ESCUELA TÉCNICA SUPERIOR DE INGENIERÍA Y SISTEMAS DE TELECOMUNICACIÓN
PROYECTO FIN DE GRADO

TÍTULO: Diseño e implementación de estimadores distribuidos para grandes conjuntos de datos.

AUTOR: Javier Martín Vegas

TITULACIÓN: Sonido e Imagen

TUTOR: David Luengo García

DEPARTAMENTO: Teoría de la señal y comunicaciones

Miembros del Tribunal Calificador:

PRESIDENTE: Marta Sánchez Agudo

TUTOR: David Luengo García

SECRETARIO: David Osés del Campo

Fecha de lectura: Abril de 2017

Calificación:

VºBº  

El Secretario,
# Contenido

Resumen .............................................................................................................................................................. 7

Abstract ............................................................................................................................................................... 9

Capítulo 1. Introducción .................................................................................................................................... 11

1.1  Planteamiento del problema ............................................................................................................. 11

1.2  Objetivos y Especificaciones ......................................................................................................... 11

1.2.1 Objetivos ........................................................................................................................................... 11

1.2.2 Especificaciones y restricciones de diseño ...................................................................................... 12

1.3  Metodología de Trabajo y Cronograma ......................................................................................... 12

1.3.1 Metodología de trabajo ................................................................................................................... 12

1.3.2 Cronograma ..................................................................................................................................... 12

1.4  Estructura del proyecto ..................................................................................................................... 13

Capítulo 2. Big Data ........................................................................................................................................... 15

2.1  Introducción ...................................................................................................................................... 15

2.2  El valor del Big Data ........................................................................................................................ 16

2.3  El desarrollo del Big Data ................................................................................................................ 18

2.4  Los desafíos del Big Data ................................................................................................................. 19

2.5  Tecnologías relacionadas ................................................................................................................ 20

2.5.1 Big Data y Cloud Computing ............................................................................................................. 20

2.5.2 Big data e Internet of Things ............................................................................................................ 21

2.5.3 Big data y Centros de Datos ............................................................................................................. 22

2.6  Aplicaciones Big Data ........................................................................................................................ 23

2.6.1 Aplicaciones de Big Data en empresas ............................................................................................. 24

2.6.2 Aplicación en IoT del Big Data ........................................................................................................... 24

2.6.3 Aplicación en el comercio electrónico .............................................................................................. 25

2.6.4 Aplicación en las redes sociales ........................................................................................................ 26

2.6.5 Aplicación en medicina y cuidados sanitarios .................................................................................. 27

2.6.6 Inteligencia colectiva ........................................................................................................................ 28

2.7  Discusión ............................................................................................................................................ 29

Capítulo 3. Inferencia Bayesiana ....................................................................................................................... 31

3.1  Introducción ...................................................................................................................................... 31

3.2  Estimación Bayesiana ........................................................................................................................ 31
<table>
<thead>
<tr>
<th>Capítulo</th>
<th>Título</th>
</tr>
</thead>
<tbody>
<tr>
<td>3.2.1</td>
<td>Teorema de Bayes</td>
</tr>
<tr>
<td>3.2.2</td>
<td>Estimadores MMSE y MAP</td>
</tr>
<tr>
<td>3.3</td>
<td>Métodos de Monte Carlo</td>
</tr>
<tr>
<td>3.3.1</td>
<td>Algoritmos de Monte Carlo</td>
</tr>
<tr>
<td>3.3.2</td>
<td>Algoritmo de Aceptación-Rechazo</td>
</tr>
<tr>
<td>3.3.3</td>
<td>Métodos MCMC: Algoritmo de Metropolis-Hastings</td>
</tr>
<tr>
<td>3.4</td>
<td>Métodos de Monte Carlo distribuidos</td>
</tr>
<tr>
<td>3.4.1</td>
<td>Consensus Monte Carlo</td>
</tr>
<tr>
<td>3.4.2</td>
<td>El problema del sesgo</td>
</tr>
<tr>
<td>3.5</td>
<td>Discusión</td>
</tr>
<tr>
<td>4.1</td>
<td>Introducción</td>
</tr>
<tr>
<td>4.2</td>
<td>Generación de los datos</td>
</tr>
<tr>
<td>4.3</td>
<td>Estimación de R</td>
</tr>
<tr>
<td>4.3.1</td>
<td>Estimadores Heurísticos</td>
</tr>
<tr>
<td>4.3.2</td>
<td>Estimador MMSE</td>
</tr>
<tr>
<td>4.3.3</td>
<td>Estimador de Monte Carlo</td>
</tr>
<tr>
<td>4.3.4</td>
<td>Algoritmo de Metropolis-Hastings</td>
</tr>
<tr>
<td>4.4</td>
<td>Discusión</td>
</tr>
<tr>
<td>5.1</td>
<td>Introducción teórica</td>
</tr>
<tr>
<td>5.2</td>
<td>Ejecución práctica: Primer Caso</td>
</tr>
<tr>
<td>5.2.1</td>
<td>Estima global</td>
</tr>
<tr>
<td>5.2.2</td>
<td>Estima local</td>
</tr>
<tr>
<td>5.3</td>
<td>Ejecución práctica: Segundo caso</td>
</tr>
<tr>
<td>5.4</td>
<td>Discusión</td>
</tr>
<tr>
<td>6.1</td>
<td>Introducción teórica</td>
</tr>
<tr>
<td>6.2</td>
<td>Modelo Gaussiano Multivariable</td>
</tr>
<tr>
<td>6.2.1</td>
<td>FDP a posteriori de la matriz de covarianza</td>
</tr>
<tr>
<td>6.2.2</td>
<td>FDP a posteriori del vector de medias</td>
</tr>
<tr>
<td>6.2.3</td>
<td>Objetivo</td>
</tr>
<tr>
<td>6.3</td>
<td>Implementación práctica del ejemplo</td>
</tr>
</tbody>
</table>
Resumen

