Redes neuronales bayesianas
Bayesian deep learning

Trabajo de Fin de Grado
Curso 2022–2023

Autor
Pablo Fornet Martín

Director
Miguel Palomino Tarjuelo

Grado en Ingeniería Informática
Facultad de Informática
Universidad Complutense de Madrid

Redes neuronales bayesianas
Bayesian deep learning

Trabajo de Fin de Grado en Ingeniería Informática

Autor
Pablo Fornet Martín

Director
Miguel Palomino Tarjuelo

Convocatoria: *Septiembre 2023*

Grado en Ingeniería Informática
Facultad de Informática
Universidad Complutense de Madrid

1 de septiembre de 2023

Dedicatoria

A mi hermano

Agradecimientos

Quiero expresar mi sincero agradecimiento a todas las personas que han sido parte fundamental en este viaje académico que culmina con la realización de mi trabajo de fin de grado. En primer lugar, quiero agradecer a mis amigos, quienes me han brindado su apoyo incondicional, su ánimo constante y su compañía inigualable durante estos años de estudio. A mi familia, les agradezco por su amor inquebrantable, su comprensión y su constante apoyo emocional y económico, sin el cual este logro no sería posible. También, quiero expresar mi gratitud a mis profesores, quienes compartieron su conocimiento y experiencia, guiándome en mi formación académica. Finalmente, un agradecimiento especial a mi tutor, Miguel Palomino Tarjuelo, por su dedicación, paciencia y valiosas orientaciones que fueron fundamentales para la realización de este trabajo. Gracias por ser parte de mi éxito.

Resumen

Redes neuronales bayesianas

Las redes neuronales modernas son una herramienta que permite resolver una gran variedad de problemas y retos en el ámbito de la inteligencia artificial. Sin embargo debido a que operan como cajas negras, la incertidumbre de sus predicciones es difícil de cuantificar. El paradigma bayesiano ofrece herramientas para cuantificar la incertidumbre asociada a las predicciones de una red neuronal. La principal diferencia del enfoque bayesiano es la marginalización, en vez de seleccionar una única configuración de parámetros. La marginalización, que consiste en tener en cuenta varias configuraciones de parámetros que se ajustan a los datos de entrenamiento puede mejorar la precisión de la red neuronal, pero sobre todo permite aproximar mejor la incertidumbre de la predicción. En este trabajo explicamos una visión general de cómo funciona y se implementa una red bayesiana y las principales ventajas que estas presentan.

Palabras clave

Bayes, Red neuronal estocástica, Red neuronal bayesiana, Marginalización, Conjuntos de redes neuronales, Incertidumbre, Distribución a posteriori.

Abstract

Bayesian deep learning

Modern neural networks are a tool for solving a wide variety of problems and challenges in the field of artificial intelligence. However, because they operate as black boxes, the uncertainty of their predictions is difficult to quantify. The Bayesian paradigm offers insights to quantify the uncertainty associated with the predictions of a neural network. The main difference of the Bayesian approach is marginalization, rather than selecting a single parameter configuration. Marginalization, which consists in taking into account several parameter configurations that fit the training data, can improve the accuracy of the neural network, but above all it allows to better approximate the uncertainty of the prediction. In this work we present an overview of how Bayesian deep learning works and how is implemented and the main advantages it offers.

Keywords

Bayes, Stochastic neural network, Bayesian deep learning, Marginalization, Deep ensembles, Uncertainty, Posterior distribution.

Índice

Índice de figuras

Capítulo 1

Introducción

El aprendizaje automático es una rama de la inteligencia artificial que estudia los métodos por los que una computadora puede aprender mediante el uso de datos de entrenamiento. El objetivo es aproximar una función $f(x)$ que desconocemos pero de la que tenemos datos de entrenamiento, normalmente una gran cantidad de ellos y de la forma $(x_i, f(x_i))$. La función f puede corresponder a un problema de clasificación o a un problema de regresión. Será un problema de clasificación cuando la salida solo pueda tomar un número finito de valores, por ejemplo, la función capaz de distinguir si una imagen es una gato o lo no es. Un problema de regresión es aquel en el que se intenta predecir un valor numérico, por ejemplo el precio de una vivienda.

Para aproximar f partimos de un conjunto de funciones $\{g(x;\theta)\}$. Este conjunto será nuestro modelo, donde θ es un vector que corresponde a los parámetros de dichas funciones. Por ejemplo, supongamos el modelo $M = \{g(x;\theta)\}$ tal que $g(x;\theta) = w_0 + w_1 x$; en este caso $\theta = (w_0, w_1)$ y dependiendo de los valores que le demos a θ tendremos una u otra función $g(x)$. Finalmente, existen algoritmos como descenso de gradiente que son capaces de encontrar un valor θ^* para los parámetros tal que $g(x;\theta^*)$ aproxima la función objetivo f. Este proceso de encontrar los valores adecuados para los parámetros es a lo que nos referimos cuando decimos que la computadora aprende. El proceso se suele plantear como un problema de optimización donde se intenta reducir la diferencia entre las predicciones dadas por la función aproximada g y la función objetivo f. Más adelante veremos que el paradigma bayesiano no está interesado en la optimización sino en la marginalización de los parámetros. Veremos qué es la marginalización y las ventajas que aporta que la computadora aprenda de esta manera.

Sobre un modelo de aprendizaje automático se pueden definir dos propiedades que nos ayudan a entender el comportamiento del modelo: el soporte y el sesgo inductivo (Wilson e Izmailov, 2022). Siguiendo con el ejemplo anterior, el modelo M se dice que tiene un soporte reducido ya que solo puede representar conjuntos de datos que tengan una relación lineal, *i.e.* sin importar qué valores tomen sus parámetros, el modelo M nunca podrá representar datos con relaciones cuadráticas u otro tipo de relaciones. La probabilidad de que el modelo M genere, si muestreamos al azar, un conjunto de datos con relación cuadrática es 0. Si $p(D|M)$ es la probabilidad de que un conjunto de datos D sea generado por el modelo M, definimos el soporte del

modelo como los conjuntos de datos D tales que $p(D|M) > 0$.

Definimos el sesgo inductivo como la distribución del soporte dada por $p(D|M)$. Continuando con el ejemplo, el sesgo inductivo dependerá de cómo se eligen a priori los parámetros w_0 y w_1. Si w_0 y w_1 siguen una distribución uniforme, entonces la distribución inducida también será uniforme, pero si w_0 y w_1 son muestras de una distribución normal $N(0, 1)$ entonces será más probable que genere datos con una relación lineal con pendiente cercana a 0 y menos probable que genere conjuntos de datos con relación lineal con mucha pendiente.

1.1. Motivación

Wilson et al. (2022) sugiere que estas dos propiedades, el soporte y el sesgo inductivo, desde un punto de vista probabilístico son principalmente de lo que depende la capacidad de generalización, esto es, la capacidad para realizar predicciones precisas y acertadas sobre datos no vistos previamente durante el entrenamiento. Idealmente queremos que un modelo sea flexible para que soporte cualquier conjunto de datos sin importar qué tipo de relación tengan, o incluso capaz de adaptarse a datos que no siguen ninguna relación. Además, queremos que el sesgo inductivo a priori del modelo asigne más probabilidad a los conjuntos de datos que tienen más posibilidades, a nuestro juicio, de representar el problema que estamos resolviendo. De esta manera el modelo colapsará rápidamente en torno a la solución. Un ejemplo de modelo que representa a priori la hipótesis son las estructuras convolucionales usadas en redes neuronales para problemas que tienen como entrada imágenes.

Hay que remarcar que la flexibilidad de un modelo no implica necesariamente que tienda a representar funciones con mucha complejidad. Existen modelos muy flexibles capaces de soportar prácticamente cualquier conjunto de datos pero que tienen un sesgo inductivo hacia funciones suaves y simples. Un ejemplo de esto son los procesos gaussianos con un *kernel* RBF, un modelo de aprendizaje automático que ha demostrado ser capaz de ajustarse a datos aleatorios mientras que las funciones que produce a priori sin entrenamiento son sencillas. Tradicionalmente las métricas utilizadas para evaluar la capacidad de generalización de un modelo, como la complejidad de Rademacher (Mohri y Rostamizadeh, 2009), similares al soporte que hemos definido antes, han sido unidimensionales y, por lo tanto, no han logrado brindar una visión completa de la generalización. Nosotros razonaremos sobre las redes neuronales bayesianas desde el punto de vista probabilístico de la generalización usando una métrica bidimensional que tiene en cuenta tanto el soporte como el sesgo inductivo.

Como hemos dicho antes, el aprendizaje desde la perspectiva bayesiana no es un proceso de optimización sino de marginalización. Esto significa intuitivamente que, en vez de quedarse con el valor óptimo de los parámetros al hacer predicciones, la idea es considerar todos los posibles valores que los parámetros pueden tomar y ponderar cada conjunto de valores en función de cómo de buenas son las predicciones que da ese conjunto, dados los datos de entrenamiento. Veremos que este enfoque tiene varias ventajas tanto en la práctica como a la hora de razonar sobre el modelo.

Una de las principales ventajas de los métodos bayesianos es que proporcionan

una forma natural de entender y cuantificar la incertidumbre de las redes neuronales y como resultado tienden a mostrar menos confianza en sus predicciones que las redes neuronales clásicas, que tienen inclinación a sobrestimar cómo de buenas son sus predicciones. También permiten distinguir entre incertidumbre epistémica del modelo y la incertidumbre aleatoria de los datos. Esto les permite aprender con un menor número de datos de entrenamiento sin sufrir sobre-ajuste. Por último, en la práctica las redes neuronales bayesianas han demostrado una mayor precisión que las redes neuronales clásicas tanto en datos poco dañados como en datos con un alto grado de alteración y han mostrado reducir o incluso acabar con el problema del doble descenso como veremos más adelante.

1.2. Objetivos

El objetivo de este trabajo es dar una visión general de las redes neuronales bayesianas de manera que un público general con unos mínimos conocimientos de matemáticas e informática sea capaz de comprender. Explicar las ideas en las que se basan estos modelos y algunos detalles sobre su implementación práctica. También interpretar los modelos clásicos que usan optimización desde el punto de vista bayesiano y presentar las ventajas que las redes bayesianas tienen sobre las no bayesianas. Por último comprobar estas ideas en un pequeño experimento. El código correspondiente se encuentra disponible en Fornet (2023)

Capítulo 2

Trabajos relacionados

MacKay (1992), MacKay (1995) y Neal (1996) son los primeros trabajos notables en torno a la idea de redes neuronales bayesianas. Estos trabajos defendían que los modelos bayesianos debían ser lo más flexibles posible, en línea con trabajos posteriores (Wilson et al., 2022).

Los trabajos más recientes se han centrado en desarrollar nuevas herramientas para aplicar el paradigma bayesiano en redes neuronales modernas. Principalmente se han centrado en métodos escalables de inferencia (e.g., Blundell et al., 2015; Ritter et al., 2018; Maddox et el., 2019) y el estudio de la distribución a priori (e.g., Yang et al., 2019; Nalisnick, 2018).

En otros trabajos, Pearce et al. (2018) sugiere que los conjuntos de modelos realizan una aproximación de la inferencia bayesiana y Gustafssonet al. (2019) brevemente menciona que los conjuntos se pueden entender como muestras de una distribución a posteriori aproximada. Sin embargo, Wilson et al. (2022) argumenta que el paradigma bayesiano va mucho más allá de una aproximación a partir de muestras de una distribución a posteriori y añade que para tener una buena distribución de predicción ni si quiera es necesario el muestreo de la distribución a posteriori.

El doble descenso describe una función de error de la generalización de un modelo que desciende, asciende y vuelve a descender con el aumento de la flexibilidad del modelo. La figura 2.1 muestra este comportamiento. El doble descenso fue pronto descubierto por Opper et al. (1990) y extensamente estudiado recientemente por Belkin et al. (2019). Nakkiran et al. (2020) mostró cómo la regularización l_2 podía mitigar el doble descenso. Alternativamente, Wilson et al. (2022) demuestra cómo ciertos algoritmos bayesianos también son capaces de aliviar e incluso acabar con este comportamiento.

