modelo base, aparecen recollidos na Táboa 3.1 que se achega a continuación.

Vemos nesta táboa (ver Táboa 3.1) que o noso modelo parece a estar a detectar e clasificar correctamente os obxectos atopados para todas as clases. De feito, para a maioría das clases obtívose unha clasificación perfecta, AP igual a 1. As peores clasificacións obtivéronse para a clase do bote de salsa de tomate, para a cal obtivemos unha clasificación correcta do 66.67 %, e para a clase da auga, cuxa clasificación correcta é do 50 %, o que fixo que a clasificación correcta fose, finalmente, do 94.79 % (mAP).

Notemos ademais que, se ben só se realizou o exemplo práctico para Fast R-CNN, polo argumentado anteriormente, e debido a que a diferenza entre ambos modelos radica na forma de obter as propostas de rexións de interese, cabería esperar uns resultados moi parecidos con Faster R-CNN se ben a execución deste sería máis rápida.

Clase	AP
Cogombros envinagrados	1.0000
Manteiga	1.0000
Iogur	1.0000
Caixa de ovos	1.0000
Mostaza	1.0000
Champáña	1.0000
Laranxa	1.0000
Auga	0.5000
Aguacate	1.0000
Tomate	1.0000
Pemento	1.0000
Tabasco	1.0000
Cebola	1.0000
Leite	1.0000
Bote de salsa de tomate	0.6667
Zume	1.0000

Cadro 3.1: Precisión media (AP) para as diferentes clases do conxunto de datos *Grocery*.

Capítulo 4

Aplicación a imaxes mamográficas

> "*Rather than thinking in terms of human vs. machine, we want to focus on how human gifts such as creativity, empathy, emotion, physicality and insight can be mixed with powerful AI computation [...] to help move society forward.* "
>
> Satya Nadella en «Hit Refresh».

Neste capítulo imos darlle aplicación aos coñecementos que vimos de ver nos dous capítulos anteriores. Para isto, centrarémonos na análise de imaxes mamográficas co fin de axudar na análise clínica do cancro de mama mediante a clasificación das mamas en función da súa composición e mediante a avaliación de riscos.

Para ver como a aprendizaxe profunda pode ser de utilidade á hora de analizar imaxes mamográficas comezaremos cunha breve sección de motivación. Na segunda sección, expoñeremos algúns dos traballos existente sobre o tema así como as debilidades dos mesmos. Na terceira sección expoñemos o obxectivo que pretendemos conseguir nós mediante a análise das mamografías xunto cos problemas aos que nos afrontamos. E, xa finalmente, presentamos unha sección de traballo futuro para asentar as guías a seguir unha vez os problemas comentados na sección anterior sexan resoltos.

4.1. Motivación

Os avances recentes que vimos de ver no marco da aprendizaxe profunda para a detección de obxectos en imaxes causou un grande interese pola súa aplicación en imaxes médicas como son as imaxes mamográficas, técnica empregada na detección do cancro de mama.

O cancro de mama é o tipo de cancro máis común nas mulleres e o segundo tipo de cancro máis común de todos, sendo unha das principais causas de morte das mulleres a nivel global. Deste xeito, no ano 2012 diagnosticáronse aproximadamente un total de 1.7 millóns de novos casos no mundo segundo o World Cancer Research Fund International (2012) [44]. Estímase tamén que preto de 232 mil mulleres foron diagnosticadas con cancro de mama nos Estados Unidos de América no ano 2015 e, ao redor de 40 mil, morreron debido a esta enfermidade. Espérase que estes números segan a medrar, esperando que as diagnoses de calquera tipo de cancro aumenten dos 14 millóns rexistrados en 2012 a aproximadamente 20 millóns nas próximas dúas décadas, como pode consultarse en International Agency for Cancer Research (2014) [17].

Pese a estes alarmantes números, as probabilidades de curación do cancro en estados iniciais son practicamente do 100 %. De feito, grazas á realización de campañas de diagnóstico temperán para a detección de cancro de mama, a mortalidade viuse reducida de forma significativa, polo menos cando se realizan no grupo de maior incidencia: mulleres de máis de 50 anos.

As imaxes mamográficas, son un método de xeración de imaxe empregando raios X que se emprega para examinar os senos das pacientes. A pesares de que estas imaxes axudan a reducir en gran medida a mortalidade de cancro de mama son tamén as causantes dun tema de intenso debate polos danos que leva asociados a exposición á radiación. Ademais destes danos, atópase outros como pode ser a ansiedade que pode carretar un diagnóstico positivo ou o temor a que o sexa, así como os falsos positivos.

