**UNIVERSIDAD AUTONOMA DE MADRID**

**ESCUELA POLITECNICA SUPERIOR**

** **

**Grado en Ingeniería Informática**

**TRABAJO FIN DE GRADO**

**APRENDIZAJE NO-SUPERVISADO CON MODELOS GENERATIVOS PROFUNDOS**

**Fernando Arribas Jara**

**Tutor: Daniel Hernández Lobato**

**Enero 2018**

**APRENDIZAJE NO-SUPERVISADO DE MODELOS GENERATIVOS PROFUNDOS**

AUTOR: Fernando Arribas Jara

TUTOR: Daniel Hernández Lobato

Escuela Politécnica Superior

Universidad Autónoma de Madrid

Enero de 2018

**Resumen (castellano)**

Este Trabajo Fin de grado engloba uno de los temas con más incandescencia del panorama actual, como es la Inteligencia artificial (IA) y todas sus vertientes. En este trabajo se focaliza sobre el aprendizaje automático que es una rama de la IA que tiene como propositico que las maquinas aprendan por si solas.

Más concretamente en este trabajo se abordan métodos de inferencias aproximada y redes neuronales para labores de aprendizaje no supervisado.

Las aplicaciones pueden ser muy diversas ya sea como mecanismos reconstrucción de imágenes, como extractor de características de alto nivel que detallan o describen de manera muy precisa los datos observados o para tareas generativas.

**Abstract (English)**

This Bachelor Thesis… (250-500 words).

**Palabras clave (castellano)**

Palabra 1, Palabra 2,…

***Agradecimientos***

**INDICE DE CONTENIDOS**

[1 Introducción 1](#_Toc175400053)

[1.1 Motivación 1](#_Toc175400054)

[1.2 Objetivos 1](#_Toc175400055)

[1.3 Organización de la memoria 1](#_Toc175400056)

[2 Estado del arte 3](#_Toc175400057)

[2.1 Subsección 3](#_Toc175400058)

[2.1.1 Subsubsección 3](#_Toc175400059)

[3 Diseño 5](#_Toc175400060)

[3.1 Subsección 5](#_Toc175400061)

[3.1.1 Subsubsección 5](#_Toc175400062)

[4 Desarrollo 7](#_Toc175400063)

[4.1 Subsección 7](#_Toc175400064)

[4.1.1 Subsubsección 7](#_Toc175400065)

[5 Integración, pruebas y resultados 9](#_Toc175400066)

[6 Conclusiones y trabajo futuro 9](#_Toc175400067)

[6.1 Conclusiones 9](#_Toc175400068)

[6.2 Trabajo futuro 9](#_Toc175400069)

[Referencias 11](#_Toc175400070)

[Glosario 13](#_Toc175400071)

[Anexos I](#_Toc175400072)

[A Manual de instalación I](#_Toc175400073)

[B Manual del programador III](#_Toc175400074)

[C Anexo … V](#_Toc175400075)

**INDICE DE FIGURAS**

[Figura 1: Nube de palabras realcionadas con IA 3](#Figura1)

[Figura 2: Deep Learning vs Metodos de aprendizajes antiguos 3](#Figura2)

[Figura 3: Reconstrucción de imágenes con autoencoders 3](#Figura3)

[Figura 4: Analisi de imágenes médicas 3](#Figura4)

[Figura 5: Cronología de redes neuronales 3](#Figura5)

[Figura 6: Neurona Biológica 3](#Figura6)

[Figura 7: Analogía entre neurona biológica y neurona artificial. 3](#Figura7)

[Figura 8: Función Binaria 3](#FigurasFuncionesActivación)

[Figura 9: Función Identidad 3](#FigurasFuncionesActivación)

[Figura 10: Función Tangente Hiperbólica 3](#FigurasFuncionesActivación)

[Figura 11: Función Relu (Rectificadora) 3](#FigurasFuncionesActivación)

[Figura 12: Función Softplus 3](#FigurasFuncionesActivación)

[Figura 13: Función Sigmoide 3](#FigurasFuncionesActivación)

[Figura 14: Función Exponencial 3](#FigurasFuncionesActivación)

[Figura 15: Estructura de una red neuronal simple 3](#Figura15)

[Figura 16: Algoritmo: Descenso de Gradiente 3](#Figura16)

[Figura 17:Comparación de tasas de aprendizaje 3](#Figura17)

[Figura 18: Gan 3](#FiguraGAN)

[Figura 19: Inferencia sobre variables latentes y generación 3](#Figura19)

[Figura 20: Arquitectura de red: Autoencoders Variacionales. 3](#Figura20)

[Figura 21: Arquitectura de red desde una perspectiva probabilística 3](#Figura21)

[Figura 22: Maximización del ELBO 3](#Figura22)

[Figura 23: Truco de la re-parametrización 3](#Figura23)

[Figura 24: Función de muestreo Re-parametrizada 3](#Figura24)

**INDICE DE TABLAS**

[Tabla 1. Funciones de Activación más utilizadas 11](#_Toc503004840)

# Introducción

Unos de los temas más complejos, más candentes y, en constante cambio en la actualidad, es sin duda la inteligencia artificial en general, y una de sus ramas más importantes, el aprendizaje automático, (Machine Learning), en particular. Es inevitable haber oído hablar una y otra vez sobre estas dos disciplinas científicas en los medios de comunicación y la repercusión que traerán a nuestras vidas.

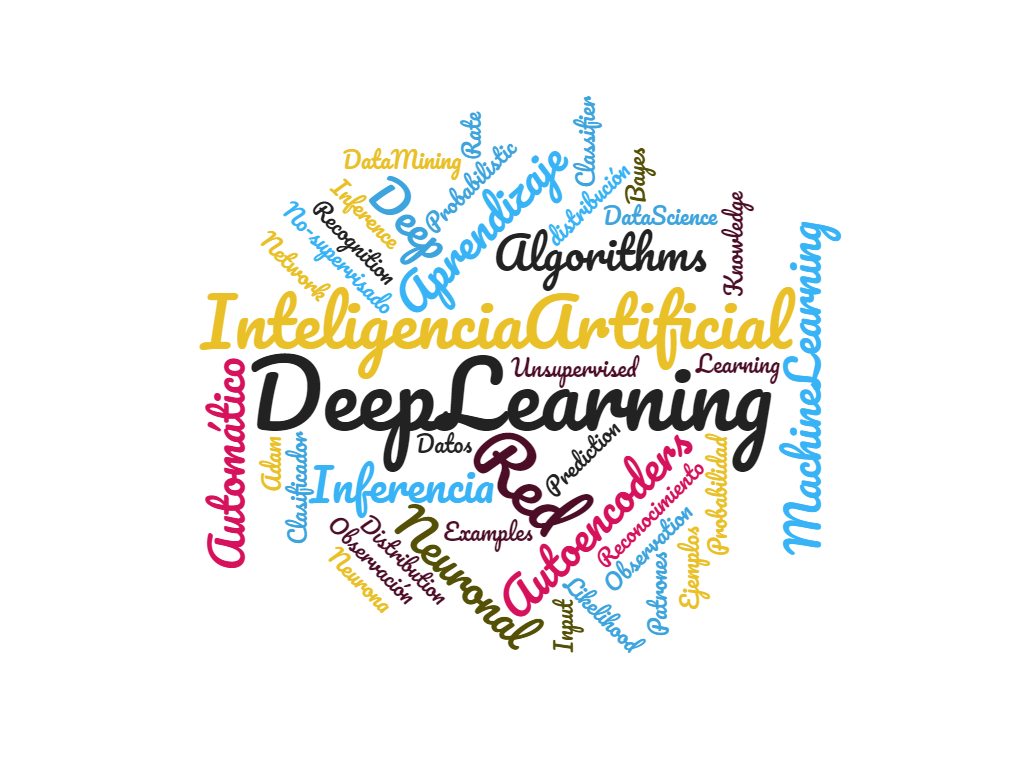


Figura 1. Nube de palabras relacionas con IA

La inteligencia artificial es la disciplina que se encarga de crear algoritmos que remplacen o simulen al ser humano en sus funciones intelectuales. Es decir, su objetivo es dotar a las máquinas de capacidad de observar, entender y actuar de una manera inteligente.