Bayes seguro (*safe Bayes*) es el nombre que se le ha dado a los modelos que atemperan la distribución a posteriori (de Heide et al., 2019; Gründwald et al., 2017; Zhang, 2006). En un artículo reciente Wenzel et al. (2020) cuestiona el atemperamiento en redes neuronales bayesianas. Wilson et al. (2022) analiza sus resultados y responde a las cuestiones que plantean.

La realización de este trabajo está basado principalmente en dos artículos: *Bayesian Deep Learning and a Probabilistic Perspective of Generalization* de Wilson

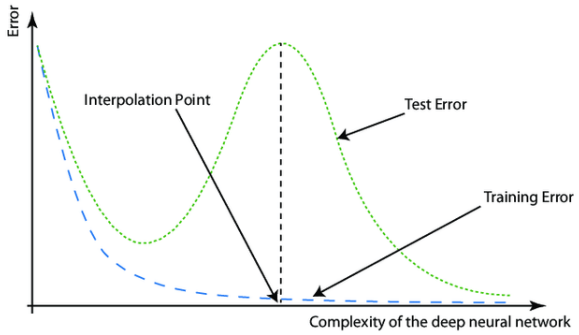


Figura 2.1: Curva del doble descenso: Fuente Kutyniok (2022).

et al. (2022) y *Hands-on Bayesian Neural Networks – A Tutorial for Deep Learning Users* de Jospin et al. (2022). Las ideas recogidas en los capítulos Redes neuronales estocásticas, Distribución a priori y Distribución a posteriori están sacadas principalmente de Jospin et al. (2022). Wilson et al. (2022) presenta las ideas descritas en la sección Ventajas del paradigma bayesiano. Por último también hemos utilizado ideas sacadas del libro Roberts et al. (2022) como las interpretaciones de los diferentes términos del teoremas de Bayes.

Capítulo 3

Teorema de Bayes

El paradigma bayesiano en estadística se basa en dos ideas principales. La primera es que la probabilidad es una medida de cómo de seguros estamos de que un suceso ocurra y la segunda es que las creencias previas influencian las creencias siguientes. Se diferencia del paradigma frecuentista en que para este la probabilidad es el límite de la frecuencia del suceso cuando se realiza el experimento infinitas veces. La principal ventaja del paradigma bayesiano frente al frecuentista es que este nos permite hacer afirmaciones probabilísticas acerca de cosas que no están sujetos a variabilidad aleatoria. Por ejemplo: «La probabilidad de que lloviera el siete de mayo de 1840 es 0.45», esto hace referencia a la certeza que yo tengo de que la proposición sea verdadera y no hace referencia a ninguna frecuencia relativa. Esto permite que se puedan hacer afirmaciones probabilísticas de los parámetros de un modelo bayesiano y hacer inferencia de estos mediante distribuciones de probabilidad.

El paradigma bayesiano parte siempre de una hipótesis H, que se puede interpretar como conocimiento previo que se tiene sobre el problema o como unas ideas que se asumen ciertas. A partir de la hipótesis H se asignan valores $p(A|H)$ a declaraciones A sobre el mundo. Algunos ejemplos de declaraciones son «Mañana va a llover» o «El valor de salida de la función $f(x)$ evaluada en a es b». De esta manera $p(A|H)$ es el número que le asignamos a lo seguros que estamos de que ocurra A, de acuerdo a la hipótesis H.

Existen dos reglas básicas de la probabilidad a tener en cuenta, siendo la primera la regla del producto,

$$p(A, B|H) = p(A|B, H)p(B|H) = p(B|A, H)p(A|H), \tag{3.1}$$

donde $p(A, B|H)$ representa la intersección de las creencias A y B dada la hipótesis H, mientras que $p(A|B, H)$ es la probabilidad de A condicionada a la hipótesis H suponiendo que también ha ocurrido B. La segunda regla es la regla de la suma,

$$p(A|H) = \sum_{B} p(A, B|H), \tag{3.2}$$

donde el sumatorio representa la suma de todos los valores de una variable discreta; en el caso en el que la variable sea continua sería necesario cambiarlo por una integral.

Con estas reglas en mente podemos ver cómo ir actualizando nuestras creencias sobre las declaraciones a medida que vamos recopilando información. Por ejemplo,

después de observar un suceso S podemos actualizar nuestra creencia sobre la declaración A de la siguiente manera:

$$p(A|S,H) = \frac{p(S|A,H)p(A|H)}{p(S|H)}. \tag{3.3}$$

La interpretación de cada factor es la siguiente:

- El factor $p(A|S,H)$ es la distribución a posteriori y se interpreta como la plausibilidad actualizada tras haber observado S.
- El factor $p(A|H)$ es la distribución a priori y se interpreta como la plausibilidad previa a haber observado S.
- El factor $p(S|A,H)$ es la verosimilitud y se interpreta como la plausibilidad de que el suceso S ocurra en el caso en el que se da también la declaración A.
- El factor $P(S|H)$ es la evidencia y la interpretación es cómo de plausible es que ocurra el suceso S dada la hipótesis. Este factor es el que normaliza la distribución a posteriori.

La ecuación 3.3 representa el teorema de Bayes que describe cómo actualizar la plausibilidad de las declaraciones según se van observando nuevos sucesos.

Nosotros usaremos el teorema de Bayes para calcular la distribución a posteriori de los parámetros θ dados los datos de entrenamiento D y la distribución a priori $p(\theta|M)$ donde M refleja la estructura del modelo que se está usando. Intuitivamente, vamos aprendiendo sobre los parámetros θ cada vez que vemos un nuevo dato de entrenamiento. La ecuación queda así:

$$p(\theta|D,M) = \frac{p(D|\theta,M)p(\theta|M)}{p(D|M)} = \frac{p(D|\theta,M)p(\theta|M)}{\int_\theta p(D|\theta,M)p(\theta|M)d\theta}. \tag{3.4}$$

Podemos reescribirla separando las entradas de entrenamiento D_x de las etiquetas correspondientes D_y de manera que nos queda tal que así:

$$\begin{aligned} p(\theta|D,M) &= \frac{p(D_x,D_y|\theta,M)p(\theta|M)}{\int_\theta p(D_x,D_y|\theta,M)p(\theta|M)d\theta} \\ &= \frac{p(D_y|D_x,\theta,M)p(D_x|\theta,M)p(\theta|M)}{\int_\theta p(D_y|D_x,\theta,M)p(D_x|\theta,M)p(\theta|M)d\theta} \\ &= \frac{p(D_x|M)p(D_y|D_x,\theta,M)p(\theta|M)}{p(D_x|M)\int_\theta p(D_y|D_x,\theta,M)p(\theta|M)d\theta} \\ &= \frac{p(D_y|D_x,\theta,M)p(\theta|M)}{\int_\theta p(D_y|D_x,\theta,M)p(\theta|M)d\theta}. \end{aligned} \tag{3.5}$$

En el penúltimo paso se asume que D_x y θ son independientes de manera que $p(D_x|\theta,M) = p(D_x|M)$ y por tanto se puede cancelar el término al sacarlo de la integral.

En modelos complejos esta distribución a posteriori de los parámetros es muy compleja y hace que calcularla de manera exacta sea un problema casi imposible. Debido a esto lo más común es aproximar la distribución con cadenas de Markov o métodos variacionales.

Capítulo 4

Redes neuronales bayesianas

El estudio de las redes neuronales es un subcampo del aprendizaje automático que está experimentando un crecimiento exponencial y en los últimos diez años ha demostrado poder resolver una amplia gama de problemas incluyendo la visión por ordenador, el procesamiento de lenguaje natural o el reconocimiento de voz. Las redes neuronales están inspiradas en la estructura del cerebro humano donde un conjunto de neuronas interactúan para aprender representaciones complejas analizando un gran volumen de datos.

Las neuronas son las unidades básicas de la red y generalmente se organizan en capas. La primera capa será un vector del tamaño de la entrada y la última otro vector del tamaño del dominio de las predicciones. Estas dos capas están conectadas por una o más capas ocultas de manera que las neuronas de una capa tienen un enlace ponderado con las neuronas de la siguiente capa. De manera más formal, la red neuronal está construida con una capa de entrada l_0, una sucesión de capas ocultas $l_i, i = 1, \ldots, n-1$, y una capa de salida l_n. Cada capa l_i se construye con una transformación lineal de la capa anterior l_{i-1} seguida de una operación no lineal llamada función de activación.

$$\begin{aligned} l_0 &= x, \\ l_i &= s_i(W_i l_{i-1} + b_i) \quad \forall i \in [1, n], \\ y &= l_n \end{aligned} \tag{4.1}$$

Los parámetros son $\theta = (W, b)$ donde W es el conjunto de las matrices que contienen los pesos $\{W_1, W_2, \ldots, W_n\}$ y b el conjunto de vectores que contienen los sesgos $\{b_1, \ldots, b_n\}$.

Las redes neuronales clásicas ajustan los pesos para conseguir la mejor aproximación posible a la función deseada. Esto se consigue optimizando una función coste. Una de las funciones coste más comunes es el error cuadrático medio, pero se podría elegir cualquier otra. La red neuronal intenta minimizar la función coste usando algoritmos como el descenso de gradiente. Al final del entrenamiento habrá encontrado un conjunto de valores θ^* para los diferentes pesos que minimizan la función coste y con los que operará para hacer predicciones $y = f(x; \theta^*)$, donde f es la función que produce la red neuronal, y es la salida y x es la entrada.

4.1. Marginalización

Vemos que el paradigma clásico de las redes neuronales usa un único conjunto de valores para hacer predicciones, el óptimo según la función coste y los datos de entrenamiento, pero podemos hacer varias observaciones. Es razonable pensar que existen otros pesos distintos del óptimo que también son capaces de aproximar correctamente la función objetivo o que para ciertas predicciones incluso son capaces de hacer aproximaciones mejores que los pesos elegidos, de manera que no podemos estar seguros de que los pesos elegidos como óptimos sean los correctos. Una forma de solucionar esto es que para cada parámetro asignemos a cada posible valor que pueda tomar una probabilidad de que efectivamente sea el valor correcto. Lo que estaremos haciendo en ese caso es asignar una distribución de probabilidad $p(\theta)$ a los parámetros θ. La distribución de los parámetros dependerá del problema que queramos resolver, que está reflejado en los datos de entrenamiento que tenemos para entrenar la red de manera que la distribución que buscamos es $p(\theta|D)$. El paradigma bayesiano incorpora estas ideas, planteando un punto de vista diferente a la optimización a la hora de analizar los datos de entrenamiento y hacer predicciones.

Las predicciones $p(y|x, D)$ serán el resultado de ponderar a lo largo de todo el dominio de θ cada solución $p(y|x, \theta)$ por la probabilidad a posteriori de dicho peso sobre los datos de entrenamiento $p(\theta|D)$. La siguiente ecuación representa justamente esta idea:

$$p(y|x, D) = \int_{\theta} p(y|x, \theta)p(\theta|D)d\theta. \tag{4.2}$$

La interpretación de los diferentes términos es la siguiente:

- El factor $p(y|x, D)$ es la probabilidad de que la predicción de y (valor de regresión, valor de clase, ...) sea correcta, dada la entrada x y los datos de entrenamiento D
- El factor $p(y|x, \theta)$ es la probabilidad de que la predicción y sea correcta según unos pesos concretos θ y la entrada x. En la práctica es una distribución dependiente del valor $f(x; \theta)$ que se elige a la hora de diseñar la red.
- El factor $p(\theta|D)$ es la probabilidad de que esos pesos sean los correctos dada la información que tenemos sobre la función que queremos aproximar, es decir, los datos de entrenamiento D. Este factor es la distribución a posteriori que hemos visto en la sección sobre el teorema de Bayes.