Entre o 10 e o 15 % de mamografías non serven por si soas para descartar a presenza, ou non, do cancro o que fai necesaria a realización de entre 3.3 e 4.5 % exames médicos adicionais como se pode ver en Tosteson et al (2014) [41]. Así, a gran maioría das mulleres teñen que someterse a ditos exames que, en moitas ocasións, inclúen novas mamografías ou técnicas de ultrasóns para aclaración. Finalmente, a maioría destes exames resultan desvelar un seno normal.

Na práctica, obter un diagnóstico baseándose soamente na mamografía é moi difícil polo que, en moitos casos, antes de que un doutor poida facer unha diagnose requírese a realización dunha biopsia. Esta é unha operación cirúrxica e, como tal, leva riscos asociados como poden ser dores crónicas e infeccións. Ademais, obter cita para unha biopsia e esperar polos resultados do laboratorio pode levar un tempo ata que se realiza a diagnose, o que pode darlle ao cancro tempo suficiente para ter efectos irreversibles na saúde da paciente.

Vexamos algunhas cifras relacionadas coa biopsia. Deste xeito, entre un 1 % e un 2 % das pacientes que teñen unha mamografía anormal teñen que realizar unha biopsia e, entre aquelas que a realizan, tan só un 20-40 % resultan nun diagnóstico de cancro, Kopans D (2015) [23].

Así e todo, pese aos riscos asociados, as mamografías son unhas das técnicas máis efectivas capaces de detectar lesións nas etapas máis iniciais da enfermidade o que reduce a mortalidade ata un 30 %, pode consultarse máis literatura en Duffy et al (2002) [6] e Jernal et al (2009) [18]. Estas técnicas poden detectar lesións na mama ata dous anos antes de que sexan palpables e antes de que teñan invadido os nodos linfáticos ou se teñan expandido a outros órganos. Cando o tumor se detecta nestas etapas iniciais é posible aplicar un tratamento menos agresivo e invasivo deixando unha menor secuela tanto física como psicolóxica na paciente.

Todo isto, xunto cos custos asociados ás distintas probas, danos unha visión do porqué a aprendizaxe profunda, especialmente a detección de obxectos, pode ser unha gran aliada para axilizar o proceso de diagnóstico. É por isto que ao longo deste capítulo falaremos da viabilidade destas técnicas para axudar na análise clínica do cancro de mama mediante a clasificación das mesmas en función da súa composición e mediante a avaliación de riscos.

4.2. Traballos relacionados

A maioría dos traballos analizados empregan un conxunto de datos moi limitados como poden consultarse en Zhou et al (2016) [45], Geras et al (2017) [9] e Ribli et al (2018) [34]. Sendo un dos conxuntos máis frecuentes o conxunto de datos mini-MIAS: formado por 322 imaxes as cales foron reducidas a un tamaño de 1024×1024. Isto non só constitúe un conxunto de datos limitados sobre o cal adestrar e validar os nosos modelos senón que tamén é bastante inconsistente no que respecta á variabilidade.

A información aportada polo conxunto de datos pode resultar vital para incrementar tanto a (*accuracy*) como a (*recall*) do modelo, pero, ao mesmo tempo poden aumentar exponencialmente os tempos

necesarios para realizar o adestramento. Isto constitúe outra oportunidade para probar a eficacia do noso modelo, pois a maioría dos traballos analizados implementaron os cálculos en servidores dun único nodo sen o soporte de procesado da GPU, se ben aqueles que si empregaron este tipo de soporte obtiveron unha maior rapidez no adestramento, como pode consultarse en Zhou (2016) [45]. Aproveitar as máquinas virtuais habilitadas para GPU en Azure é unha forma poderosa, á vez que económica, de xerar novas oportunidades para a análise clínica de imaxes dixitais.

Ademais das plataformas de hardware e os marcos computacionais para o modelado de rede, tamén se observou que, salvo un (ver Ribli et al (2018) [34]), ningún dos métodos publicados ata o de agora empregaba Fast R-CNN, nin calquera das súas variantes, o cal pode axudar de xeito masivo coa redución dos tempos de adestramento así como para aumentar a precisión do noso modelo. Ademais, o conxunto de datos á nosa disposición, próximo a 1 millón, constitúe un mellor conxunto de datos que o empregado en Ribli (2018) [34] polo que se espera a obtención de mellores resultados.