***“¿Qué somos las personas sino máquinas muy evolucionadas?”***

Marvin Minsky (Padre de la IA)

Todavía debemos esperar algunos años para poder comparar o intercambiar las capacidades intelectuales de las máquinas y de los humanos en todos sus puntos. Tema que no se tratara en esta memoria debido a su distendido dilema y su amplia controversia. Donde ya es más que conocido el gran debate sobre si presentará o no, una amenaza real en un futuro no muy lejano. Este miedo se debe al crecimiento exponencial que estamos viviendo con la inteligencia artificial que nos hace temer por una *“La rebelión de las maquinas”* al más estilo de Terminator 3[1].

Por otro lado, el aprendizaje automático es un campo de la IA que se especializa en proporciona a los sistemas, habitualmente informáticos, la capacidad de aprender por sí mismo sin que nadie los programe explícitamente para ello. Los algoritmos de aprendizaje automático tienen como fuente de conocimiento: los datos. De ahí que el machine learning este estrechamente relacionado con otro de los temas de la actualidad, el Big Data.

Las aplicaciones de la IA y el Machine learning en la actualidad son incontables, casi todos los dispositivos electrónicos que nos rodean están utilizando la IA sin que nos demos cuenta. El último informe de IDC (International Data Corporation) revela que “*Para 2019, el 40% de las iniciativas de transformación digital emplearán servicios de inteligencia artificial; para el año 2021, el 75% de las aplicaciones empresariales comerciales usarán IA”[2]* .

En esta memoria nos centraremos en un tipo de aprendizaje automático, Aprendizaje Profundo (Deep learning).

Deep Learning es una rama del machine learning, que habitualmente se conoce como redes neuronales profundas y tiene como objetivo orientar el proceso aprendizaje de forma muy similar a como aprendemos los seres humanos, simulando el funcionamiento de nuestro cerebro. Al igual que en el cerebro humano, la unidad básica de trabajo de un algoritmo de Deep learning es la Neurona.

Los algoritmos de Deep Learning se componen de muchas neuronales artificiales que forman una red neuronal, donde las diferentes capas de neuronas interaccionan entre si del mismo modo que lo hacen las neuronas de nuestro cuerpo. Este aprendizaje no requiere de participación directa de un humano en el proceso.

A modo de ejemplo el aprendizaje profundo es análogo al proceso de aprendizaje de un niño. A un niño nadie le enseña lo que es un coche, pero durante su proceso de aprendizaje va sacando características comunes que poseen los coches y crea un concepto coche.

El Deep learning se define pues, de forma más forma mal como el conjunto de algoritmos que tienen como objetivo el modelaje de abstracciones de alto nivel usando redes neuronales de varias capas de procesamiento.

Esta orientación clara hacia el aprendizaje no-supervisado, que a diferencias del aprendizaje supervisado solo se tienen los datos de entrada y su objetivo es encontrar una distribución o estructura dentro de los datos. Puesto que no se tiene un conocimiento previo, como si se tiene en el aprendizaje supervisado que conocemos en todos los datos (Salida y Entrada) dentro del proceso de aprendizaje.

Este enfoque nos lleva a un cambio drástico en la manera de orientar los problemas, ya que ahora el científico de datos no tiene que preocuparse de especificar el mismo las características que presenta una imagen. Si contiene un coche o no, sí el coche es amarillo o es verde… Además, este visón de aprendizaje logra conseguir resultados más rápidos y más precisos. Resultados que antes eran inimaginables con algoritmos de aprendizaje automáticos supervisados.

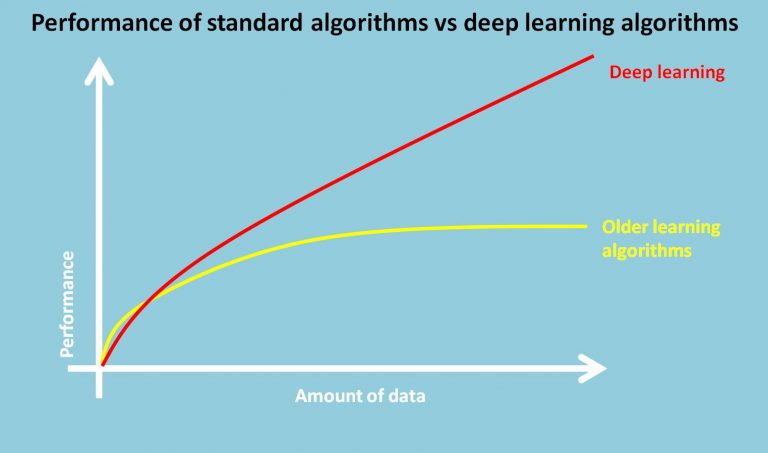


Figura 2. Comparación algoritmos: Deep Learning vs Algoritmos Machine Learning

Estos resultados son posibles gracias, entre otras cosas a la gran cantidad de datos que manejamos en la actualidad y a la potencia de computación de la que disponemos hoy en día.

El abanico de uso de esta disciplina abarca infinidad de temas. Como pueden ser temas de salud, bancarios, relacionados con el marketing y la publicidad, transporte…

Actualmente se han conseguido avances importantes en medicina con Deep Learning [4].

Una vez hecha una pequeña introducción al Deep Learning, vamos a centrarnos en el tema específico que abarca este trabajo de fin de grado, los modelos generativos profundos concretamente los **Autoencoders Variacionales (VAE)**.

Los Autoencoders variacionales son un tipo de red neuronal perteneciente al grupo de los Autoencoders, que al igual que estos tienen como objetivo reconstruir los datos de entrada. Para ello disponen de un codificador y un decodificador.



**AUTOENC****ODER**



Figura 3. Reconstrucción de imagen Autoencoders

Además, los Autoencoders variacionales permiten ser modelos generativos con resultados muy satisfactorios, a diferencia de los Autoencoders tradicionales que no funcionan tan bien para funciones generativas.

Este trabajo de fin de grado se centra en estas redes neuronales que están tanta fama en la comunidad de la Inteligencia artificial en general, y en la del Machine Learning en particular.

## Motivación

El aprendizaje profundo está viviendo una gran explosión en los últimos años. Sin lugar a duda ha roto con todos los techos existentes hasta el momento, con métodos tradicionales de machine learning.

Con la aplicación del aprendizaje no-supervisado usando redes neuronales se han conseguido avances en muchos ámbitos, que antes eran impensables. Hecho que está revolucionando, por ejemplo, la sanidad [11], con la aparición de técnicas de Deep Learning aplicadas directamente en los entornos de trabajo del personal sanitario. Estas técnicas ayudan a los médicos a realizar diagnósticos más rápidos y más precisos. Así como predecir enfermedades a tiempo para poder curarlas.



Figura 4: Análisis de imágenes médicas[10]

Este proyecto de fin de grado se ha realizado con el objetivo de crear un algoritmo de aprendizaje no-supervisado con técnicas de Deep learning que sea un modelo generativo con buenos resultado con el fin de poder utilizarlos en tareas reales muy diversas.

Los modelos generativos y en especial los Autoencoders Variacionales se están utilizando para dibujar imágenes, para trabajar de forma colaborativa con sistemas de recomendación[5], para modelar las reacciones de la audiencia a las películas[6], para clasificación de textos de forma semi-supervisada[7], para recuperar imágenes de teledetección en alta resolución [8] y para analizar imágenes médicas[9].

Estas son solo algunas de las aplicaciones que tiene los Autoencoders variacionales en la actualidad.

Por esta razón este trabajo se centrará en los VAE puesto que es un tipo de red neuronal muy utilizado en la actualidad y que esta consiguiendo grandes resultados en varios entornos de nuestra sociedad.