La ecuación 4.2 representa la ponderación bayesiana de modelos y surge de combinar la regla de la suma y la regla del producto de la probabilidad. De la regla del producto y asumiendo independencia se obtiene

$$\int_{\theta} p(y|x, \theta)p(\theta|D)d\theta = \int_{\theta} p(y, \theta|x, D)d\theta \tag{4.3}$$

Aplicando a continuación la regla de la suma en el caso de una variable continua se obtiene el resultado de la ecuación 4.2

$$\int_{\theta} p(y, \theta|x, D)d\theta = p(y|x, D). \tag{4.4}$$

Vemos como las predicciones $p(y|x, D)$ ya no dependen de los pesos θ sino únicamente de los datos de entrenamiento D y de la entrada x. A esto se le llama marginalización de los parámetros θ ya que la distribución de las predicciones ya no está condicionada sobre θ.

4.2. Redes neuronales estocásticas

Para capturar la idea de no saber con seguridad cuáles son los parámetros correctos en una red neuronal, podemos construir la red introduciendo componentes estocásticos en los pesos o en las funciones de activación. Para resolver un problema de regresión con pesos estocásticos podría definirse una red de la siguiente manera:

$$\begin{aligned} \theta &\sim p(\theta|D) = N(\mu, \sigma_\theta) \\ y &\sim p(y|x, \theta) = N(f(x; \theta), \sigma_y), \end{aligned} \tag{4.5}$$

donde la primera distribución es de donde se muestrean los parámetros de la red y la segunda es de donde se muestrea la predicción final.

Presumiblemente las muestras de los parámetros θ cogidas de la distribución serán diferentes entre sí, lo que permite simular diferentes parametrizaciones de la red. Lo que en conjunto hemos definido al introducir estos componentes estocásticos es una red neuronal estocástica o bayesiana. Nótese además que la salida de la red no solo depende de los parámetros y la entrada, es decir, la distribución $p(y|x, \theta)$ no es igual a $f(x; \theta)$ sino a una distribución dependiente de la salida de la red. Esto es porque en caso de que $p(y|x, \theta) = f(x; \theta)$, el modelo no estaría teniendo en cuenta valores atípicos. En el ejemplo anterior $f(x; \theta)$ devuelve un valor que luego se toma como media de una distribución normal. En otros casos $f(x; \theta)$ podría devolver tanto la media μ como la desviación σ de manera que $p(y|x, \theta) \sim N(y|\mu, \sigma)$. En resumen, a la hora de definir la red estocástica no solo hay que definir la distribución de los pesos sino también una distribución q para el resultado final de la ejecución de la red que dependerá del valor $f(x; \theta)$.

A la hora de implementar una red neuronal bayesiana la integral de la ecuación 4.2 no se calcula de manera exacta. En la práctica se aproxima muestreando un número de pesos $\theta_i \sim p(\theta|D)$ para $i = 1, \ldots, N$ de la distribución a posteriori que son usados para computar una serie de predicciones $y_i = q(f(x; \theta_i))$, donde q es una distribución que se elige en el diseño de la red. La predicción final de la red será una combinación de las diferentes predicciones hechas por las distintas configuraciones de los parámetros. Por ejemplo en un problema de regresión, la predicción final $\hat{y}$ suele ser la media aritmética. El algoritmo 1 resume todo este proceso.

Además la distribución a posteriori de la cual muestreamos θ_i también es una aproximación. Sin embargo, generalmente no nos importará tener una aproximación poco precisa, sino una aproximación que sea muy cercana a la distribución real en las regiones que más contribuyen a la marginalización. Esto se debe a que nuestro verdadero interés es aproximar correctamente la ecuación 4.2, y no tanto la distribución a posteriori en sí misma. A la hora de aproximar correctamente la ponderación bayesiana de modelos es importante tener diversidad funcional en las hipótesis de los pesos. Si dos configuraciones de pesos θ_i y θ_j proporcionan una verosimilitud

Algoritmo 1 Red neuronal bayesiana

Define $p(\theta|D) = \frac{p(D_y|D_x,\theta,M)p(\theta|M)}{\int_\theta p(D_y|D_x,\theta,M)p(\theta|M)d\theta}$
for $i = 0$ **to** N **do**
  Muestrea $\theta_i \sim p(\theta|D)$
  Muestrea $y_i \sim q(f(x;\theta_i))$
end for
$\hat{y} = \frac{1}{N}\sum_{i=1}^{N} y_i$
return $\hat{y}$

alta y por tanto una densidad alta en la distribución a posteriori pero dan como resultado funciones muy similares $f(x;\theta_i)$, $f(x;\theta_j)$ entonces habrá redundancia en la ponderación de modelos y la segunda configuración de los pesos no tendrá una gran contribución a la hora de estimar la integral de la ecuación 4.2. Justo por eso lo que se busca son pesos que resulten en funciones significativamente diferentes ya que aportan más información a la hora de aproximar correctamente la ecuación 4.2.

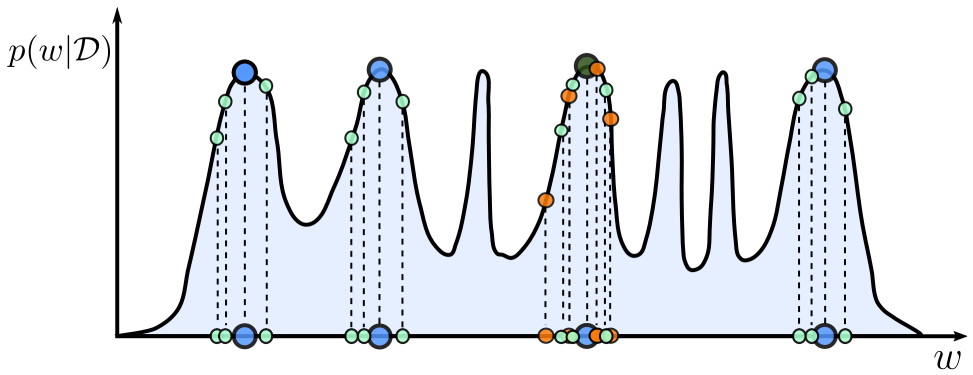


Figura 4.1: Posible distribución a posteriori $p(w|D)$ de un parámetro w. Fuente: Wilson et al. (2022).

En la figura 4.1 podemos ver una posible distribución a posteriori de un parámetro w. Los valores de w señalados en naranja están todos en una misma zona de alta densidad y por tanto las funciones que genera en el modelo son parecidas, pero los puntos naranjas nos están dando información de una única zona de alta densidad y por tanto el resultado de aproximar la integral 4.2 solo usando estos puntos no captura la realidad de la distribución a posteriori. Sin embargo, los puntos en verde claro se encuentran en torno a diferentes máximos de la distribución que corresponden a funciones que de manera diferente se ajustan a los datos dándonos la diversidad funcional que buscábamos. De esta manera la aproximación de la ecuación 4.2 se ajustará más al resultado real.

En la práctica, las redes neuronales bayesianas consisten en ponderar varias hipótesis para los pesos, varias parametrizaciones del modelo, para dar una mejor predicción. Esta idea de combinar varias hipótesis generalmente se conoce como conjuntos de modelos y hablaremos de ellos más adelante.

4.3. Distribución a priori

El primer paso para poder calcular la distribución a posteriori es elegir la distribución inicial de los parámetros de nuestra red, es decir, debemos elegir la distribución a priori $p(\theta)$. Esto no es una tarea intuitiva debido a que no es trivial deducir cuál debería ser esta distribución para que el modelo generalice bien.

Elegir una buena distribución a priori es una tarea muy relevante para conseguir una buena generalización. Como explicamos en la introducción, la generalización de un modelo depende de su soporte y su sesgo inductivo. La estructura de una red neuronal nos da mucha flexibilidad y soporte en $f(x;\theta)$, pero una buena distribución a priori $p(\theta)$ es esencial para concentrar una mayor masa del sesgo inductivo en conjuntos de datos relevantes y que por tanto generalice bien. Con una buena distribución a priori la distribución de las funciones $p(f(x;\theta))$ dará un mayor peso a las funciones más relevantes para nuestro problema.

Una práctica común es usar como distribución a priori una distribución normal centrada en el cero y una desviación típica σ.

$$p(\theta) = N(0, \sigma). \tag{4.6}$$

Con esta distribución estamos reflejando que creemos que los valores de los parámetros se encuentran cercanos y en torno al cero. Podemos variar el valor de σ en función de cómo de probable creemos que haya valores lejanos al cero. Esta distribución a priori es equivalente a la regularización l_2 en redes neuronales clásicas. La regularización consiste en añadir un término extra en la función de coste que se intenta optimizar para penalizar valores muy altos en los parámetros, justamente lo mismo que hacemos cuando establecemos que la distribución a priori sea $N(0, \sigma)$. En el caso de la regularización l_2 este término es:

$$\alpha \frac{1}{2|\theta|} \sum_{i=1}^{|\theta|} \theta_i^2, \tag{4.7}$$

donde α es un parámetro que sirve para ponderar el peso de la regularización en la función coste. Una distribución normal con desviación típica σ es equivalente a una regularización con $\alpha = 1/\sigma$.

Aunque el uso de la distribución normal está ampliamente extendido no existen argumentos teóricos que indiquen que sea mejor que otras distribuciones. Su uso está relacionado con la propiedades matemáticas de la distribución normal y con su simplicidad y conveniencia en algoritmos de aprendizaje automático.

4.4. Distribución a posteriori

En el apartado de redes estocásticas hemos visto un algoritmo (algoritmo 1) que no tenía una fase de entrenamiento debido a que lo único que necesitaba era tomar muestras de la distribución a posteriori $p(\theta|D)$. Sin embargo tomar muestras de esta distribución generalmente no es fácil. Recordemos que la distribución a posteriori es

dada por la siguiente fórmula:

$$p(\theta|D,M) = \frac{p(D_y|D_x,\theta,M)p(\theta|M)}{\int_\theta p(D_y|D_x,\theta,M)p(\theta|M)d\theta}. \tag{4.8}$$

Los factores $p(D_y|D_x,\theta,M)$ y $p(\theta|M)$ son fáciles de calcular. El primero es la distribución que hayamos definido en nuestra red para muestrear y, por ejemplo $N(f(D_x;\theta),\sigma)$. El segundo es la distribución a priori que hayamos decidido usar. Sin embargo el término $\int_\theta p(D_y|D_x,\theta,M)p(\theta|M)d\theta$ suele ser excesivamente complicado de calcular además de que aún en el caso de poder calcularlo, muestrear de la distribución exacta es muy difícil debido a su alta dimensionalidad. Los dos métodos más usados para muestrear de esta distribución son los métodos de Montecarlo basados en cadenas de Markov (Monte Carlo Markov chains, MCMC), una familia de algoritmos que de manera exacta muestrean de la distribución a posteriori, y el método de inferencia variacional que nos permite hallar una distribución de la que podemos muestrear que se aproxima a la distribución a posteriori.

4.4.1. Métodos Montecarlo basados en cadenas de Markov

La idea de los métodos MCMC es la de construir una secuencia de muestras de manera que una muestra S_i se construya solo a partir de la anterior S_{i-1}. La principal desventaja de usar este método es que muestras consecutivas pueden estar correlacionadas lo que significa que no son muestras tomadas de manera independiente. La solución es generar un conjunto grande Θ para luego tomar un subconjunto de manera que las muestras sean aproximadamente independientes y que se ajusten a la distribución a posteriori que buscamos. El conjunto Θ se debe guardar en memoria lo que conlleva un gasto demasiado grande para ciertas arquitecturas de redes con muchos parámetros.

A pesar de las desventajas que tienen, los métodos MCMC son los mejores y más populares para obtener de manera exacta muestras de la distribución a posteriori en algoritmos bayesianos. Entre ellos el algoritmo Metropolis-Hasting es especialmente relevante debido a que no se necesita conocer con exactitud la distribución de la que se desea muestrear sino que con conocer una función $f(\theta)$ que sea proporcional a la distribución es suficiente. De esta manera no hace falta calcular la evidencia $\int_\theta p(D_y|D_x,\theta)p(\theta)d\theta$ para obtener muestras de la distribución a posteriori, simplemente definiremos la función $f(\theta) = p(D_y|D_x,\theta)p(\theta)$ para tener una función proporcional a ella.