4.3. Obxectivo

As mamografías son clasificadas de acordo ao BI-RADS (do inglés *Breast Image Reporting and Data System*), unha clasificación desenrolada en 1993 para estandarizar os informes de mamografía co fin de minimizar o risco de interpretación dos mesmos e facilitar a comparación dos resultados para futuros estudos clínicos. Esta clasificación ocúpase de clasificar a mama segundo a composición da mesma e segundo a anomalía que presenta, máis información sobre esta clasificación pode consularse en Eberl et al (2006) [7].

Actualmente, o sistema de detección asistida por computadora (DAC) é empregado principalmente en detección de carcinoma de mama como unha ferramenta de traballo para a detección de lesións mamarias. Sen embargo, a pesar da súa utilidade, non é capaz de asignar un grado de sospeita. Isto conleva que a posibilidade de realizar un diagnóstico de falsos positivos sega a estar presente.

Aproveitamos isto para introducir os estatísticos máis relevantes para o problema en cuestión. Así, cando se está avaliar o modelo en conxuntos de datos tan sensibles, a puntuación de (*recall*) é máis importante que a precisión. Isto é obvio posto que o principal obxectivo do clasificador é identificar anormalidades malignas. Dende unha perspectiva estatística isto significaría que estamos centrados en minimizar o número de falsos negativos. Dende o punto de vista médico, diagnosticar de forma errada un caso negativo tradúcese no tipo de diagnose que pode conlevar á morte do paciente.

Por unha banda, o que pretendemos conseguir coa análise de mamografías é a clasificación das mamas segundo a súa densidade para identificar de forma obxectiva aquelas pacientes que puideran precisar unha proba de ultrasón como parte complementaria da mamografía. Por outra banda, introducindo a descrición de BI-RADS sería posible non só detectar a patoloxía, pero ademais, dependendo de que foi detectado, asignar unha categoría de risco, o que representaría un paso máis alá na diagnose realizada actualmente polo DAC.

Ademais, un experto no campo pode clasificar axeitadamente unha mamografía en, aproximadamente, 15 minutos. Polo tanto, un sistema automatizado que puidese realizar unha clasificación inicial poderá ser de axuda tanto á hora de minimizar os tempos como os potenciais erros que se puidesen cometer.

Deste xeito, o obxectivo ao que nos enfrontamos é ver como os avances na aprendizaxe profunda que vimos de ver abren novas vías para a diagnose do cancro de mama mediante a extracción da seguinte información a partir da mamografía:

- Composición do tecido da mama. A saída do sistema será a clasificación nunha das seguintes composicións, ver Mandelson et al (2000) [28]:

  - As mamas son completamente graxas
  - As mamas presenta áreas dispersas de densidade fibro glandular
  - As mamas son densas de forma heteroxénea
  - As mamas son extremadamente densas

- Avaliación de riscos. A saída será a clasificación nun dos seguintes niveis de risco.

  0 Incompleto

  1 Negativo

  2 Benigno

  3 Probablemente benigno

  4 Sospeitoso

  5 Probablemente maligna

  6 Biopsia coñecida, probada maligna.

Para adestrar ás nosas redes e realizar a clasificación que acabamos de ver, precisamos dispor dun conxunto de imaxes mamográficas amplo onde en cada mamografía apareza localizada e clasificada segundo o grao de confianza cada unha das anomalías presentes na mamografía. Estas anomalías clasificaranse de acordo coa seguinte lista:

- Calcificación
- Masas circunscritas ben definidas
- Masa espiculada
- Outras masas con tecido enfermo
- Distorsión arquitectónica
- Asimetría
- Normal

Pola lei de protección de datos, non é posible facer públicas as imaxes do noso conxunto de datos. Se ben si presentamos, para facernos a idea de como son estas imaxes, aquelas pertencentes ao conxunto de datos público mini Mias (ver Suckling et al (1994) [40]).

Así, na Figura 4.1 podemos ver as mamografías correspondentes ás dúas mamas dunha paciente. Ditas mamas son de composición graxa e ámbalas dúas presentan masas circunscritas benignas. Do mesmo xeito, na Figura 4.2 amósanse as mamografías correspondentes a mamas de composición graxa, detectándose na correspondente á mama dereita masas circunscritas ben definidas malignas.