## Objetivos

El propósito de este trabajo es implementar un Autoencoder Variacional que arroje buenos resultados con la finalidad de ser utilizado en tareas de la vida real con éxito.

De forma detallada, las metas de este proyecto son:

* Definir sobre que conjunto de datos se va a trabajar. En este caso se utilizarán con los dataset MNIST y FREY FACE
* Investigar los diferentes tipos de Autoencoders/Autoencoders Variacionales que hay en la actualidad, sus características de modelaje, así como sus resultados.
* Investigar líneas de mejora de los algoritmos ya existentes.
* Diseñar modelo
* Implementar modelo
* Realizar pruebas exhaustivas sobre el modelo creado.
* Evaluar los resultados obtenidos.
* Sacar conclusiones.

## Organización de la memoria

La memoria consta de las siguientes secciones:

* **Estado del arte**

En este apartado se hará una breve introducción al concepto de redes neuronales y su analogía con el cerebro humano. Se realizará un repaso a la historia y actualidad de las redes neuronales. Se finalizará profundizando en la actualidad de los modelos generativos y en particular en los Autoencoders variacionales.

* **Diseño**

En este apartado se detallará de forma minuciosa el diseño elegido para implementar el modelo generativo de tipo VAE

* **Desarrollo**

En este apartado se describirá detalladamente como se ha realizado la implementación de la red neuronal dando lugar a un Autoencoder Variacional totalmente funcional.

* **Integración, pruebas y resultados**

En este apartado se integrarán todas las partes del desarrollo y se realizarán pruebas exhaustivas para obtener resultados para un gran número de configuraciones diferentes. También se analizar los resultados obtenidos tanto de forma visual como escrita.

* **Conclusiones y trabajos a futuro**

En este apartado se llevará a cabo una reflexión sobre los resultados obtenidos así como sus posibles aplicaciones. Se finalizara proponiendo nuevas línea de investigación para el futuro.

# Estado del arte

## Redes neuronales

Las redes neuronales son modelos computacionales que simulan el comportamiento del cerebro humano. Las unidades básicas se llaman, Neuronas. Estas funcionan de forma conjunta, interconectadas entre sí para resolver problemas que no tienen un algoritmo definido para convertir una entrada en una salida deseada.

### Cronología de las redes neuronales[12]

A lo de la historia han sido numerosos los científicos que han perseguido el sueño de construir maquinas capaces de realizar tareas con inteligencia simulando el funcionamiento del cerebro humano. Fueron los autómatas la primera aproximación a esas máquinas que aspiraban hacer cosas de forma inteligente simulando alguna labor del ser humano.

Este sueño se ha continuado hasta la actualidad y lo enmarcamos bajo el nombre de **Inteligencia Artificial**.

Pese a que las primeras explicaciones sobre el cerebro datan de la época de Heron de Alejandría (100 a.C), Platón (427-347 a.C) y Aristóteles (348.422 a.C). No fue hasta el 1936 cuando Alan Turing empezó a interesarse por el cerebro como forma de ver la computación.

Cronología de las redes neuronales:

* **Alan Turing-1936**

Fue el primero en interesarse por el cerebro como forma de ver la computación.

* **Warren McCylloch y Walter Pitts**

Fueron los primeros que desarrollaron teorías de la computación neuronal. Crearón una red neuronal simple mediante circuitos eléctricos

* **Donald Heeb-1949**

Fue el primero en explicar los procesos de aprendizaje. Aún hoy esta presente en la mayoría de las redes neuronales con la regla de aprendizaje de Hebb[13].

* **Karl Lashley-1950**

En sus ensayos, descubrió que la información en nuestro cerebro no está de forma centralizada, sino que está distribuida.

* **Congreso de Dartmouth-1956**

En este congreso se empieza hablar del nacimiento de la Inteligencia artificial.

* **Frank rosenblatt-1957**

Empezó el desarrollo del Perceptrón. Se trata de la red neuronal más antigua.

* **Bernard Widrow y marcial Hoff-1960**

Desarrollaron el modelo Adaline que fue la primera red neuronal utilizada en la vida real para eliminar el eco en las líneas telefónicas.

* **Karl Steinbeck y Die Lernmatrix-1961**

Desarrollaron una red neuronal que simulaba la memoria asociativa.

* **Stephen Grossberg-1967**

Creo la red neuronal llamada Avalancha, que permitía el reconocimiento continuo de habla y el aprendizaje de los brazos de un robot.

* **Marvin Minsky y Seymour Papert-1969**

Estos dos investigadores van a para el crecimiento de las redes neuronales por unas fuertes críticas al perceptrón hasta el 1982. A grandes rasgos, detractaron esta red neuronal porque no era capaz de resolver problemas simples como funciones no lineales o el XOR-exclusive.

* **Paul Werbos-1974**

Empezó a desarrollar la idea del aprendizaje de propagación hacia atrás.

* **Stephen Grossberg-1977**

Teoría de la resonancia adaptada. Es una arquitectura de red que simula la memoria a corto y largo plazo del cerebro.

* **Kunihiko Fukushima-1980**

Crea una red neuronal para reconocer patrones visuales.

* **John Hopfield-1982**

En su persona se reconoce el renacimiento de las redes neuronales con su obra “*Computación neuronal de decisiones en problemas de optimización”*

* **A partir de 1986**

El crecimiento de las investigaciones sobre redes neuronales fue en aumento.

* **David Rumelhart y G.Hinton-1986**

Definieron el algoritmo de aprendizaje de propagación hacia atrás (backpropagation).

Aparece por primera vez la idea de los AutoEncoders.

* **Diederik P. Kingma y Max Welling-2013**

Aparecen los Autoencoders Variacionales

* **Actualidad**

Son abundantes los estudios e investigaciones que se publican cada año sobre redes neuronales. Así como empresas que se suman a esta transformación digital utilizando redes neuronales.

**Figura 5.** **Cronología de redes neuronales**

### Analogía entre Neurona Artificial y Neurona biológica

De forma análoga al concepto de neurona biológica en el cerebro humano. Las neuronas artificiales son la unidad básica dentro de una red neuronal (cerebro humano). Estas tienen como propósito simular el comportamiento de las neuronas biológicas en él cerebro.

El cerebro está formado de millones de neuronas que se conectan entre sí, transmitiendo y recibiendo información de una a otra a través de la sinapsis.

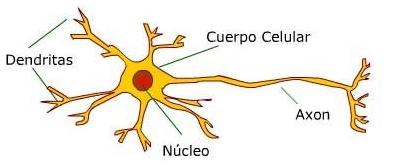


Figura 6. Neurona Biológica[14]

La sinapsis se lleva a cabo cuando una neurona transmite un impulso nervioso, a través de su axón, a otra neurona receptora por sus dendritas. Durante el proceso de la sinapsis se liberan una sustancia química que se denominan, Neurotransmisores.

Esta sustancia química actúa como una llave, esta llave regula el impulso eléctrico. Facilitando su paso (Neurotransmisores excitatorios) en algunos casos o dificultándoselo en otros (Neurotransmisores inhibitorios). La neurona receptora del impulso recibe múltiples señales de diferentes sinapsis que al unirlas todas, marcan el nivel de activación (Potencia postsináptica). El nivel de activación es el que marca la intensidad con que la neurona transmitirá el impulso a una nueva neurona.

Una vez hecha una breve y simplificada explicación del funcionamiento de una neurona biológica podemos realizar una analogía con las neuronas artificiales.



Figura 7. Analogía entre neurona biológica y neurona artificial.[15]

Las entradas de la neurona artificial serían los impulsos que envían una neurona biológica a través de su axón. Los pesos serian la intensidad de la sinapsis (Eficacia sináptica). El signo de los pesos serían los neurotransmisores. El producto de los pesos por las entradas da lugar a la entrada de la función de activación que sería la potencia postsináptica. La salida (ej: salida de la función sigmoide) sería la respuesta al estímulo que ha recibido por su entrada y seria análogo al axón.