El algoritmo Metropolis-Hasting empieza con una elección aleatoria para θ_0. A partir de θ_0 se construye un nuevo candidato θ' usando una distribución $Q(\theta'|\theta)$ previamente definida. Algunas elecciones comunes para la distribución Q son una distribución normal $N(\theta,\sigma)$ o una distribución uniforme $U(\theta-\epsilon,\theta+\epsilon)$ centradas en la muestra anterior. En el caso en el que θ' sea una muestra más probable que la anterior acorde a la distribución de la que queremos muestrear entonces θ' es aceptada. Esto ocurre cuando $p(\theta'|D) > p(\theta_n|D)$, pero si en ambos lados de la inecuación multiplicamos por el factor de la evidencia nos queda $f(\theta') > f(\theta_n)$ que como hemos visto es más fácil de calcular. En el caso en que θ' no sea más probable que la muestra anterior todavía tiene cierta probabilidad p de ser aceptada. A p se le

denomina el factor de aceptación que será 1 cuando $f(\theta') > f(\theta_n)$ y será menor que 1 en caso contrario. Si la distribución $Q(\theta'|\theta_n)$ no es simétrica el factor de aceptación no solo dependerá de los valores de f sino que también se le deberá aplicar un factor de corrección que depende de la distribución Q elegida como se puede ver en el algoritmo 2. Cuando Q es simétrica al algoritmo simplemente se le llama método Metropolis: como se cumple que $Q(\theta'|\theta_n) = Q(\theta_n|\theta')$, el factor de aceptación p se simplifica ya que el factor de corrección que se aplicaba pasa a valer 1. Las muestras que no son aceptadas son descartadas. El algoritmo 2 es pseudo-código para la idea descrita.

Algoritmo 2 Metropolis-Hasting

Muestrea $\theta_0 \sim$ DISTRIBUCIÓN INICIAL
$n = 0$
while $n <= N$ **do**
  Muestrea $\theta' \sim Q(\theta'|\theta_n)$
  $p = \texttt{min}\left(1, \frac{Q(\theta'|\theta_n)}{Q(\theta_n|\theta')} \frac{f(\theta')}{f(\theta_n)}\right)$
  Muestrea $k \sim Bernoulli(p)$
  if k **then**
    $\theta_{n+1} = \theta'$
    $n = n + 1$
  end if
end while

La dispersión de $Q(\theta'|\theta)$ tiene que ser ajustada correctamente. Si es demasiado pequeña las muestras estarán muy correlacionadas y no serán muestras independientes. Si es muy grande el índice de rechazo será también demasiado grande. Sin embargo algoritmos como el algoritmo Hamiltoniano de Monte Carlo (HMC) proponen otros métodos para obtener θ' de manera que minimizan el número de muestras rechazadas. El algoritmo más usado hoy en día es el algoritmo No-U-Turn sampler que es una mejora del HMC permitiendo que los hiperparámetros del algoritmo se ajusten automáticamente en vez de tener que ajustarlos de manera manual.

4.4.2. Inferencia variacional

Los métodos MCMC son los mejores para obtener una muestra exacta de la distribución a posteriori. Sin embargo la falta de capacidad de para adaptarse a modelos más grandes ha hecho que no sean tan populares en los algoritmos de redes neuronales bayesianas. La inferencia variacional, que sí es capaz de adaptarse a redes mas complejas mejor que los algoritmos de MCMC, ha ganado una mayor popularidad.

La inferencia variacional no es un método exacto. En vez de tomar muestras exactas de la distribución a posteriori, la idea es tener una distribución $q(\theta; \phi)$, llamada la distribución variacional, parametrizada por un conjunto de parámetros ϕ. Los valores de ϕ son elegidos para que $q(\theta; \phi)$ sea lo más parecida a la distribución a posteriori exacta $p(\theta|D)$ que sea posible. Para medir cuánto se parece la distribución q a la distribución p generalmente se usa una medida de cercanía llamada divergencia

Kullback-Leibler (divergencia-KL) que se define de la siguiente manera:

$$D_{KL}(q(\theta;\phi)||p(\theta|D)) = \int_{\theta} q(\theta;\phi)\ \log\left(\frac{q(\theta;\phi)}{p(\theta|D)}\right) d\theta. \tag{4.9}$$

En la ecuación 4.9 todavía aparece el término $p(\theta|D)$. Para evitar que aparezca, de la fórmula anterior se puede derivar otra ecuación llamada ELBO (*evidence lower bound*):

$$\int_{\theta} q(\theta;\phi)\ \log\left(\frac{p(\theta,D)}{q(\theta;\phi)}\right) = \log(p(D)) - D_{KL}(q(\theta;\phi)||p(\theta|D)). \tag{4.10}$$

Nótese que en esta ecuación el término $\log(p(D))$ es constante al solo depender de la distribución a priori. Debido a esto minimizar la divergencia-KL, $D_{KL}(q(\theta;\phi)||p(\theta|D))$, es equivalente a maximizar la parte izquierda de la ecuación, el ELBO.

Los parámetros ϕ son aprendidos mediante inferencia variacional estocástica (SVI) que es de hecho descenso de gradiente estocástico sobre el ELBO para encontrar unos valores de ϕ que lo maximicen. Esto permite a la inferencia variacional adaptarse a conjuntos de datos mayores ya que no hace falta calcular el ELBO teniendo en cuenta todos los datos en cada iteración del entrenamiento sino que en cada iteración se optimiza teniendo en cuenta un subconjunto. Esto permite optimizar en épocas con lotes de datos lo que supone una ventaja para el rendimiento. Aún así hay que tener en cuenta que este proceso solo es para crear una distribución q que aproxima la distribución a posteriori y de la cual es más fácil muestrear, pero el muestreo todavía es necesario hacerlo

Capítulo 5

Ventajas del paradigma bayesiano

Una de las ventajas de las redes neuronales bayesianas es que otro tipo de modelos como las redes neuronales clásicas o los conjuntos de redes se pueden reinterpretar como redes neuronales bayesianas. En concreto, las redes neuronales clásicas se pueden interpretar como una red bayesiana cuya distribución a posteriori de los pesos $p(\theta|D) = \delta(x = \theta^*)$. La función δ es la función delta de Dirac que representa una distribución donde toda la masa está concentrada en un punto, en este caso en $\theta^* = argmax_\theta p(\theta|D)$. Por tanto, para cualquier $x \neq \theta^*$ el valor de la función delta de Dirac es 0. Es decir, una red neuronal clásica es en el fondo una red neuronal bayesiana donde se supone que el único valor posible de θ es θ^* y por tanto, $p(\theta^*|D) = 1$.

Cuando la distribución real a posteriori de los pesos dados los datos de entrenamiento es una distribución con un único máximo y toda la masa de la distribución está próxima y en torno a ese máximo, una red neuronal clásica se comportará de una manera similar a una red neuronal bayesiana. En cambio, cuando la distribución a posteriori no es como la descrita, aproximaciones muy gruesas como una distribución gaussiana ya son preferibles al uso de $\delta(x = \theta^*)$ como aproximación de la distribución a posteriori $p(\theta|D)$. En realidad, como ya dijimos al introducir el teorema de Bayes, la distribución a posteriori $p(\theta|D, M)$ donde M refleja la estructura de una red neuronal moderna, es muy compleja y de alta dimensionalidad además de altamente no convexa, lo que significa que tiene muchos máximos locales y muchas irregularidades. Esto es consecuencia de que en las redes neuronales modernas la verosimilitud $p(D|\theta)$ no favorece ninguna configuración concreta de θ; dicho de otra manera, hay muchas configuraciones de θ que dan soporte a los datos de entrenamiento. Como consecuencia de ello la distribución presenta varias zonas de atracción con una alta densidad de masa. Intuitivamente, esto quiere decir que hay muchos valores de θ que son capaces de adaptarse a los datos de entrenamiento gracias a la flexibilidad de la red neuronal. Esta es la situación donde mejores resultados da la marginalización y en general el paradigma bayesiano, ya que significará que tendremos muchas configuraciones diferentes de parámetros con buen rendimiento que podremos combinar para obtener resultados más precisos que con redes neuronales clásicas además de calcular de manera más precisa la incertidumbre de la predicción.

En esta sección explicaremos el concepto de incertidumbre en el aprendizaje

automático y qué ventajas presentan las redes bayesianas a la hora de aproximar esta incertidumbre. También haremos una comparación entre los conjuntos de modelos y la redes bayesianas determinando en qué aspectos se parecen y en cuáles se diferencian. También explicaremos el problema del doble descenso y cómo ciertos algoritmos bayesianos parecen mitigarlo por completo o al menos reducirlo. Por último discutiremos un poco más sobre la distribución a posteriori.

5.1. Incertidumbre

La incertidumbre es un concepto estrechamente relacionado con el teorema de Bayes y su interpretación bayesiana. Decíamos que la interpretación bayesiana del teorema de Bayes tiene que ver con cómo de seguros estamos de que un suceso ocurra, y no con el límite en el infinito de la frecuencia. En el contexto del aprendizaje automático la incertidumbre reflejará lo seguro que está el modelo de que su predicción sea correcta. La incertidumbre en una predicción implica que la información de la que disponemos es imperfecta o incompleta.

En el aprendizaje automático es importante tener una medida de cómo de seguro está el modelo de cada predicción que hace, es decir, una medida de incertidumbre. En concreto, la salida de una red neuronal dada una entrada no solo debería ser la predicción para la entrada sino además un valor de la incertidumbre de la predicción. Una buena estimación de la incertidumbre es particularmente importante en ciertas aplicaciones donde las predicciones incorrectas pueden tener consecuencias graves. Un ejemplo de esto es el clasificador que determina si un correo es o no *spam*, donde es crucial no clasificar erróneamente correos legítimos como *spam*.

Existen dos tipos de incertidumbre que afectan a los modelos de aprendizaje automático: la incertidumbre epistémica y la incertidumbre aleatoria. Imaginemos la siguiente sucesión $1, 2, 4, 8, \ldots$ ¿Cuál es el siguiente número en la sucesión? La falta de información acerca de la regla que genera la sucesión mantiene una incertidumbre epistémica acerca del siguiente número. Podríamos asumir que el siguiente número será 16, ya que la sucesión parece ser las potencias de dos, pero también podríamos considerar que el siguiente número es 10, si asumimos que la sucesión son los números positivos palíndromos en base 3. La incapacidad de saber con certeza cuál es el siguiente número en la sucesión se debe a la presencia de incertidumbre epistémica. Cuantos más números sepamos de la sucesión más se reducirá la incertidumbre epistémica sobre cuál es el siguiente número pero la incertidumbre epistémica nunca puede reducirse del todo a no ser que tengamos todos los infinitos valores o la regla de la sucesión.

La incertidumbre epistémica en un modelo surge de tener un número de datos de entrenamiento finito. La incertidumbre epistémica es reducible con nuevos datos de entrenamiento que revelen una nueva característica sobre la función que se está intentando aproximar al igual que con la sucesión de números. Las predicciones hechas sobre entradas muy alejadas de los datos de entrenamiento tendrán una incertidumbre epistémica más alta que aquellas predicciones que se hagan sobre entradas cercanas a los datos de entrenamiento. En la Figura 5.1 se puede observar como en los extremos donde no existen datos de entrenamientos el intervalo de confianza del

99 % es decir, la zona donde hay un 99 % de probabilidad de que una predicción sea correcta se ensancha. Esto es debido a la incertidumbre epistemológica, nos faltan datos en esa región para poder hacer mejores predicciones.