A localización destas anomalías vén especificada no conxunto de datos mediante as coordenadas do centro e o radio do círculo que circunscribe á mesma. Así, por exemplo, na Figura 4.1 (esquerda) a anomalía estaría circunscrita por un círculo que ten como centro as coordenadas $(535, 425)$ e un radio de 33 píxeles.

Como vimos na sección de traballos relacionados, traballar en conxuntos de datos moi limitados non proporciona tan bos resultados como nos gustaría. Nun primeiro lugar, isto non supoñía un problema para nós, pois dispoñíamos dun conxunto balanceado de imaxes de dimensións $5000 \times 4000 \times 16$ bits de profundidade, o cal constitúe un conxunto de datos de alto valor.

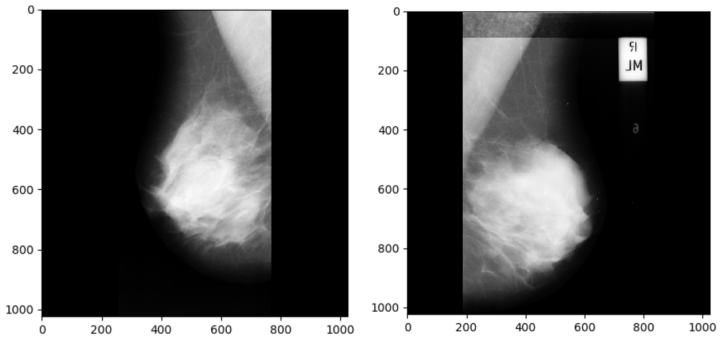

Figura 4.1: Mamografías da mama esquerda (esquerda) e dereita (dereita) dunha paciente con áreas dispersas de densidade fibro glandular. Anomalía: Masas circunscritas ben definidas (benigna), presentes tanto na mama esquerda como dereita.

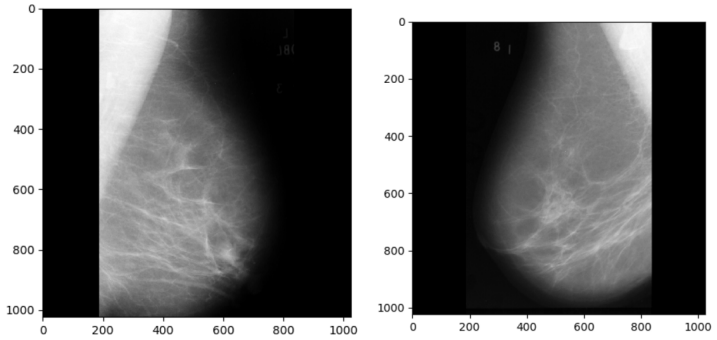

Figura 4.2: Mamografías da mama esquerda (esquerda) e dereita (dereita) dunha paciente de composición graxa. Anomalía: a mama esquerda non presenta ningunha anomalía mentres que a mama dereita presenta masas circunscritas ben definidas (malignas).

Sen embargo, á hora de realizar unha primeira análise dos datos, vimos que as imaxes non estaban etiquetadas, isto é, as anomalías non estaban correctamente clasificadas e localizadas o que imposibilitou completamente a realización do proxecto, que tivo que ser pausado ata que un equipo de expertos no ámbito do cancro da mama realice este etiquetado.

Isto sírvenos para darnos conta de que no mundo real os datos non son de xoguete e que problemas como este poden xurdir, facendo que teñamos que deternos por completo ata que sexan arranxados. De tódolos xeitos, engádese a continuación unha sección sobre o traballo futuro a realizar unha vez teñamos o conxunto de datos propiamente etiquetado.

4.4. Traballo Futuro

Nesta sección imos ver como procederíamos para acadar o obxectivo exposto unha vez tivésemos o conxunto de datos preciso para tal fin. Vexamos entón os pasos a seguir. Así, o enfoque suxerido baséase, como xa dixemos, na aprendizaxe profunda encadeando os modelos que enumeramos a continuación.