### Funcionamiento de una Neurona Artificial

Una única neurona por sí misma no puede resolver problema complejo, pero al combinarse con miles o millones de ellas son capaces de resolver problemas muy complejos con resultados fantásticos.

Las neuronas artificiales están interconectadas las unas con las otras, siendo las salidas de unas las entradas de otras. Es **la función de red o función de ponderación** la que se encarga de transformar las entradas de una neurona en una única entrada global a ella, para ello necesita combinar las entradas con los pesos. Las entradas se multiplican por los pesos, estos últimos actúan como un potenciómetro, regulando la intensidad de influencia de esa entrada dentro de la neurona. Es por eso por lo que los pesos están en continuo ajuste (por ejemplo: Backpropagation) durante el proceso de aprendizaje. Un ejemplo de función de red podría ser:

Existen otros operadores aplicables a la función de red en lugar de la sumatoria como pueden ser el máximo, mínimo, el productoria…

Una vez se ha calculado la función de red, la salida de esta, se conectada con **la función de activación** que se encarga de calcular el nivel de activación de una neurona. Una neurona puede estar activa o inactiva, de la misma forma en que una neurona biológica esta excitada o inhibida.

Para calcular dicho nivel de activación, esta función transforma la entrada global que previamente ha calculado la función de red, en un estado de la neurona, o bien activa o bien inactiva.

La salida de la función de activación se combina con un valor umbral, que se trata de un parámetro que regula si la neurona emite una señal o no. Si la salida de la función de activación es mayor que el valor del umbral, la neurona enviará la señal, en caso contrario la neurona no emitirá señal.

|  |  |  |  |
| --- | --- | --- | --- |
| Funciones | Definición | Dominio | Gráfica |
| Binaria  o Escalón |  | (0,1) | Figura 8. Función Binaria |
| Identidad |  | (-∞,∞) | Figura 9. Función Identidad |
| Tangente Hiperbólica |  | (-1,1) | Figura 10. Función Tangente Hiperbólica |
| Relu  (Rectificadora) |  | (0, ∞) | Figura 11. Relu (Rectificadora) |
| Softplus |  | (0,∞) | Figura 12. Función Softplus |
| Sigmoide |  | (0,1) | Figura 13. Función sigmoide |
| Exponencial |  | (0, ∞) | Figura 14. Función exponencial |

Tabla 1. Funciones de Activación más utilizadas

### Estructura de las Redes neuronales

Las neuronas artificiales se organizan dentro de la red neuronal en capas.

Generalmente las redes neuronales constan de tres capas: una capa de entrada, una o varias capas ocultas y una capa de salida.

Capa de entrada

ENTRADAS

Capa oculta

Capa de salida

SALIDA

Figura 15. Estructura de una red neuronal simple

La capa de entrada es la que recibe los datos del exterior mientras que las capas ocultas procesan la información y se conectan con otras capas. La capa de salida se encarga de enviar información ya procesada al exterior.

### Propagación de la Información y proceso de aprendizaje en redes neuronales

Una red neuronal de forma simplificada aprende durante su fase de entrenamiento, que no es más que ajustar los pesos y el umbral que hagan que nuestra red nos devuelva los resultados deseados.

La búsqueda de estos parámetros es un proceso adaptativo e iterativo, por el cual la red va adquiriendo mejores resultados, ajustando los valores de estos parámetros y midiendo la diferencia entre los resultados obtenidos y los resultados deseados.

La actualización de los pesos y el umbral[[1]](#footnote-1) se lleva a cabo mediante propagación de la información a lo largo de toda la red. Existen dos tipos de propagación: Propagación hacia delante y propagación hacia delante o retropropagación (Backpropagation).

**Algoritmos de propagación**

* **Propagación hacia delante:** En esta fase la red trasmite los valores de la capa entrada hacia delante a lo largo de toda la red.
* **Retropropagación:** Esta fase corresponde con la fase de aprendizaje, que tiene como objetivo propagar los errores medidos en la salida de la red hacia atrás con el fin de actualizar los parámetros de la red (Pesos y el umbral).

Las funcione que miden el error entre la salida obtenida y la salida deseada, se denominan función de pérdida o función de error. Hay diversas funciones de perdida como el error medio cuadrático[18], la entropía cruzada[19], divergencia de Kullback-Leibler[20], máxima verosimilitud…

El objetivo del aprendizaje de una red neuronal es por lo tanto minimizar la función de perdida. Aunque cabe destacar que en ocasiones se desea, en su lugar, maximizar una función de perdida como es el caso de la función de pérdida del Lower Bound (ELBO) que se tratará más adelante.

Para conseguir minimizar la función de perdida se utilizan algoritmo de optimización que van cambiando poco a poco ([Tasa de aprendizaje](#Figura17)) los parámetros de la red (Actualización de pesos y bias) en dirección donde se minimice la pérdida o error.

Uno de los algoritmos más utilizados es el **descenso de gradientes[16]** que, de forma simplificada, se trata de un algoritmo iterativo que va dando pequeños saltos (Dependiendo de una tasa de aprendizaje) en dirección negativa al gradiente[[2]](#footnote-2) en busca del mínimo de la función a optimizar.

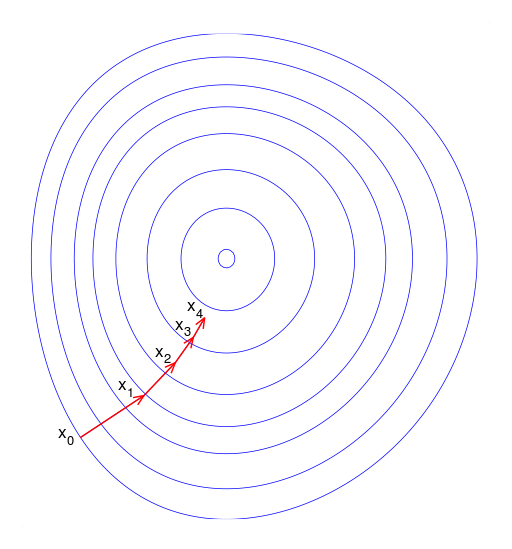


Figura 16. Algoritmo: Descenso de Gradiente

La velocidad a la que se actualizan los pesos de una red viene dada por la tasa de aprendizaje (learning rate). El valor de la tasa de aprendizaje suele estar entre 0 y 1.

Los valores muy próximos a 0 hacen que los pesos cambien lentamente, por lo tanto, la red tardará más tiempo en encontrar el mínimo (Convergencia), mientras que los valores próximos a 1 hacen que los pesos se actualicen rápidamente y por lo tanto la red encuentre el mínimo con más rapidez. En ocasiones utilizar tasas de aprendizajes grandes puede provocar oscilaciones que dificulten, o incluso imposibiliten encontrar el mínimo y como consecuencia los resultados serán pésimos.

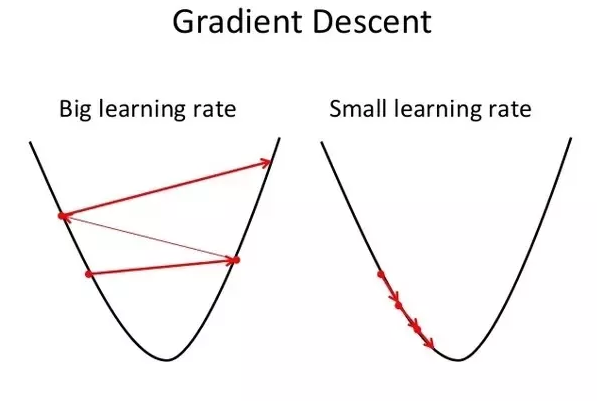


Figura 17. Comparación de tasas de aprendizaje

### Modelos generativos: Autoencoders Variational

Existen numerosos tipos de redes neuronales y enfoques dentro del mundo del machine learning y del Deep Learning, abordar todos ellos en este trabajo de fin de grado, sería muy extenso y nos desviaríamos del objetivo del mismo, que no es otro que profundizar en un tipo concreto, los **modelos generativos profundos**.