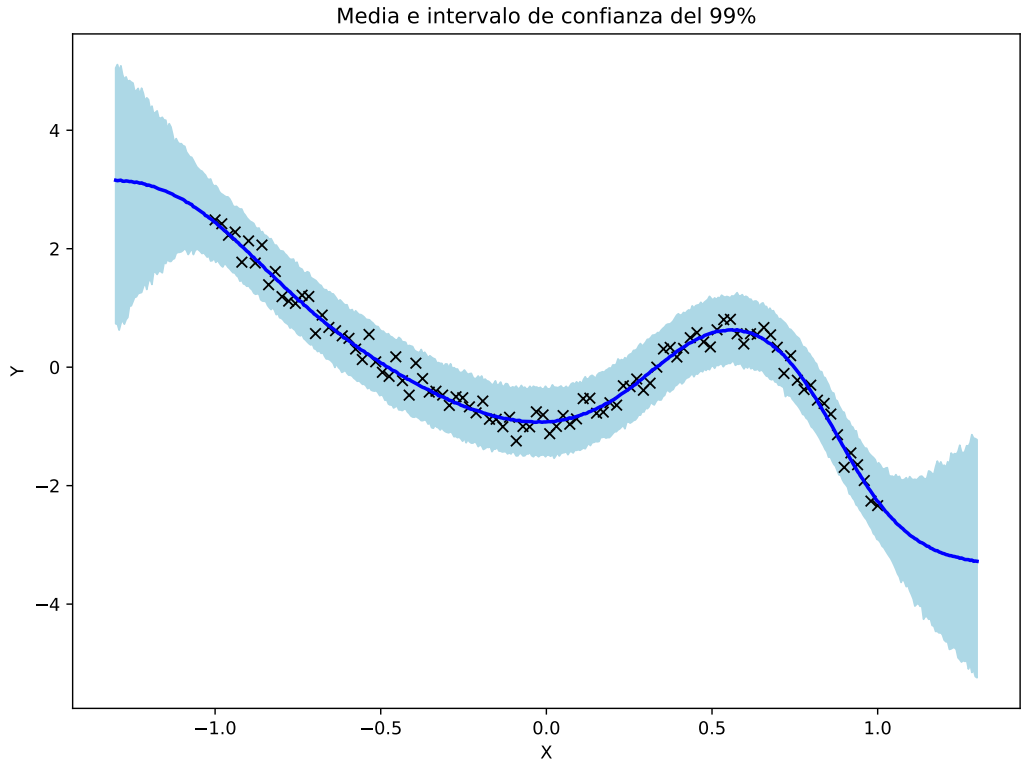


Figura 5.1: Regresión de un conjunto de datos. La función predicha es la línea azul oscuro. La región en azul claro muestra dónde está el intervalo de confianza del 99 %. Las cruces son los datos de entrenamiento.

La incertidumbre aleatoria, por otro lado, no se puede reducir y es inherente a la función que se intenta aproximar. Surge de la aleatoriedad de los procesos en la vida real y de la imperfección en la mediciones, es la incertidumbre que surge de información imperfecta. Matemáticamente se puede caracterizar añadiendo a la función que intentamos aproximar un factor de ruido

$$f'(x) = f(x) + \epsilon(x), \tag{5.1}$$

donde $\epsilon(x)$ es una variable aleatoria que representa la incertidumbre aleatoria de la función f que intentamos aproximar. En la Figura 5.1 se observa como los datos de entrenamiento no encajan perfectamente con la función de regresión sino que los puntos se encuentran algo por encima o algo por debajo. La incertidumbre total del modelo será la suma de la incertidumbre epistémica y la incertidumbre aleatoria.

Las redes neuronales no bayesianas tienden a sobrestimar cómo de buenas son sus predicciones, es decir, tienden a dar incertidumbres más bajas de lo que se esperaría viendo la precisión del modelo y por tanto son demasiado confiadas en sus predicciones. Esto surge de aproximar la distribución a posteriori $p(\theta|D)$ con $\delta(x = \theta^*)$ y como consecuencia suponer implícitamente que $p(\theta^*|D) = 1$. De esta manera ignoran la incertidumbre epistémica de no tener todos los datos para poder

distinguir con seguridad los pesos correctos. Seleccionando solo unos pesos se pierde la incertidumbre cuando otros pesos con probabilidad parecida están en desacuerdo con la salida de los pesos seleccionados para una cierta entrada.

La marginalización llevada a cabo por la redes neuronales permite dar una mejor aproximación de la incertidumbre de la predicción. La incertidumbre epistémica se puede calcular como la dispersión de la distribución de la predicción. Tiene sentido hacerlo de esta manera porque, como hemos dicho, la incertidumbre epistémica sobre una entrada de prueba cercana a los datos de entrenamiento debería ser pequeña; en cambio, cuando la entrada se aleja de los datos de entrenamiento la incertidumbre epistémica crece. Esto ocurre con la dispersión de la distribución de la predicción. Cuando la entrada es cercana a los datos de entrenamiento la dispersión de las predicciones es pequeña porque los pesos que tienen una mayor probabilidad de ser correctos coinciden en su resultado; justamente porque coinciden en los datos de entrenamiento, son los que tienen más probabilidad de ser correctos pero también coinciden para entradas cercanas a los datos de entrenamiento. En cambio la dispersión para entradas cada vez más lejanas a los datos de entrenamiento es también cada vez mayor ya que hay varias hipótesis para los pesos que son consistentes con los datos de entrenamiento y las diferentes hipótesis significan diferentes predicciones para la misma entrada. Esta mejor aproximación de la incertidumbre trae consigo una ventaja frente a las redes neuronales clásicas, y es que cuando se le da una entrada muy lejana a los datos de entrenamiento, la red neuronal bayesiana responderá con una incertidumbre muy alta en vez de dar una predicción casi a ciegas como hacen las redes clásicas. Esto permite que las redes neuronales bayesianas puedan entrenarse con conjuntos de datos pequeños sin que sufran sobreajuste.

La forma natural por tanto de calcular la incertidumbre de una predicción $\hat{y}$ es calcular la varianza de los resultados $y_i = Q(f(x;\theta_i))$ de distintas ejecuciones de la red con una misma entrada pero con las diferentes configuraciones de parámetros:

$$\sigma^2_{y|x,D} = \frac{1}{N-1}\sum_{i=1}^{N}(y_i - \hat{y})^2. \tag{5.2}$$

Cuando el problema es de clasificación, en cambio, el resultado de hacer la media de los modelos es una distribución de la probabilidad de cada clase de la clasificación, que ya se puede considerar como una medida de incertidumbre. La predicción final es la clase con una mayor probabilidad:

$$\hat{y} = argmax_i p_i \quad p_i \in \hat{p} = \frac{1}{N}\sum_{i=1}^{N} y_i. \tag{5.3}$$

En este caso y_i es un vector y por tanto el sumatorio es una suma de vectores. El vector resultante de hacer la media es el vector $\hat{p}$. La predicción final es el índice del valor máximo de $\hat{p}$, *i.e.* la clase con una mayor probabilidad de ser la correcta. La incertidumbre de la predicción viene dada por p_i.

5.2. Bayes contra conjuntos de redes

Imaginemos que tenemos una moneda sesgada que cae de cara un 51 % de las veces. Obviamente si tiramos la moneda una vez la probabilidad de que salga cara no es más que 0,51. Cuando tiramos la moneda n veces podemos calcular la probabilidad de que salga cara exactamente k veces tal que así:

$$p(k \text{ veces cara}) = \binom{n}{k} p^k (1-p)^{n-k}, \tag{5.4}$$

donde p es la probabilidad de que salga cara, en este caso 0,51. Ahora, lo que nos interesa es saber qué probabilidad hay de que la mayoría de tiradas sean cara cuando tiro la moneda n veces. Como cada tiro de moneda es independiente la fórmula es la siguiente:

$$p(\text{mayoría cara en } n \text{ tiradas}) = \sum_{k=\frac{n}{2}+1}^{n} \binom{n}{k} p^k (1-p)^{n-k}. \tag{5.5}$$

Lo interesante es que para $n = 1000$, a pesar de que el sesgo de la moneda sea solo del 51 %, la probabilidad de que la mayoría sean cara es aproximadamente 0,73.

Imaginemos ahora que en vez de tener una moneda sesgada lo que tenemos son 1000 clasificadores independientes con una precisión del 51 %. A pesar de que los clasificadores que tenemos son muy malos a la hora de hacer una predicción, si los ponemos a votar cuál es el resultado acertaremos en la predicción aproximadamente un 73 % de las veces. Esta es la idea que hay detrás de los conjuntos de modelos: combinar, a veces con ponderación, las predicciones de varios modelos lo más independientes posible para obtener mejores resultados. Vemos como los conjuntos consiguen una mayor precisión a costa de también necesitar una mayor capacidad de cómputo.

En la práctica los modelos no son completamente independientes por lo que no se puede usar el mismo razonamiento que con la moneda sesgada, pero se puede considerar una aproximación si la dependencia no es mucha. Para minimizar la dependencia de los modelos se usan varias técnicas como *bagging*, que consiste en entrenar cada modelo con un subconjunto de los datos de entrenamiento o *boosting*, que consiste en intentar que cada modelo mejore las predicciones del modelo entrenado previamente ponderando los datos de entrenamiento tal que los datos que se predicen de manera incorrecta tienen más peso en el modelo siguiente.

Ya hemos visto como en la práctica la marginalización consiste en la ponderación de modelos con diferentes hipótesis para los pesos, y como esto se puede entender como un conjunto de redes. Sin embargo, también los conjuntos se pueden interpretar desde el punto de vista bayesiano como hacíamos con la redes neuronales clásicas. Por ejemplo, decíamos que en los conjuntos es importante que los modelos sean lo más independientes posible para que la combinación sea enriquecedora. Esto se puede interpretar como la necesidad en las redes neuronales bayesianas de que las diferentes configuraciones de los pesos tengan diversidad funcional, es decir, reflejen hipótesis diferentes de los pesos que dan lugar a funciones diferentes, y no hipótesis

parecidas que dan lugar a funciones parecidas y por tanto no aportan casi a la ponderación bayesiana de modelos.

Sin embargo existe una diferencia fundamental entre la redes neuronales bayesianas y algunas formas de conjuntos. El paradigma bayesiano asume que solo una hipótesis para los parámetros es correcta pero como no tiene datos suficientes para distinguir cuál es hace una ponderación de las configuraciones más probables. Según vamos observando más datos, la distribución a posteriori va colapsando en torno a un único valor para los parámetros, una única hipótesis. Si la realidad de los datos es que son una combinación de diferentes hipótesis, entonces la red bayesiana puede empeorar su rendimiento según se van observando más datos. En este contexto, conjuntos que funcionen enriqueciendo el espacio de hipótesis no colapsarían de la misma manera que la marginalización y por tanto podrían dar mejor resultado que las redes bayesianas.

5.2.1. MultiSWAG

La mayoría de los métodos bayesianos de aprendizaje profundo consisten en aproximar la distribución a posteriori sobre los pesos θ en una única región de alta densidad, lo que limita la diversidad funcional en comparación con los conjuntos que exploran soluciones en diferentes regiones de alta densidad de la distribución a posteriori. Debido a esto los conjuntos logran una mejor aproximación de la marginalización que muchos algoritmos bayesianos. En Wilson et al. (2022) se propone un nuevo algoritmo llamado MultiSWAG que fusiona la idea de los conjuntos con los algoritmos bayesianos. MultiSWAG combina múltiples aproximaciones gaussianas entrenadas de manera independiente centradas en diferentes máximos con alta densidad en la distribución a posteriori. El algoritmo 1 que hemos visto donde se toman muestras de la distribución a posteriori para aproximar la distribución de predicción $p(y|x, D) \approx \frac{1}{N}\sum_{i=1}^{N} p(y|x, \theta_i)$ es el método de Montecarlo[1] para aproximar la integral de la ecuación 4.2.

Wilson et al. (2022) argumenta que el paradigma bayesiano va más allá de aproximaciones Montecarlo. MultiSWAG es un ejemplo de esto. En vez de muestrear de $p(\theta|D)$ mediante MCMC o inferencia variacional, aproxima $p(\theta|D)$ mediante un conjunto de distribuciones normales, que pondera y son las que usa para muestrear θ. MultiSWAG lleva los conjuntos y los algoritmos bayesianos un paso más allá porque no consiste en un conjunto de redes que usan optimización sino que es un conjunto de aproximaciones gaussianas de las distribución a posteriori que ya de por sí es una algoritmo bayesiano. De esta manera según Wilson et al. (2022) MultiSWAG generaliza mejor que los algoritmos Montecarlo basados en muestrear de la distribución a posteriori directamente.