En primeiro lugar, empregaremos redes convolucionais como AlexNet ou VGG para realizar a clasificación da mamografía segundo a vista que represente, a cal pode ser cráneo-caudal ou oblicua mediolateral. Seguidamente, empregaremos de novo redes convolucionais para facilitar a comprensión da composición do tecido da mama (graxo, denso...). En terceiro lugar, empregaremos unha rede Faster-RCNN tanto para a localización da área específica onde se atopan as diferentes anormalidades como para a súa clasificación. Para comprender mellor esta parte presentamos a Figura 4.3[1] onde podemos ver unha esquematización da parte correspondente a Faster R-CNN. Así, vemos que a imaxe é procesada a través das redes convolucionais e distintas rexións de interese son propostas, estas serán empregadas seguidamente para realizar a clasificación, emitindo como saída a imaxe de entrada coa anormalidade circunscrita nun cadro delimitador e a súa clasificación. Finalmente, empregaremos un clasificador denso para xerar a clasificación BI-RADS a partir dos datos extraídos nos pasos anteriores.

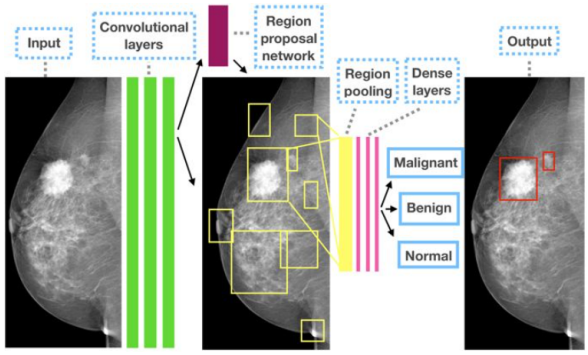

Figura 4.3: Esquema do modelo Faster R-CNN para aplicación a mamografías.[2]

Co obxectivo de acelerar o proceso de adestramento engadiremos un *autoencoder* como paso inicial do proceso para lograr unha redución de dimensionalidade significativa. Este *autoencoder* deberá ser probado axeitadamente xa que, noutro caso, podería xerar unha perda significativa de información durante o proceso de comprensión.

Durante o adestramento do modelo engadirase tanto ruído como transformacións ao conxunto de adestramento para axudar a reducir o sobreaxuste dos datos que poida producirse ao traballar coas mesmas imaxes dixitais en todo o conxunto de datos.

[1] Tomada de Ribli (2018) [34].

Estes modelos serán implementados e adestrados empregando a libraría CNTK de Python nun entorno de Microsoft Azure que conte con múltiples nodos e GPUs. Con isto, preténdese aproveitar o enorme repositorio de imaxes dispoñibles para adestrar os modelos nun período de tempo razoable e sen a necesidade de aprovisionamento dunha potencia de cálculo masiva que non se tería empregado unha vez que o adestramento tivese rematado.

Capítulo 5

Conclusións

Ao longo deste traballo fixémonos eco do interese que está suscitar a Intelixencia Artificial dende estes últimos anos, tanto no mundo industrial como na comunidade investigadora, centrándonos especialmente no campo da aprendizaxe profunda.

Para isto, fixemos unha revisión da literatura existente, dando, en primeiro lugar, unha breve introdución histórica na cal se explicou a evolución que foi sufrindo este campo. En segundo lugar, explicamos os conceptos básicos e fundamentais das redes neuronais, imprescindibles para a comprensión do tema. Seguidamente, presentamos as redes neuronais convolucionais, un tipo de redes amplamente empregadas para a análise de imaxes e vimos tres das arquitecturas máis empregadas: R-CNN, Fast R-CNN e Faster R-CNN. Finalmente, vimos unha das aplicacións da análise de imaxes, a detección e clasificación de obxectos, a cal foi aplicada á análise de imaxes mamográficas.

Este exemplo, se ben non nos permitiu poñer en práctica os coñecementos adquiridos pola falta dun conxunto de datos que puidese ser empregado, si nos permitiu ver algún dos problemas cos que nos podemos atopar cando nos enfrontamos a un experimento real. Por último, comentar que, grazas a este exemplo, tamén puidemos comprobar en primeira persoa o activa que está a área da aprendizaxe profunda posto que, se ben nun principio, á hora de facer a proposta de proxecto, non se atopara ningún outro método que empregara Faster R-CNN para análise de imaxes mamográficas, a data de marzo de 2018 este xa se tiña implementado.