**Los modelos generativos** representan un área amplia dentro del machine learning, son un tipo de modelos de aprendizaje no-supervisado que tiene como objetivo generar datos.

Algunos de los problemas que han presentado los modelos generativos a lo largo de su historia son, por ejemplo, que han requerido fuertes suposiciones sobre la estructura de los datos. Además, han supuesto problemas computacionales sobre todo en procesos de inferencia costos como Monte Carlo.

Últimamente se están consiguiendo avances muy prometedores en este sentido, resolviendo el problema de las suposiciones fuerte sobre la estructura de datos y consiguiendo un entrenamiento rápido, usando técnicas de Backpropagation. Una de las redes neuronales más populares en esta dirección, es **el Autoencoder Variacional**, tema en el cual se centra este trabajo de fin de grado.

**Los Autoencoders variacionales** son un tipo de redes neuronales, relativamente modernas (2013), y en constante evolución. Pertenecen a la familia de los Autoencoders (1986), tienen una estructura muy similar a un perceptrón multicapa y, se utilizan principalmente en aprendizaje no-supervisado. Lo que significa que solo necesita datos de entrada sin etiquetar.

Su objetivo principal eso reconstruir los datos de entrada, sin olvidar su interés generativo. Para ello disponen de un codificador y un decodificador.

Gracias al pequeño error que tiene en sus aproximaciones, son consideradas como uno de los enfoques más populares dentro del aprendizaje no-supervisado, en concreto de distribuciones de datos complicadas. Esta cualidad le está permitiendo seguir ganando adeptos.

Actualmente se relación este tipo de modelos generativos con el arte y los diseñadores gráficos. Ya que permite generar más ejemplo de los que se poseen inicialmente, pero con algunas diferencias. Un ejemplo aplicado al sector de los videojuegos seria generar a partir de una base de datos de modelos 3D de árboles, generar un bosque repleto de ellos y todos ellos similares, pero no iguales. También se podría tomar un texto escrito a mano y generar más texto. Otro uso muy generalizado es la eliminación de ruido para reconstruir datos dañados, especialmente en imágenes.

Los avances más notables con este tipo de modelo generativo vienen desde ámbito del arte, un ejemplo de ello se ha producido en el 2017, en el que se ha conseguido generar música mediante un Autoencoder Variational[24] .Pese a ello, cabe destacar que, en otras áreas, tan importantes, como la salud, se han conseguido avances muy prometedores al respecto. Uno bastante relevante ha sido la extracción de un espacio latente biológicamente relevante, de transcriptomas de cáncer con Autoencoders Variational.[25]

En la actualidad, hay otro modelo generativo confrontado con los Autoencoders variacionales por ser el dominador en este terreno tan candente: Las redes generativas adversas (GAN[[3]](#footnote-3)).

**Los GAN** están formados por dos redes neuronales: Un generador y un discriminador que compiten entre sí en un juego de suma cero[[4]](#footnote-4). Una analogía muy usada para su explicación en este campo [27] es considerar un falsificador (Generador) que trata de generar dinero falso, mientras, por otro lado, un detective (Discriminador) intenta detectar si el dinero es falso o es real. Estas dos personas están enfrentadas, de ahí el nombre de estas redes (Generative Adversial Networks). Para salir victorioso, el falsificador tiene que generar dinero falso tan real como sea posible, en cambio, el detective tiene que ser muy bueno distinguiendo el dinero falso del real. Es fácil ver gracias con esta analogía, que la red va aprendiendo acorde el falsificador y detective (Generador y discriminador) van aprendiendo.

Imagen que contiene captura de pantalla

Descripción generada con confianza alta

Figura 18. Generative Adversial Networks

# Autoencoders Variacionales

Como se ha introducido en el apartado anterior, en la actualidad, hay dos modelos generativos confrontados por dominar en este campo: Las redes generativas adversas y los Autoencoders variacionales.

Ambos modelos presentan diferencias, en su arquitectura de red, así como, en la manera que tienen de entrenar los modelos. Los modelos basando en redes generativas adversas están basadas en la teoría del juego (Juegos de suma cero) y su propósito principal es encontrar el equilibrio entre la red discriminadora y la red generadora.

Por otra parte, los Autoencoders variacionales se basan en técnicas de inferencia bayesiana.

**Los Autoencoders variacionales** son una técnica de aprendizaje profundo no-supervisado que aprenden representaciones latentes de los datos observados, con la finalidad de entender mejor los datos observados y usar estas representaciones para generar datos adicionales. De forma más formal, son modelos que generan ejemplos X, siguiendo una distribución desconocida . La finalidad es, por lo tanto, aprender un modelo del cual podamos tomar muestras que sean tan similares a como sea posible.

***“What I cannot créate, I do not understand”***

*Richard Feynman*

## Funcionamiento de un Autoencoder Variacional

Para dar una aproximación a los Autoencoder variacionales supongamos por un momento, que deseamos generar rostros humanos. Lo primero que debemos hacer es imaginarnos que características tienen los rostros humanos, a partir de, los rostros que ya hemos visto alguna vez. Tiene ojos, tiene boca, está sonriendo tiene los ojos azules, es un rostro masculino o femenino… Una vez que tenemos una idea firme de cómo son los rostros humanos, podemos tomar muestras y generar rostros humanos.

Imaginemos que nuestra imaginación ha creado un vector de variables latentes a partir de todos los rostros de los que hemos visto alguna vez de esta forma [ojos azules, barbilla afilada, nariz redonda, boca pequeña, pómulos afilados, cejas rubias y voluminosas…]. A partir de este vector de variables latentes podemos generar rostros simplemente tomando muestras de este vector.

Un Autoencoder funciona a grandes rasgos de forma análoga, todos los rostros de los seres humanos de los que tenemos constancia serian análogo a nuestros datos de entrada, nuestra imaginación sería análogo al espacio latente o variables latentes[[5]](#footnote-5), que se obtienen de los datos de entrada, y los rostros generados por nuestra imaginación sería los datos devueltos por nuestra red.

1

2

Figura 19. Inferencia sobre variables latentes y generación

De forma muy simplificada el funcionamiento de un Autoencoder se representa en la Figura 19. Los X son los datos observados mientras que las Z son variables latentes. Nosotros queremos inferir las variables latentes desde puntos observados (1). Y posteriormente con las variables latentes se pueden generar datos (2). En realidad, nuestras variables latentes son una distribución de probabilidad, hecho en el que más tarde se profundiza.

## Estructura de la Red Neuronal

Como se ha introducido en el apartado anterior un Autoencoders variacional es una red neuronal que está formada por un codificador, un decodificado y un espacio latente. El codificador corresponde a una capa, o varias, de convolución que codifican la entrada en un espacio latente, es decir, estas capas de convolución extraen características de alto nivel. El decodificador, sin embargo, corresponde a una o varias capas de deconvolución, que a partir de las características de alto nivel genera nuevos datos.

Podemos decir que los Autoencoders Variacionales son una composición de redes neuronales, que como ya os habéis podido imaginar, está compuesta por dos redes que son el codificador y el decodificador. Tienen como punto de conexión el espacio latente o variables latentes, Aunque a menudo veamos esta composición de redes como una sola debemos tener presente esta connotación. Ambas redes neuronales (decodificador y decodificador) tiene una arquitectura igual que la de un perceptrón multicapa (MLP).

La clave del funcionamiento de un Autoencoder variacional reside en sus tres componentes principales: El codificador, el espacio latente y el decodificador.

* **Codificador**

Es la parte de la red que se encarga capturar información de alto nivel sobre los datos de entrada, estos datos de alto nivel son las variables latentes. Además, realiza una representación comprimida de los datos de entrada, fenómeno que se denomina reducción de la dimensionalidad. Su cometido es fundamental a la hora de obtener buenos resultados en la fase de decodificación ya que debe crear variables latentes ricas en información.

* **Espacio latente**

Son la representación de alto nivel de las características extraídas de los datos de entrada. Puesto que las variables latentes son en realidad una distribución de probabilidad, necesitaremos tomar muestras de este espacio latente para generar datos. Son la entrada del decodificador.