5.3. Doble descenso

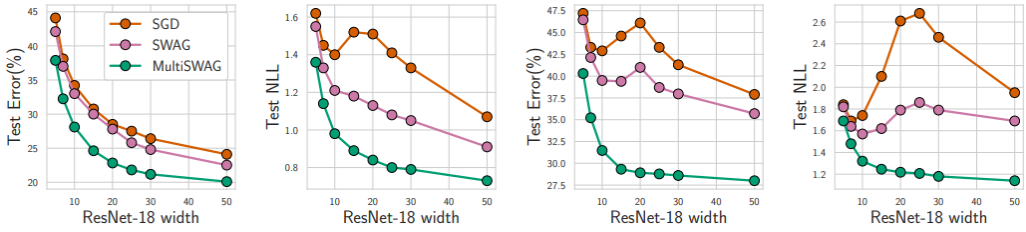
El efecto del doble descenso estudiado por primera vez por Opper et al. (1990) describe el descenso, ascenso y finalmente un segundo descenso del error de gene-

[1]No se debe confundir con el método MCMC para calcular la distribución a posteriori.

ralización de un modelo según aumenta su flexibilidad. El primer descenso junto con el primer ascenso se llama régimen clásico: modelos con mayor flexibilidad son capaces de capturar mejor la realidad de los datos hasta que empiezan a ajustarse demasiado y aumenta el error de generalización, lo que llamamos sobre-ajuste. Al segundo descenso en el error de la generalización se le llama régimen de interpolación moderno. Las causas del régimen de interpolación no se conocen y es un comportamiento misterioso dentro del aprendizaje automático.

Desde nuestra perspectiva probabilística de la generalización presentada en la introducción, un modelo que generaliza bien no debería sufrir doble descenso porque el sesgo inductivo debería priorizar hipótesis más simples que expliquen los datos antes que hipótesis que resulten en funciones muy complejas que tienden a provocar sobreajuste. Esto es lo que ocurre con los conjuntos y sobre todo con algoritmos bayesianos que usan varias zonas de alta densidad en la distribución a posteriori como MultiSWAG y que parten de una correcta distribución a priori. Es decir, estos modelos no sufren sobreajuste aun cuando sean altamente flexibles.

En Wilson et al. (2022) se comprueba esto midiendo el error y la probabilidad logarítmica negativa (negative log likelihood, NLL) en una red con una estructura ResNet-18 mientras aumenta su anchura. Comparan el resultado de entrenar la red usando MultiSWAG, SWAG (algoritmo de aproximación gaussiana en un único máximo) y con un algoritmo no bayesiano, en este caso descenso de gradiente estocástico.



(a) True Labels (Err) (b) True Labels (NLL) (c) Corrupted (Err) (d) Corrupted (NLL)

Figura 5.2: **(a)**: Error en los casos de prueba y **(b)**: pérdida NLL para una red ResNet-18 aumentando su anchura usando el dataset CIFAR-100. **(c)**: Error en los casos de prueba y **(d)**: pérdida NLL cuando un 20 % de las etiquetas son intercambiadas de manera aleatoria. Fuente: Wilson et al. (2022).

Lo primero que observamos es como la red entrenada con descenso de gradiente (SGD) sí que sufre doble descenso, sobre todo en NLL con datos corruptos. También se puede observar como SWAG reduce el doble descenso pero sin terminar de mitigarlo por completo. En cambio, MultiSWAG es capaz de acabar por completo con el doble descenso. El rendimiento de MultiSWAG aumenta siempre con el tamaño del modelo incluso con datos significativamente corruptos.

Esto muestra la importancia de hacer una marginalización multimodal. SWAG que es unimodal sufre doble descenso con un 20 % de datos corruptos mientras MultiSWAG no lo hace. Wilson et al. (2022) también mide como el doble descenso se alivia cada vez más con el aumento de modelos SWAG entrenados de manera independiente aliviándolo por completo en torno a los diez modelos.

5.4. Temperatura de la distribución a posteriori

La distribución a posteriori se puede pensar como la verosimilitud $p(D|\theta)$ multiplicada por la distribución a priori $p(\theta)$ normalizado por una constante Z tal que así:

$$p(\theta|D) = \frac{1}{Z}p(D|\theta)p(\theta), \tag{5.6}$$

pero en el ámbito de las redes neuronales bayesianas es típico usar una distribución a posteriori atemperada:

$$p(\theta|D) = \frac{1}{Z(T)}p(D|\theta)^{1/T}p(\theta). \tag{5.7}$$

El valor T es un parámetro de temperatura y $Z(T)$ es una función que indica cuál es el factor de normalización para la temperatura T. La temperatura controla cómo la verosimilitud y la distribución a priori interactúan para formar una distribución a priori:

- $T < 1$ corresponde a una distribución a posteriori fría. En este caso el exponente $1/T$ es mayor a uno y por tanto se le da más peso a los parámetros con una verosimilitud más alta. Esto significa que la distribución está más concentrada en torno a las soluciones con una mayor verosimilitud.
- $T = 1$ corresponde a la distribución a posteriori estándar.
- $T > 1$ corresponde a una distribución a posteriori caliente. En este caso el exponente de la verosimilitud es menor a uno y por tanto pierde peso. Esto significa que la distribución está más dispersa y su colapso en los puntos con mayor verosimilitud es más lento.

A atemperar la distribución a posteriori se le llama Bayes seguro.

Se ha visto que las redes bayesianas que mejor generalizan suelen usar una distribución a posteriori fría. En general, estas redes con una distribución a posteriori fría superan los resultados de redes neuronales clásicas entrenadas con optimización, mientras que redes bayesianas con $T = 1$ pueden dar peores resultados incluso que las clásicas. Nótese que esto solo significa que el error de generalización puede ser algo menor en una red no bayesiana, pero esta red seguirá teniendo problemas en el cálculo de la incertidumbre que la red bayesiana no tendrá.

Puede ser desesperanzador que para $T = 1$ el paradigma bayesiano no dé tan buenos resultados como los que se podrían esperar. Incluso podría ser sorprendente como una distribución a posteriori fría, que da más peso a la configuración de parámetros con mayor verosimilitud y se lo quita a las configuraciones más improbables, acercándose a lo que hace una red neuronal clásica, sea lo que mejor generaliza. Pero encontrarse con que usar $T < 1$ ayuda a la red neuronal bayesiana no es ni desesperanzador, ni sorprendente.

Las redes neuronales bayesianas son modelos que no están del todo bien especificados. La distribución a posteriori se forma a partir de un conjunto de datos y una combinación de creencias sobre el problema que se está tratando, materializándose en la distribución a priori de los parámetros y la estructura de la red utilizada.

Asumir que con esto el problema que queremos modelizar queda bien especificado es generalmente un error. Concretamente, en general la distribución a priori usada tiende a dar mucho más peso del deseado a soluciones que no concuerdan con los datos de entrenamiento. La forma de reflejar la creencia de que el modelo no está bien especificado es atemperando la distribución a posteriori. De esta manera una distribución a posteriori fría está más en línea con nuestras creencias sobre la solución que una distribución sin atemperar.

Capítulo 6

Comparación de la incertidumbre en un caso práctico

Hemos visto como una de las principales ventajas de las redes neuronales bayesianas es tener una forma natural de calcular la incertidumbre de una predicción. Mientras que una red neuronal basada en optimización será evaluada nada más que con una configuración de parámetros, la red bayesiana será evaluada en un número de diferentes configuraciones muestreadas de la distribución a posteriori. Si las diferentes muestras no presentan grandes diferencias funcionales no habrá grandes diferencias con la red clásica. Sin embargo, si las muestras presentan diversidad funcional entre ellas no solo es posible que la red sea más precisa en las predicciones, sino que la incertidumbre de la predicción se podrá obtener calculando la dispersión de las diferentes predicciones.

Para comprobar que lo anterior se cumple hemos planteado un pequeño experimento. El objetivo es observar cómo se comporta la incertidumbre de las predicciones de una red neuronal bayesiana en comparación con una red no bayesiana. Para esto hemos usado MNIST, una base de datos con imágenes de tamaño 28x28 en blanco y negro de los dígitos del cero al nueve. La idea del experimento es entrenar dos redes neuronales, una bayesiana y una no bayesiana, con todos los dígitos del cero al ocho pero sin enseñarles durante la fase de entrenamiento una imagen del dígito nueve. Por último comparamos cómo se comportan ambas redes neuronales cuando intentan predecir el dígito de una imagen que corresponde al dígito nueve. En concreto queremos comprobar si la incertidumbre de la predicción en la red bayesiana es más realista que la incertidumbre dada por la red no bayesiana, que como hemos visto, suele estar demasiado segura de sus predicciones y por tanto tiende a dar valores de incertidumbre por debajo de la incertidumbre real.

6.1. Detalles sobre la implementación

La red bayesiana la hemos implementado usando la librería NumPyro (NumPyro (2019b)) para Python que nos permite hacer programación probabilística gracias a que nos ofrece distribuciones ya implementadas y métodos de inferencia. Estas dos

funcionalidades son las que necesitamos para implementar nuestra red bayesiana.

El modelo cuenta con tres capas: el vector de entrada que tiene un tamaño de 784, una capa oculta de 50 nodos y el vector de salida que tiene tamaño 10. El vector de salida se transforma en una distribución categórica. Finalmente se calcula una muestra de esta distribución y el resultado es la salida final del modelo. Esto permite que el modelo genere diferentes predicciones incluso cuando la configuración de parámetros es la misma. Para terminar de especificar el modelo nos falta establecer la distribución a priori de los parámetros. Para esto hemos usado una distribución normal centrada en el cero y con la varianza igual a uno, que como hemos visto es la distribución a priori más usada si no se quieren añadir restricciones extra. Recordemos que además es equivalente a la regularización de parámetros l_2.

Para muestrear de la distribución a posteriori hemos usado el método de Montecarlo basado en cadenas de Markov. Este método requiere un periodo de calentamiento antes de converger y poder muestrear de la distribución deseada. El experimento lo hemos hecho con 1100 muestras, 100 de calentamiento y las otras 1000 son las muestras de la distribución a posteriori de nuestra red. Esto significa que para cada imagen de los datos de prueba vamos a tener 1000 predicciones. Como se trata de un problema de clasificación el dígito más repetido entre las 1000 predicciones será la predicción final para esa imagen. La incertidumbre será la frecuencia relativa de ese dígito entre las 1000 predicciones.

Por otra parte, para la implementación de la red neuronal clásica hemos usado la librería PyTorch. La estructura es la misma a la de la red bayesiana, las mismas tres capas y del mismo tamaño. La última capa se transforma en una distribución categórica igual que la red bayesiana, pero no se calcula una muestra sino que el índice del valor más alto será la predicción de la red y dicho valor la certidumbre de la predicción. La optimización de parámetros se ha llevado a cabo en 20 épocas con lotes de 64 imágenes utilizando descenso de gradiente estocástico (SGD).

La estructura de la red elegida es bastante simple pero como veremos en los resultados es lo suficientemente compleja para resolver el problema de clasificación que se le presenta. La elección de la estructura de la red ha estado muy condicionada al algoritmo utilizado para calcular la distribución a posteriori en la red bayesiana, ya que como veremos en los resultados incluso con una estructura de red tan simple le ha llevado un 96170 % más de tiempo de ejecución que a la optimización.

6.2. Resultados

Aunque el objetivo del experimento no sea comparar la precisión de las dos redes es interesante señalar que la red bayesiana ha dado unos resultados ligeramente superiores en las imágenes de prueba. Mientras que la red no bayesiana tuvo una precisión del $86,56\,\%$, la red bayesiana tuvo una precisión del $87,76\,\%$. Teniendo en cuenta que aproximadamente una décima parte de las imágenes con las que fueron probadas eran del dígito nueve, que no vio ninguna durante el entrenamiento, ambas redes tuvieron una precisión bastante alta. Concretamente, en la figura 6.1 podemos ver como ambas redes se comportan de manera muy similar en las predicciones. En ambas el dígito uno es el que mejor reconocen y el cinco el que peor. Además ambas

tienden a confundir el dígito nueve con el cuatro y el siete. Quizá la única diferencia significativa es que la red bayesiana confunde el nueve significativamente más con el cuatro de lo que lo hace la red no bayesiana que lo confunde más con otros dígitos como el tres o el cinco.