Para concluír, gustaríame incluír tamén neste apartado final as conclusións obtidas tras a realización das prácticas en Plain Concepts, por ser esta a miña primeira experiencia laboral no ámbito da Estatística. Así pois, considero que as prácticas foron de grande utilidade para facerme medrar tanto profesional como persoalmente, ao terme que enfrontar a distintos retos completamente novos para min, os cales me permitiron poñer en práctica moitos dos coñecementos adquiridos ao longo do máster.

Bibliografía

[1] Breiman L (2001) Statistic Modeling: The two cultures. Statistical Science, Vol. 16 (3), p. 199-231.

[2] Chavan A, Bendale D, Shimpi R, Vikhar P (2016) Object Detection and Recognition in Images. International Jounal of Computing and Technology, Vol. 3 (3), p. 148-151.

[3] Chatfield K, Simonyan A, Vedaldi A, Zisserman A (2014) Return of the devil in the details: Delving deep into convolutional nets. arXiv: 1405.3531 [cs.CV].

[4] Darrel T, Donahue J, Girshick R, Malik J (2014) Region based convolutional networks for accurate object detection and segmentation. arXiv: 1311.2524 [cs.CV].

[5] DasGupta B, Schnitger G (1993) The Power of Approximating: A Comparison of Activation Functions. Advances in Neural Information Processing Systems, 5, pp. 615-622.

[6] Duffy SW et al (2002) The impact of organized mammography service screening on breast carcinoma mortality in seven swedish counties. Cancer Vol. 95 (3), p. 458-469.

[7] Eberl MM, Fox CH, Edge SB et al (2006) BI-RADS Classification for Management of Abnormal Mammograms. Journal of the American Board of Family Medicine. Vol 19 (2), p. 161-164.

[8] Gareth J, Witten D, Hastie T, Tibshirani R (2013) An Introduction to Statistical Learning with Applications in R. Springer, p. 176.

[9] Geras K, Wolfson S, Kim S, Moy L, Cho K (2017) High-Resolution Breast Cancer Screening with Multi-View Deep Convolutional Neural Networks. arXiv: 1703.07047v1 [cs.CV].

[10] Géron, A (2017) Hands-On Machine Learning with Scikit-Learn Tensorflow. OâReilly (1st Ed)

[11] Girshick R (2015) Fast R-CNN. arXiv: 1504.08083v2 [cs.CV]

[12] Girshick R, Donahue J, Darrell T, Malik J (2014) Rich feature hierarchies for accurate object detection and semantic segmentation. arXiv: 1311.2524v5 [cs.CV]

[13] Goodfellow IJ, Warde-Farley D, Mirza M, Courville A, Bengio Y (2013) Maxout Networks. arXiv: 1302.4389v4 [stat.ML].

[14] He K, Zhang X, Ren S, Sun J (2015) Delving Deep into Rectifiers: Surprassing Human-Level Performance on ImageNet Classification. arXiv: 1502.01852v1 [cs.CV].

[15] Hebb DO, Wiley J, Sons Inc (1949) A neuropsychological theory. Nova York.

[16] Hutter F, Loshchilov I, (2017) SGDR: Stochastic Gradient Descent with Warm Restarts. arXiv: 1608.03983v5 [cs.LG]

[17] International Agency for Cancer Research (2014) World Cancer Report 2014.

[18] Jernal A et al (2009) Cancer Statistics. CA: A Cancer Jounal for Clinics, Vol 59 (4), p. 225-249.

[19] Kandel R, Schwartz JH, Jessel TM (2000) Principles of Neural Science. McGraw-Hill.

[20] Karlik B, Olgac AV Performance Analysis of Various (2011) Activation Functions in Generalized MLP Architectures of Neural Networks. Dispoñible en

`https://www.researchgate.net/publication/228813985_Performance_Analysis_of_Various_Activation_Functions_in_Generalized_MLP_Architectures_of_Neural_Networks?enrichId=rgreq-47e081b1a3694786503d418a424b2fcd-XXX&enrichSource=Y292ZXJQYWdlOzIyODgxMzk4NTtBUzoxMDM1NDgzMDE4NzMxNjdAMTQwMTY5OTI0Nzk1OQ\%3D\%3D&el=1_x_2&_esc=publicationCoverPdf`. Accedido o 19 de abril de 2018.

Accedido o 3 de abril de 2018.

[21] Kranen P, Hillebrand M, Hopkins T,... Karampatziakis N, (2017) Object detection using Fast R-CNN. Dispoñible en `https://docs.microsoft.com/en-us/cognitive-toolkit/object-detection-using-fast-r-cnn`. Accedido o 28 de marzo de 2018.