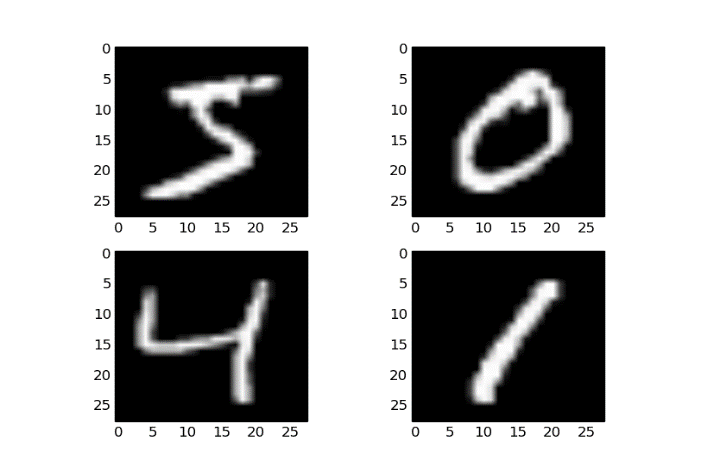
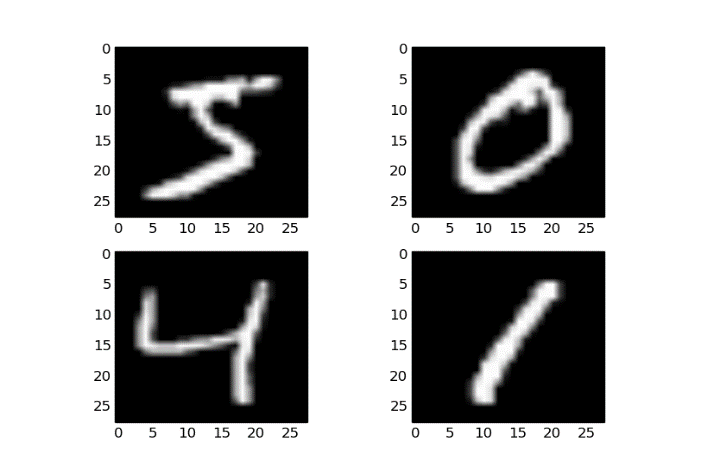
* **Decodificador**

Parte de la red que, a partir de las variables latentes aprendidas por el codificador, reconstruye las entradas.

ENCODER

ESPACIO

LATENTES



ENCODER

Figura 20. Arquitectura de red: Autoencoders Variacionales.

Los Autoencoders variacionales se diferencian de los Autoencoders típicos en que, estos aprender una distribución de probabilidad del espacio latente y, por lo tanto, se pueden utilizar para tareas generativas. Es decir, el codificador genera variables latentes que siguen una distribución, normalmente gaussiana. Esta restricción es la que lo diferencia de un Autoencoder típico, en el que su código latente presenta aleatoriedad.

## Fundamentos matemáticos

Una vez dada una explicación de los Autoencoders, desde un punto de vista más teórico. En esta sección abarcaremos una explicación desde una perspectiva matemática.

Todas las redes neuronales están fundamentadas en unos principios matemáticos que las explican, en el caso de los Autoencoders Variacionales se encuentra bajo un marco claramente probabilístico. Los VAE[[6]](#footnote-6) son por lo tanto un modelo probabilístico gobernado por los datos de entrada X, y las variables latentes Z.

En el prolegómeno de la explicación teórica de los VAE se ha expuesto que estos constan de tres partes fundamentes: el codificador, el decodificar y las variables latentes. Pues bien, desde un punto de vista matemático vamos añadir un cuarto elemento fundamental, la función de perdida, que será la encargada de medir el error de nuestra red, la cual explicaremos en breve.

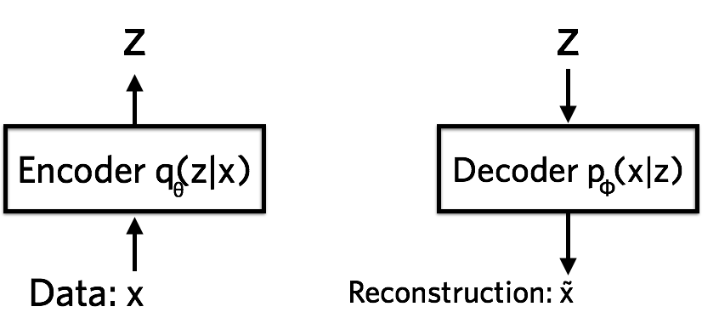


Figura 21. Arquitectura de red desde una perspectiva probabilística[29]

Dada esta representación de la red desde una visión probabilista, se pueden redefinir las partes de esta en términos probabilísticos. El codificador aprende una representación a alto nivel de los datos que, traducido a términos probabilístico, consiste en aprender una distribución de probabilidad (normalmente con una gaussiana) de los datos x, esto se representa en un espacio latente. Por este motivo el codificador se puede expresar como , donde representa los pesos y sesgos del codificador.

Las variables latentes Z, en términos probabilísticos se denotan de la misma manera. Por otra parte, el decodificador se puede definir como . Esta toma como entrada la representación de alto nivel dada por las variables latentes y reconstruye las entradas siguiendo una distribución de probabilidad, normalmente una distribución de Bernoulli.

Para dar más luz a este tema, de la misma manera que cuando se aprende un lenguaje de programación nuevo se realiza el típico ejemplo “Hola mundo”. En redes neuronales se suele utiliza el dataset MNIST[[7]](#footnote-7) para explicar modelos a modo de ejemplo.

Tenemos una imagen de entrada X, del dataset MNIST con 784 dimensiones (28x28), el codificador aprende características de alto nivel de esa imagen en un espacio latente Z que es considerablemente más pequeño en dimensión que X, este fenómeno se conoce como reducción de la dimensionalidad. El decodificador toma este espacio latente Z como entrada y devuelve una imagen de 784 dimensiones que representa la reconstrucción de la imagen de entrada X dada su espacio latente Z.

Para medir cuantitativamente como de bien, o de mal, está aprendiendo nuestra red, debemos medir, por un lado, la capacidad que tienen el codificador de representar las características de alto nivel de la imagen X en un espacio latente z, siguiendo una distribución . Y por otro lado medir la competencia del decodificador para reconstruir una imagen de entrada x dados su espacio latente z, siguiendo una distribución . A esta medida nos referimos como función de perdida.

Ahora que se han redefinido los componentes de un VAE en termino de probabilidad, podemos explicar los fundamentos matemáticos detrás de los VAE con una nomenclatura probabilística.

Retomando la analogía que introducíamos en la [sección 3.1](#_Funcionamiento_de_un) entre nuestra imaginación y los Autoencoders variacionales, se puede ir un paso más allá, y definir esta analogía en términos probabilísticos:

* : serian todos los rostros humanos.
* : sería la distribución de probabilidad que siguen nuestros datos, es decir, las características de alto nivel que se extraen de los rostros humanos.
* : seria análogo a nuestra imaginación.
* : sería la distribución de los datos generados dada la variable latente, es decir, el rostro generado a partir de nuestra imaginación.

Atendiendo puramente a temas probabilísticos un VAE es un modelo regido por los datos x y las variables latentes z. De tal forma que la probabilidad conjunta del modelo viene definida como:

Supongamos un muy simple del que podemos tomar muestras, podemos generar x similares usando una distribución . Por ejemplo, siendo una distribución gaussiana[[8]](#footnote-8) factorizada con parámetros dados por un perceptrón multicapa.

Nuestro objetivo es encontrar un qué maximice para crear variables latentes z ricas en información, con la finalidad de usar estas variables latentes aprendidas para tomar muestras.