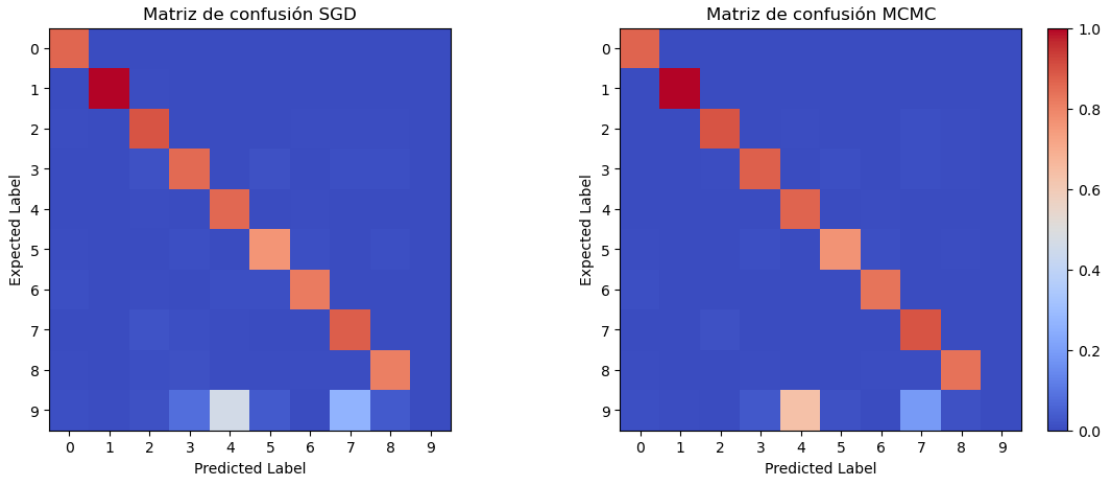


Figura 6.1: Matrices de confusión en mapa de colores. A la izquierda la correspondiente a la red entrenada con SGD. A la derecha la correspondiente a la red bayesiana.

Sin embargo donde sí se contemplan grandes diferencias es en la incertidumbre que presentan sobre las diferentes predicciones que hacen. En la figura 6.2 presentamos la distribución de la incertidumbre en cuatro casos diferentes. La primera imagen compara la incertidumbre de todas las predicciones de ambas redes; la segunda solo tiene en cuenta las predicciones en las que la red acierta; la tercera presenta la incertidumbre de las predicciones que no coincidieron con el dígito de la imagen y por último estudiamos qué ocurre con la incertidumbre cuando a la red se le enseña una imagen de un dígito que no se le ha enseñado durante el entrenamiento, en este caso el dígito nueve.

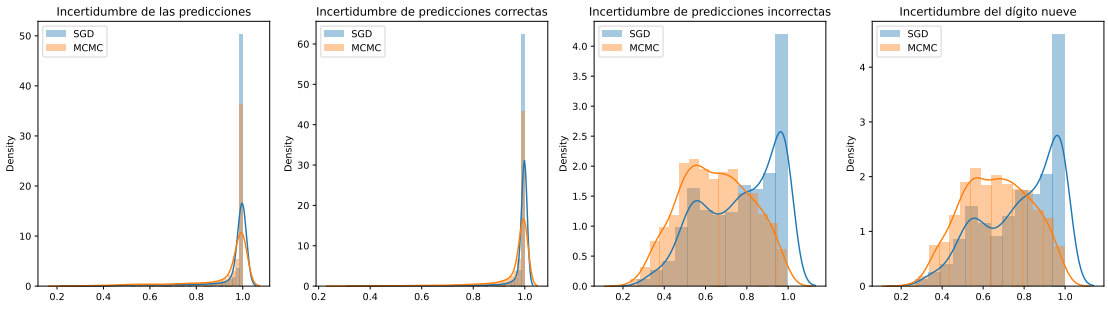


Figura 6.2: Distribuciones de la certidumbre. De izquierda a derecha: certidumbre de todas las clasificaciones, certidumbre de solo clasificaciones correctas, certidumbre de solo clasificaciones incorrectas, certidumbre de clasificaciones del dígito nueve.

Observamos como en las predicciones correctas ambas redes tienen una certidumbre muy alta, siendo prácticamente del 100 % en casi todas las imágenes co-

rrectamente clasificas. Comparando las dos redes se puede ver que la red bayesiana no tiene tantas certidumbres del 100 % como la red no bayesiana. Sin embargo, aún así a la mayoría de las imágenes que clasifica correctamente les asigna esta certidumbre. Se podría decir que la red bayesiana duda algo más que la no bayesiana en sus predicciones correctas pero nada demasiado significativo. Que la certidumbre de las clasificaciones hechas correctamente sean altas por sí solo no nos aporta mucho sobre cómo de bien aproxima la certidumbre el modelo, de hecho para estudiar cómo de precisa es la certidumbre que presenta un modelo es más interesante ver qué certidumbre da en los casos en los que se equivoca en la clasificación.

Es en la certidumbre que nos dan ambas redes cuando clasifican incorrectamente una imagen donde se ve la mayor diferencia entre ambos modelos. Como habíamos dicho en la sección dedicada a la incertidumbre, la red no bayesiana al entrenarse optimizando parámetros está demasiado segura de sus predicciones y por tanto no es capaz de detectar entradas con alta incertidumbre, o bien porque dicha entrada tiene mucha incertidumbre aleatoria, o bien porque tiene mucha incertidumbre epistémica, o ambas. El caso de la incertidumbre epistémica alta es el que estudiamos cuando intenta clasificar el dígito nueve que no ha visto en el entrenamiento. La optimización de parámetros es equivalente a aproximar la distribución a posteriori $p(\theta|D) \approx \delta(x = \theta^*)$. Esto se puede interpretar como estar seguro de que los parámetros correctos son los calculados en la optimización y no otros. La falsa seguridad en tener los parámetros correctos se traduce en una falsa seguridad en todas las predicciones del modelo.

Por otro lado frente a esto contrasta la distribución de la certidumbre dada por la red bayesiana. Podemos observar en la tercera y cuarta imágenes de la figura 6.2 que la red bayesiana presenta una certidumbre razonable cuando se equivoca en la predicción. Es importante darse cuenta de que la certidumbre que da la red bayesiana no es siempre baja. Cuando la predicción es correcta tiende a dar certidumbres altas lo que significa que es capaz de detectar entradas con alta incertidumbre y dudar en esos casos de su propia predicción, pero no dudar cuando la incertidumbre de la entrada es baja.

Estos son los resultados usando una red muy simple con únicamente una capa oculta de 50 nodos. Sin embargo la red bayesiana con esta estructura tan simple tardó en muestrear 1100 parametrizaciones 22142 segundos, lo que corresponde a 6 horas y 9 minutos. Puede no parecer mucho pero si tenemos en cuenta que la optimización en la red no bayesiana llevó 23 segundos significa que la red bayesiana tardó un 96170 % más que la red no bayesiana. Definitivamente el algoritmo MCMC presenta un incremento en el tiempo de entrenamiento inasumible para redes con muchos más parámetros y cabría estudiar cómo otros algoritmos tienen diferentes rendimientos a la hora de muestrear de la distribución a posteriori. Sin embargo, ese no es el objetivo de este trabajo.

Capítulo 7

Conclusiones y Trabajo Futuro

El paradigma bayesiano es uno de los más prometedores en el campo del aprendizaje automático debido a las ventajas que presenta frente a algoritmos clásicos basados en optimización de parámetros. Las principales ventajas son: una forma natural de medir la incertidumbre de las predicciones hechas por el modelo y la capacidad de los algoritmos bayesianos de evitar el sobre ajuste incluso en modelos extremadamente flexibles. Estas dos características surgen de tener en cuenta varias configuraciones de parámetros que resultan en diferentes funciones en la red con la capacidad de ajustarse a los datos de entrenamiento. La incertidumbre de una predicción surge de calcular la dispersión del resultado de estas funciones evaluadas en la entrada de esa predicción. Por otro lado se evita el sobre-ajuste al ponderar las diferentes predicciones para una entrada resultantes de las diferentes configuraciones de parámetros usados. En definitiva, el paradigma bayesiano se puede resumir en la ecuación 4.2. Sin embargo la principal desventaja es el alto coste de entrenamiento que requiere para calcular la distribución a posteriori. Un futuro trabajo podría ser el comparar los rendimientos de distintos algoritmos para calcular la distribución a posteriori u otros algoritmos bayesianos como MultiSWAG.

En este trabajo hemos presentado varias formas de aproximar los resultados de la ecuación 4.2 debido a que no es realista calcular la distribución de predicciones $p(y|x, D)$ de manera exacta. Hemos visto el método de Montecarlo donde se coge un conjunto de muestras de la distribución a posteriori y se hace la media para calcular la predicción final. También hemos mencionado un algoritmo, MultiSWAG, que combina diferentes aproximaciones gaussianas en torno a los puntos de mayor probabilidad. Por último hemos hecho un pequeño experimento para probar en la práctica estos conceptos y hemos podido observar el beneficio sobre la incertidumbre incluso en redes muy simples donde es posible que no existan tantas zonas de alta densidad en la distribución a posteriori como en redes neuronales altamente complejas.

Introduction

Machine learning is a branch of artificial intelligence that studies the methods by which a computer can learn by using training data. The goal is to approximate a function $f(x)$ that we do not know, but for which we have training data, usually a large amount of it and of the form $(x_i, f(x_i))$. The function f may correspond to a classification problem or to a regression problem. It will be a classification problem when the output can only take a finite number of values, for example, the function able to distinguish whether an image is a cat or not. A regression problem is one in which we try to predict a numerical value, for example the price of a house.

In order to approximate f we start from a set of functions $\{g(x;\theta)\}$. This set will be our model, where θ is a vector corresponding to the parameters of these functions. For example, suppose the model $M = \{g(x;\theta)\}$ such that $g(x;\theta) = w_0 + w_1 x$; in this case $\theta = (w_0, w_1)$ and depending on the values we give to θ we will have one or another function $g(x)$. Finally, there are algorithms such as gradient descent that are able to find a value for the parameters θ^* such that $g(x;\theta^*)$ approximates the objective function f. This process of finding the appropriate values for the parameters is what we mean when we say that the computer learns. Learning the problem is usually posed as an optimization problem where we try to reduce the difference between the predictions given by the approximate function g and the objective function f. Later we will see that the Bayesian paradigm is not interested in optimization but in parameter marginalization. We will see what marginalization is and the advantages of the computer learning in this way.

Two properties can be defined about a machine learning model that help us understand the behavior of the model: support and inductive bias (Wilson and Izmailov, 2022). Continuing with the previous example, the model M is said to have reduced support since it can only represent data sets that have a linear relationship, *i.e.* no matter what values its parameters take, the model M will never be able to represent data with quadratic or other types of relationships. The probability that the model M generates, if we sample randomly, a data set with quadratic relationship is 0. If $p(D|M)$ is the probability that a dataset D is generated by the model M, we define the support of the model as the datasets D such that $p(D|M) > 0$.

We define the inductive bias as the support distribution given by $p(D|M)$. Continuing with the example, the inductive bias will depend on how the parameters w_0 and w_1 are chosen a priori. If w_0 and w_1 are sampled from a uniform distribution, then the induced distribution will also be uniform, but if w_0 and w_1 are samples

from a normal distribution $N(0, 1)$ then it will be more likely to generate data with a linear relationship with slope close to 0 and less likely to generate datasets with linear relationship with a lot of slope.

7.1. Motivation

Wilson et al. (2022) suggest that these two properties, support and inductive bias, from a probabilistic point of view are primarily what generalization ability depends on, that is, the ability to do accurate and precise predictions on data not previously seen during training. Ideally we want a model to be flexible so that it supports any dataset no matter what type of relationship they have, or even be able to adapt to data that do not follow any relationship. In addition, we want the inductive bias to assign more probability to the hypothesis that we a priori think the problem we are solving has, *i.e.*, that the model quickly collapses around the solution. An example of a model that represents the a priori hypothesis is the convolutional structures used in neural networks for problems that have images as input.