[22] Krizhevsky A, Sutskever I, Hinton E (2012) ImageNet Classification with Deep Convolutional Neural Networks. Communications of the ACM, Vol. 60 (6), p. 84-90.

[23] Kopans D (2015) An open letter to panels that are deciding guidelines for breast cancer screening. Breast Cancer Res Treat, Vol. 151 (1), p. 19-25.

[24] Kuhn M, Johnson K (2013) Applied Predictive Modeling. Springer, p. 67.

[25] Li F, Karpathy A (2017) Convolutional Neural Networks for Visual Recognition. Course Notes. Ver `https://http://cs231n.stanford.edu/`.

[26] LeCun Y, Bengio Y, Hinton G (2015) Deep Learning. Nature, Vol. 521, p. 436-444.

[27] Maas, AL, Hannun, AY, Ng, A (2014) Rectifier Nonlinearities Improve Neural Network Acoustinc Models. Computer Science Department, Stanford University. `http://web.stanford.edu/~awni/papers/ReLU_hybrid_icml2013_final.pdf` accedido 5 de Marzo do 2018.

[28] Mandelson MT, Oestreicher N, Porter PL et al (2000) Breast Density as a Predictor of Mammographic Detection: Comparison of Interval -and Screen-Detected Cancers. JNCI: Journal of the National Cancer Institute, Vol. 92 (13), p. 1081-1087.

[29] McCulloh W, Pitts W (1943) A logical calculus of the ideas immanent to nervous activity. Bulletin of Mathematical Biophysics, Vol. 5, p. 115-133.

[30] Patterson J, Gibson A (2017) Deep Learning, A Practitionerâs Approach. OâReilly (1st Ed)

[31] Ramachandram P, Zoph B, V.Le Q (2017) Searching for activation functions. arXiv: 1710.05941v2 [cs.NE].

[32] Redmon J, Divvala S, Girshick R, Farhadi A (2016) You Only Look Once: Unified, Real-Time Object Detection. arXiv: 1506.02640v5 [cs.CV].

[33] Ren S, He K, Girshick R, Sun J (2016) Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks. arXiv: 1506.01497v3 [cs.CV].

[34] Ribli D, Horváth A, Unger Z, Pollner P, Csabai I (2018). Detecting and classifying lesions in mammograms with Deep Learning. Scientific Reports, Vol. 8, artigo número: 4165.

[35] Rosenblatt F (1957) The perceptron, a perceiving and recognizing automaton Project Para. Cornell Aeronautical Laboratory.

[36] Rumelhart DE, Hinton GE, Williams RJ (1986) Learning representations by back-propatation errors. Nature, 323, 533-536.

[37] Russel S, Norvig P (2009) Artificial Intelligence: A Modern Approach. Pearson Educated Limited, p. 709.

[38] Simonyan K, Zisserman A (2015) Very deep convolutional networks for large-scale image recognition. arXiv: 1409.1556 [cs.CV]

[39] Stahl, S (2013) Structure and Functions of Neurons. En: Stahl, Stephen M (2013) Stahl's Essential Psychopharmacology: Neuroscientific basis and Practical Applications. Cambridge University Press, p. 1-10.

[40] Suckling J et al (1994): The Mammographic Image Analysis Society Digital Mammogram Database Exerpta Medica. International Congress Series 1069 pp375-378.

[41] Tosteson A et al (2014) Consecuences of false-positive screening mammograms. JAMA internal medicine Vol 174 (6), p. 954-961.

[42] Webos, P (1974) Beyond Regression: New Tools for Prediction and Analysis in the Behavioral Sciences. Tesis, Harvad University.

[43] Widrow B, Lehr M (1990) 30 years of adaptive neural networks: perceptron, madaline, and backpropagation. Proceedings of the IEEE, Vol. 78 (9), p. 1415-1442.

[44] World Cancer Research Fund International (2012) `https://www.wcrf.org/int/cancer-facts-figures/data-specific-cancers/breast-cancer-statistics`. Accedido o 1 de maio de 2018.

[45] Zhou H, Zaninovich Y, Gregory C (2016) Mammogram Classification Using Convolutional Neural Networks. Dispoñible en `http://chrisgregory.me/projects/papers/cnn-mammogram.pdf`. Accedido o 17 de abril de 2018.