Lamentablemente es intratable, ya que la derivada del marginal likelihood (Evidencia) , lo es.

se obtiene de la expresión que calcula la probabilidad posterior real, por el teorema de Bayes[[9]](#footnote-9):

Si marginamos las variables latentes podemos calcular como:

Como hemos dicho calcular esta integral es intratables, ya que requiere tiempo exponencial para ser calculada, por consiguiente es intratable. Además, existe otro inconveniente y es que los dataset pueden ser muy grandes y no caben en memoria.

Es aquí donde aparece el Autoencoder Variacional solucionando el problema de la intratabilidad, que presenta la probabilidad posterior real, . Además, soluciona el problema de dataset muy grandes usando minibatches[[10]](#footnote-10) en lugar del dataset completo.

Para ello los VAE añaden una red de reconocimiento , con la finalidad de aproximar esta distribución, a la distribución posterior real . En otras palabras, un VAE añade una red que sea capaz de aproximar (Inferir) la distribución posterior real , usando una distribución más simple y tratable, representada por . Esta red no es otra que nuestro ya conocido codificador.

Para inferir la distribución posterior real , en los Autoencoders Variacionales se usa un método denominado, Inferencia Variacional (VI). Qua aproxima la distribución posterior real con una familia de distribuciones .

### Inferencia Variacional

La Inferencia Variacional es uno de los métodos más utilizados de inferencia bayesiana, otros métodos muy populares son Metropolis-Hastings [32] y Monte Carlo [33]. Tanto Metropolis-Hastings como Monte Carlo son métodos basados en técnicas de muestreo [33]. Los métodos de muestreo presentan algunas deficiencias. A pesar ser capaces de encontrar una solución óptima con el tiempo suficiente es difícil saber si la solución que dan es buena debido al tiempo limitado que tiene en la práctica.

Las diferencias principales entre las técnicas de muestreo y la inferencia variacional residen en que los métodos de muestreo son capaces de encontrar soluciones optimas con suficiente tiempo y práctica, mientras que con técnicas variacionales es difícil encontrar esta solución óptima. Pero sin embargo con técnicas variacionales sabemos cuándo han convergido y habitualmente son más adecuados para técnicas de optimización del gradiente estocástico.

La idea principal de inferencia variacional reside en transformar un problema de aproximación a una distribución condicional compleja en un problema de optimización.

En nuestro caso tenemos una distribución de probabilidad intratable que denotaremos como . El método de inferencia variacional tratará de resolver un problema de optimización encontrando una distribución , dentro de una familia de distribuciones tratables y simples, que sea muy cercana a la distribución de probabilidad . Resumiendo, el problema de optimización consiste en encontrar una distribución simple y tratable que sea muy similar a .

Para poder usar en lugar de , obteniendo así una solución aproximada.

El parámetro variacional depende de la familia de distribuciones. Si por ejemplo es una gaussiana, el parámetro corresponde con los parámetros de una gaussiana, la media y la varianza de las variables latentes para cada punto de los datos.

Para formular la inferencia como un problema de optimización demos medir la similitud entre , es decir, como de bien se aproxima a . Para medir la diferencia entre dos distribuciones se utiliza la divergencia de Kullback-Leibler (KL), que mide la información que se pierde cuando se usa aproximando .

La divergencia de Kullback-Leibler [34] es una medida de la diferencia entre dos distribuciones de probabilidad, por lo tanto, es siempre positiva o igual a cero. Se define como:

Nuestro cometido es por lo tanto resolver este problema de optimización:

Podemos observar que nuevamente aparece el problema de la intratabilidad en la divergencia KL, debido a la evidencia , que como ya sabemos es intratable.

Como no podemos calcular la divergencia KL directamente, optimizamos un objetivo alternativo llamado límite inferior variacional (ELBO), que si es tratable:

El termino ELBO proviene del logaritmo del marginal likelihood de los datos y se trata del negativo de la divergencia mas , es decir:

y se comprueba que:

Por este motivo este término se le conoce como límite inferior en la evidencia “Lower Bound”. A menudo este término, en la jerga de Autoencoders variacionales se menciona como “Stochastic Gradient Variational Bayes” (SGVB)

Ahora podemos reescribir la evidencia como:

Observando esta expresión y sabiendo que el KL es siempre positivo o igual a cero. Podemos ver que maximizando el ELBO respecto de hacemos que la divergencia KL sea muy pequeña y consecuentemente mejora la evidencia . Minimizar la divergencia KL es por consiguiente equivalente a maximizar el ELBO respecto de

Si una vez maximizado el ELBO, este se aproxima a la distribución de los datos P(x) la distancia entre las dos distribuciones es cercana a cero y habremos minimizado la distancia entre es decir minimizado la divergencia KL.

ELBO

KL

Figura 22. Maximización del ELBO

### Optimización y Función de perdida

La función de perdida como ya se comento anteriormente en la [sección 2.1.5](#_Propagación_de_la) es la encargada de medir el error entre la salida obtenida y la salida deseada.

El objetivo del aprendizaje de una red neuronal es siempre minimizar la función de perdida. Sin embargo, en nuestro caso como se ha detallado en la sección anterior, nuestro objetivo es maximizar la función de perdida, concretamente el ELBO.

Dicho esto, nuestra función de perdida quedaría:

Siendo el ELBO:

Nuestra función de perdida (ELBO) está compuesta por dos términos:

* **Perdida latente**

Se mide con la divergencia de Kullback-Leibler y representa la perdida de información latente cuando aproximamos la distribución posterior real con la distribución variacional.

* **Perdida generativa**

Mide con que precisión se reconstruyen las imágenes de entrada respecto de las imágenes de salida de la red. Se utiliza la entropía cruzada[19] para medir el error.

Para conseguir minimizar la función de perdida se utilizan algoritmo de optimización que van cambiando poco a poco los parámetros de la red (Actualización de pesos y bias) en dirección donde se minimice la pérdida o error.

El descenso de gradientes es uno de los algoritmos de optimización más utilizados que, de forma simplificada, se trata de un algoritmo iterativo que va dando pequeños saltos (Dependiendo de una tasa de aprendizaje) en dirección negativa al gradiente en busca del mínimo de la función a optimizar. Una vez más, como nuestro objetivo es maximizar el ELBO, en lugar de ir en dirección opuesta al gradiente para buscar un mínimo, vamos en el mismo sentido que el gradiente, utilizando técnicas de **ascenso de gradiente estocástico** para buscar un máximo de la función.

Para poder aplicar métodos de optimización basadas en el gradiente, necesitamos que toda la red sea diferenciable, ya que estas técnicas se basan en el cálculo de derivas.

Aquí nos encontramos ante uno de los problemas que presentan los Autoencoders variacionales que para poder utilizar el ascenso de gradiente tenemos que ser capaces de calcular gradientes en todos los nodos de la red. Pero las variables latentes presentan aleatoriedad por lo que no son diferenciables. Un ejemplo, podríamos muestrear z con una distribución gaussiana cuyos parámetros son las salidas del codificador, pero esta operación de muestreo no tiene gradiente.

Para solucionar este problema se usa el “Truco de Re-parametrización”.

### Truco de la Re-parametrización

El truco de la Re-parametrización surge porque para entrenar un Autoencoder variacional se necesita realizar retropropagación a través de toda la red para utilizar técnicas de descenso/ascenso de gradiente. Para ello necesita por calcular derivadas en todos los nodos de la red. Esta condición se rompe cuando queremos muestrear las variables latentes z, por ejemplo, a partir de una distribución gaussiana cuyos parámetros son la salida del codificador.

Este truco es muy simple y consiste en desviar la operación no diferenciable fuera de la red, aunque siga estado presente, ya no interfiere en el entrenamiento de la red. En nuestro caso, convierte la variable aleatoria z no diferenciables en una función diferenciable de x, desacoplada de aleatoriedad.