It should be noted that the flexibility of a model does not necessarily imply that it tends to represent functions with high complexity. There are very flexible models capable of supporting virtually any data set but which have an inductive bias towards smooth and simple functions. An example of this is Gaussian processes with an RBF kernel, a machine learning model that has been shown to be able to represent random data while the functions represented a priori are not overly complex. Traditionally, the metrics used to assess the generalization capability of a model, generally similar to the support we defined above, have been one-dimensional and thus have failed to provide a complete picture of generalization. We will reason about Bayesian neural networks from the probabilistic point of view of generalization using a two-dimensional metric that takes into account both support and inductive bias.

As we have said before, learning from the Bayesian perspective is not an optimization process but a marginalization process. Intuitively this means that, instead of sticking with the optimal value of the parameters when making predictions, we consider all possible values that the parameters can take and weight each set of values according to how good the predictions given by that set are, given the training data. We will see that this approach has several advantages both in practice and when reasoning about the model.

One of the main advantages of Bayesian methods is that they provide a natural way to understand and quantify the uncertainty of neural networks and as a result tend to be less confident in their predictions than classical neural networks, which are inclined to overestimate how good their predictions are. They also allow them to distinguish between epistemic uncertainty and the random uncertainty of the data. This allows them to learn with a smaller number of training data without suffering from over-fitting. Finally, in practice Bayesian neural networks have demonstrated higher accuracy than classical neural networks on both sparsely corrupted data and data with a high degree of perturbation and have been shown to reduce or even end

the double descent problem as we will see below.

7.2. Objective

The aim of this thesis is to give an overview of Bayesian neural networks in a way that a general audience with a minimum knowledge of mathematics and computer science will be able to understand and to explain the ideas on which these models are based and some details about their practical implementation. We also aim at interpreting the classical models using Bayesian optimization and to present the advantages that Bayesian networks have over non-Bayesian ones. Finally, we test these ideas in a small experiment.

Conclusions and Future Work

The Bayesian paradigm is one of the most promising in the field of machine learning due to its advantages over point estimate algorithms based on parameter optimization. The main advantages are: a natural way to measure the uncertainty of predictions made by the model and the ability of Bayesian algorithms to avoid overfitting even in extremely flexible models. These two features arise from taking into account various parameter configurations that result in different functions in the network with the ability to fit the training data. The uncertainty of a prediction arises from calculating the dispersion of the result of these evaluated functions in the input of that prediction. On the other hand, over-fitting is avoided by weighting the different predictions for an input resulting from the different parameter settings used. In short, the Bayesian paradigm can be summarized in the equation 4.2. However, the main disadvantage is the high training cost required to compute the posterior distribution. A future work could be to compare the performances of different algorithms to compute the a posteriori distribution or other Bayesian algorithms such as MultiSWAG.

In this thesis we have presented several ways to approximate the results of the equation 4.2 because it is unrealistic to compute the prediction distribution $p(y|x, D)$ exactly. We have seen the Monte Carlo method where a set of samples is taken from the posterior distribution and averaged to compute the final prediction. We have also mention an algorithm, MultiSWAG, which combines different Gaussian approximations around the points of highest probability. Finally we have done a small experiment to test in practice these concepts and we have been able to observe the benefit on uncertainty even in very simple networks where it is possible that there are not as many high density areas in the posterior distribution as in highly complex neural networks.

Bibliografía

Belkin, M., Hsu, D., Ma, S. y Mandal, S. Reconciling modern machine-learning practice and the classical bias–variance trade-off. *Proceedings of the National Academy of Sciences*, vol. 116(32), páginas 15849–15854, 2019.

Blundell, C., Cornebise, J., Kavukcuoglu, K. y Wierstra, D. Weight Uncertainty in Neural Networks. 2015.

Fornet, P. Redes Bayesianas. 2023. `https://github.com/Pablof27/Redes-Bayesianas-tfg`.

Grünwald, P. y van Ommen, T. Inconsistency of Bayesian Inference for Misspecified Linear Models, and a Proposal for Repairing It. 2018.

Gustafsson, F. K., Danelljan, M. y Schön, T. B. Evaluating Scalable Bayesian Deep Learning Methods for Robust Computer Vision. 2020.

de Heide, R., Kirichenko, A., Mehta, N. y Grünwald, P. Safe-Bayesian Generalized Linear Regression. 2021.

Jospin, L. V., Laga, H., Boussaid, F., Buntine, W. y Bennamoun, M. Hands-On Bayesian Neural Networks—A Tutorial for Deep Learning Users. *IEEE Computational Intelligence Magazine*, vol. 17(2), páginas 29–48, 2022.

Kutyniok, G. The Mathematics of Artificial Intelligence. 2022.

Mackay, D. J. *Bayesian methods for adaptive models*. Tesis Doctoral, California Institue of Technology, 1992.

Mackay, D. J. Probable networks and plausible predictions — a review of practical Bayesian methods for supervised neural networks. *Network: Computation in Neural Systems*, vol. 6(3), páginas 469–505, 1995.

Maddox, W., Garipov, T., Izmailov, P., Vetrov, D. y Wilson, A. G. A Simple Baseline for Bayesian Uncertainty in Deep Learning. 2019.

Mohri, M. y Rostamizadeh, A. Rademacher Complexity Bounds for Non-I.I.D. Processes. En *Advances in Neural Information Processing Systems (NIPS 2008)*. Vancouver, Canada, 2009.

Nakkiran, P., Venkat, P., Kakade, S. y Ma, T. Optimal Regularization Can Mitigate Double Descent. 2021.

Nalisnick, E. T. *On Priors for Bayesian Neural Networks*. Tesis Doctoral, 2018.

Neal, R. M. *Bayesian Learning for Neural Networks*. Springer-Verlag, Berlin, Heidelberg, 1996. ISBN 0387947248.

NumPyro. Example: Bayesian Neural Network. 2019a. Accessed: 2023-08-30. `https://num.pyro.ai/en/stable/examples/bnn.html`.

NumPyro. Numpyro documentation. 2019b. Accessed 2023-08-30. `https://num.pyro.ai/en/stable`.

Opper, M., Kinzel, W., Kleinz, J. y Nehl, R. On the ability of the optimal perceptron to generalise. *Journal of Physics A: Mathematical and General*, vol. 23(11), página L581, 1990.

Pearce, T., Leibfried, F., Brintrup, A., Zaki, M. y Neely, A. Uncertainty in Neural Networks: Approximately Bayesian Ensembling. 2020.

Ritter, H., Botev, A. y Barber, D. A Scalable Laplace Approximation for Neural Networks. En *International Conference on Learning Representations*. 2018.

Roberts, D. A., Yaida, S. y Hanin, B. *The Principles of Deep Learning Theory*. Cambridge University Press, 2022. `https://deeplearningtheory.com`.

Wenzel, F., Roth, K., Veeling, B. S., Świątkowski, J., Tran, L., Mandt, S., Snoek, J., Salimans, T., Jenatton, R. y Nowozin, S. How Good is the Bayes Posterior in Deep Neural Networks Really? 2020.

Wikipedia contributors. Ensemble learning — Wikipedia, the free encyclopedia. 2023a. Accessed 30-August-2023. `https://en.wikipedia.org/w/index.php?title=Ensemble_learning&oldid=1172129024`.

Wikipedia contributors. Machine learning — Wikipedia, the free encyclopedia. 2023b. Accessed 30-August-2023. `https://en.wikipedia.org/w/index.php?title=Machine_learning&oldid=1172951363`.

Wilson, A. G. y Izmailov, P. Bayesian Deep Learning and a Probabilistic Perspective of Generalization. 2022.

Yang, W., Lorch, L., Graule, M. A., Srinivasan, S., Suresh, A., Yao, J., Pradier, M. F. y Doshi-Velez, F. Output-Constrained Bayesian Neural Networks. 2019.

Zhang, T. Information-theoretic upper and lower bounds for statistical estimation. *IEEE Transactions on Information Theory*, vol. 52(4), páginas 1307–1321, 2006.

Apéndice A

Visualización de la distribución a posteriori

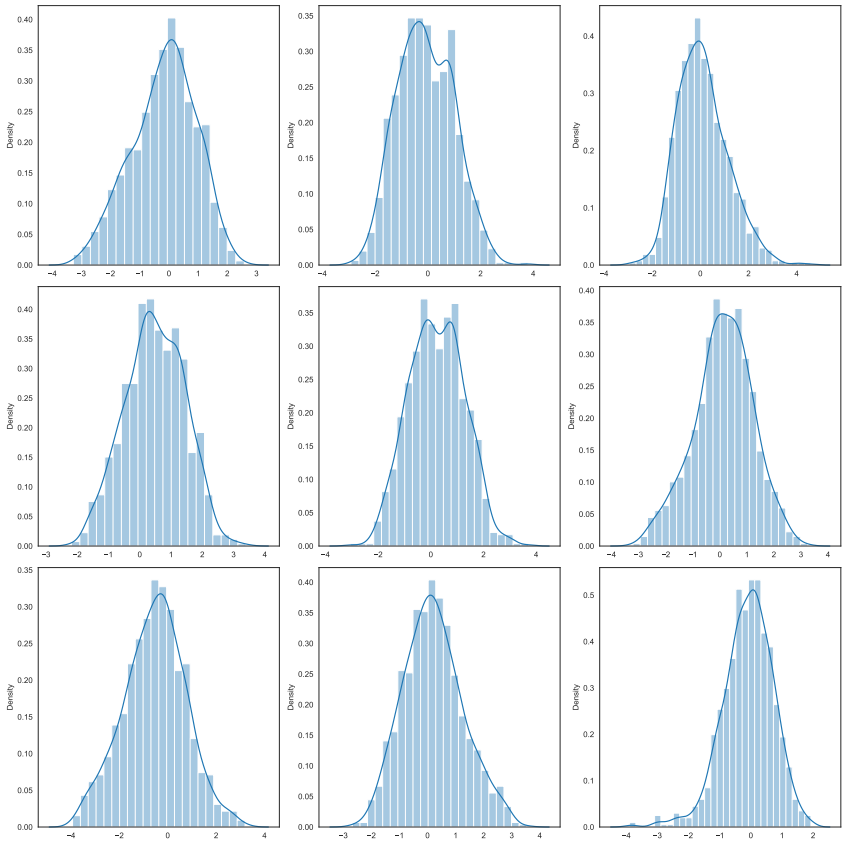


Figura A.1: Distribución de las muestras calculadas usando un algoritmo MCMC.

En el experimento del capítulo 6 vimos la incertidumbre que la red bayesiana presentaba tanto en las predicciones que hacía correctamente, como en las predicciones que no hacía correctamente, como en predicciones con entradas con alta incertidumbre epistémica. Vimos que la red bayesiana sabía cuándo debía dudar de sus predicciones, cosa que no sabía hacer la red entrenada con optimización. Explicamos

que esto se debía a una mejor aproximación en la distribución a posteriori de los parámetros.

En este apéndice queremos mostrar gráficamente la distribución de parámetros final después de muestrear con un algoritmo MCMC la distribución a posteriori. En la figura A.1 podemos ver la distribución de los nueve últimos parámetros de la red. Podemos observar cómo la distribución a priori ha conseguido que los pesos estén todos entorno al cero y no demasiado dispersos aunque el entrenamiento ha afectado a su forma. Como para cada parámetro se calcularon 1000 muestras, las gráficas están hechas con solo esas 1000 muestras y cabría esperar ligeros cambios en el caso de haber calculado más.

–¿Qué te parece desto, Sancho? – Dijo Don Quijote –
Bien podrán los encantadores quitarme la ventura,
pero el esfuerzo y el ánimo, será imposible.

Segunda parte del Ingenioso Caballero
Don Quijote de la Mancha
Miguel de Cervantes

–Buena está – dijo Sancho –; fírmela vuestra merced.
–No es menester firmarla – dijo Don Quijote–,
sino solamente poner mi rúbrica.

Primera parte del Ingenioso Caballero
Don Quijote de la Mancha
Miguel de Cervantes