ENCODER

DECODER

(

DENTRO DE LA RED

EXTERNO A LA RED

Figura 23. Truco de la Re-parametrización

Supongamos el ejemplo de utilizar una distribución gaussiana cuyos parámetros son dados por la salida del codificador, que además de ser el diseño elegido en mi propio Autoencoders variacional, es el más utilizado. Si por otro lado tomamos muestras de una gaussiana estándar con media 0 y varianza 1, podemos convertirla en cualquier gaussiana que nos imaginemos siempre y cuando conozcamos la media y la varianza. Dicho esto, una función de muestreo de siendo podría ser:

Una vez dada esta función de muestreo, a partir de aquí en lugar de muestrear z como . Z es ahora una función que toma como parámetros un ∈ y una media y una varianza ) que son las salidas del codificador. Ahora para utilizar técnicas de descenso/ascenso de gradiente solo necesitamos derivadas parciales respecto de mientras que ∈ no se tiene en cuenta a la hora de calcular derivadas, ya que esta fuera de la red.

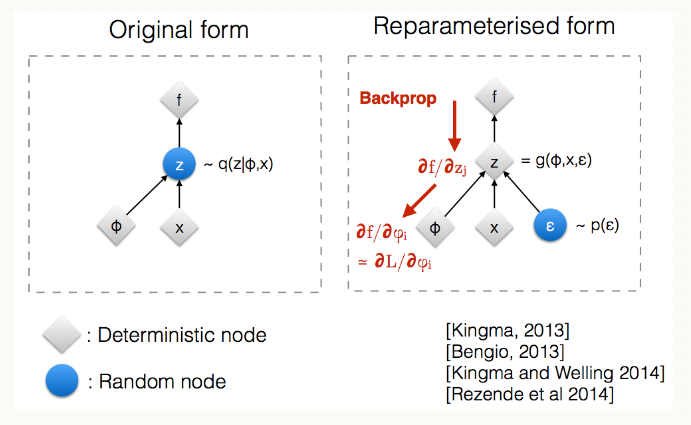


Figura 24. Función de muestreo Re-parametrizada.

# Diseño de la Red Neuronal

# Implementación

## Subsección

### Subsubsección

# Experimentos y resultados

# Conclusiones y trabajo futuro

## Conclusiones

## Trabajo futuro

# Referencias

1. Terminator3: <https://www.filmaffinity.com/es/film477986.html>
2. IDC (International Data Corporation) Predicciones para 2018 <http://www.blog-idcspain.com/predicciones-idc/>
3. ConversionUplift:https://www.conversion-uplift.co.uk/glossary-of-conversion-marketing/artificial-intelligence/
4. Deep learning for healthcare: <https://www.nvidia.com/en-us/deep-learning-ai/industries/healthcare/>
5. KDD papers: <http://www.kdd.org/kdd2017/papers/view/collaborative-variational-autoencoder-for-recommender-systems>
6. Disney research: <https://www.disneyresearch.com/publication/factorized-variational-autoencoder/>
7. Cornell University Library: <https://arxiv.org/pdf/1603.02514.pdf>
8. American Association of geographers: <https://aag.secure-abstracts.com/AAG%20Annual%20Meeting%202018/abstracts-gallery/702>
9. Qure.ai Blog: <http://blog.qure.ai/notes/using-variational-autoencoders>
10. Forbes: <https://www.forbes.com/sites/tomdavenport/2017/11/05/revolutionizing-radiology-with-deep-learning-at-partners-healthcare-and-many-others/#63e01ad85e13>
11. NetWorkWorld: <https://www.networkworld.com/article/3183745/health/how-deep-learning-is-transforming-healthcare.html>
12. Redes Neuronales: <http://www.itnuevolaredo.edu.mx/takeyas/apuntes/Inteligencia%20Artificial/Apuntes/tareas_alumnos/RNA/Redes%20Neuronales2.pdf>
13. Regla de aprendizaje de Hebb:<https://es.wikipedia.org/wiki/Teor%C3%ADa_hebbiana>
14. Figura de una Neurona: <http://www.educarchile.cl/ech/pro/app/detalle?ID=137486>
15. Analogía entre neurona biologia y neurona artificial: <http://www.um.es/LEQ/Atmosferas/Ch-VI-3/F63s4p3.htm>
16. Descenso de gradiente:<https://en.wikipedia.org/wiki/Gradient_descent>
17. Variational Autoencoders:<https://arxiv.org/pdf/1606.05908.pdf>
18. EMC: <https://es.wikipedia.org/wiki/Error_cuadr%C3%A1tico_medio>
19. Cross-Entropy:<https://en.wikipedia.org/wiki/Cross_entropy>
20. Divergencia KL: <https://en.wikipedia.org/wiki/Kullback%E2%80%93Leibler_divergence>
21. Christopher M.Bishop: Pattern Recognition and Machine Learning
22. P.Kingma y Max Welling. Auto-encoding Variational Bayes:<https://arxiv.org/pdf/1312.6114.pdf>
23. Rajesh Ranganath, Sean gerrish y David M.Blei. Black Box Variational Inference: <https://arxiv.org/pdf/1401.0118.pdf>
24. Music generation: <https://arxiv.org/pdf/1705.05458.pdf>
25. Extracting a Biologically Relevant Latente Space from cancer Transcriptomes <https://www.biorxiv.org/content/early/2017/08/11/174474>
26. Juego de suma cero: <https://en.wikipedia.org/wiki/Zero-sum_game>
27. Generative Adversial Networks: <https://arxiv.org/pdf/1406.2661.pdf> y <http://shashwatverma.com/generative-adversarial-networks.html>
28. Blog de Agustinus Kristiadi: <https://wiseodd.github.io/techblog/2016/12/10/variational-autoencoder/>
29. Encoder-decoder:<https://jaan.io/what-is-variational-autoencoder-vae-tutorial/>
30. Distribución Gaussiana:<https://es.wikipedia.org/wiki/Distribuci%C3%B3n_normal>
31. Teorema de bayes:<https://es.wikipedia.org/wiki/Teorema_de_Bayes>
32. Metropolis-hastings: <https://en.wikipedia.org/wiki/Metropolis%E2%80%93Hastings_algorithm>
33. Metodo de Montecarlo:<https://es.wikipedia.org/wiki/M%C3%A9todo_de_Montecarlo>
34. Divergencia Kullback-Leibler: <https://es.wikipedia.org/wiki/Divergencia_de_Kullback-Leibler>
35. Variational inference: <https://arxiv.org/pdf/1601.00670.pdf>
36. Distribución de Bernoulli: <https://es.wikipedia.org/wiki/Distribuci%C3%B3n_de_Bernoulli>

# Glosario

VAE Variational Autoencoders (En Castellano: Autoencoders Variacional)

GAN Generative Adversial Networks

VI Variational Inference

# Anexos

## Manual de instalación

## Manual del programador

## Anexo …

## 

1. El parámetro umbral pasa a llamarse “bias” en el proceso de entrenamiento y se considera como un peso sináptico () más asociado a la neurona. [↑](#footnote-ref-1)
2. El gradiente de forma sencilla es la pendiente que representa, la tangente en un punto de la función [↑](#footnote-ref-2)
3. GAN: Generative Adversial Networks [↑](#footnote-ref-3)
4. Juego de suma cero [26]: Lo que unos jugadores ganan otros lo pierden en la misma proporción. Es decir, si damos un valor positivo a las ganancias y un valor negativo a las perdidas, la suma total es cero. [↑](#footnote-ref-4)
5. Variable latente: Son variables que no son observadas explícitamente, sino que son inferidas por datos observados directamente. [↑](#footnote-ref-5)
6. VAE: Autoencoders Variational [↑](#footnote-ref-6)
7. MNIST: es un dataset de imágenes de dimensión 28x28 (784 Pixeles), de dígitos escritos a mano comprendidos entre el 0 y el 9. [↑](#footnote-ref-7)
8. Distribución Gaussiana, también denominada normal es una distribución muy utilizada en estadística, como en el caso de la inferencia estadística. [30] [↑](#footnote-ref-8)
9. Teorema de Bayes: se trata de la probabilidad condicionada por un evento aleatorio. [31] [↑](#footnote-ref-9)
10. Minibatches: son porciones de menor tamaño del dataset completo. [↑](#footnote-ref-10)