En el presente proyecto, titulado “Diseño e implementación de estimadores distribuidos para grandes conjuntos de datos”, se busca un objetivo claro: el tratamiento de grandes conjuntos de datos, ampliamente conocidos como Big Data. Partiendo de una introducción en la que se explica el estado actual del Big Data, se describirá la base matemática que se necesita para comprender y desarrollar algoritmos que permitan extraer información y valor de los grandes conjuntos de datos.

Este proyecto tiene su punto inicial en la explicación de la aparición del Big Data y su importancia en los diferentes ámbitos de la vida. Es decir, se hace un análisis de la situación del Big Data hoy en día en el que se explica la historia que rodea a este concepto, además de las aplicaciones que surgen a partir de la investigación con grandes conjuntos de datos.

Seguidamente el proyecto toma la dirección hacia su objetivo principal: aplicar herramientas matemáticas y estadísticas para llevar a cabo análisis de Big Data. Para ello se explican algunos conceptos que marcarán los puntos de partida: la estimación Bayesiana, los métodos de Monte Carlo, el algoritmo de aceptación y rechazo o la computación en paralelo.

Una vez explicadas estas herramientas, se pondrán en práctica en los Capítulos 4, 5 y 6. Cada uno de estos capítulos considera problemas distintos, pero el punto de partida en todos los casos consiste en disponer de una serie de información y de parámetros que caracterizan a un gran conjunto de datos. A partir de estos conjuntos de datos se realizarán estimaciones mediante algoritmos implementados por el autor. Para la implementación de los mismos se utiliza el software matemático Matlab. Continuamente se harán referencias al código para que el lector pueda seguir paralelamente el código del algoritmo y la explicación teórica.

Por otra parte, cabe destacar que los conjuntos de datos, a partir de los cuales se llevarán a cabo estimaciones y análisis, son conjuntos de datos sintéticos. Es decir, son generados por el autor y tienen unas características determinadas según el caso práctico que estemos ejecutando. Es destacable que los casos prácticos se corresponden con modelos matemáticos determinados. Para la implementación de los algoritmos de análisis de conjuntos de datos es necesaria una base estadística que permita entender la situación de la que se parte en cada caso práctico. Algunos modelos que son vistos son el modelo binomial y el modelo mutivariable Gaussiano. En cada capítulo práctico se explica la situación de partida y se da una pequeña explicación matemática que permita entender los problemas que se presentan, de modo que se pueda seguir el desarrollo del algoritmo.

En el primer caso práctico se realizará la estimación del valor de una resistencia y se implementará el algoritmo de Metropolis-Hastings, muy utilizado en inferencia Bayesiana. En el segundo se afrontará un caso de distribución binomial, en el que se llevará a cabo la estimación de los parámetros que la caracterizan. Por último, se afrontará un caso multivariable y se explicará un método para la corrección del sesgo en estas estimaciones. Este método es conocido como método Jackknife.

Cabe destacar que los resultados obtenidos en los casos considerados en este proyecto son óptimos. Realizando comparaciones con artículos que se referenciarán a lo largo del texto se puede estar satisfecho acerca del rendimiento de los algoritmos.
Abstract

The present project, entitled “Design and implementation of distributed estimators for Big Data applications” has a clear goal: the introduction to analysis of big data sets commonly known as Big Data. From an introduction in which the currently state of Big Data is explained, the next step would be the explanation of the mathematical tools needed to understand and to implement the algorithms used for extracting value from the data sets.

The starting point is the history of Big Data, i. e., the birth of Big Data, all the concepts that it involves, and the influence it has in all aspects of life. Apart from explaining the history of Big Data, some real applications will be shown in this project.

Straightaway, this work focuses on its main objective: to explain the mathematical and statistical tools needed for analyzing big datasets. For achieving this goal some concepts will be explained: Bayesian estimation, Monte Carlo methods, the accept-reject algorithm and parallel computing.

Once that these concepts have been explained, we will put them into practice in Chapter 3. Each of those chapters considers different problems, but, the starting point is the same: a dataset and the parameters which characterize that dataset. Using this information, we will carry out estimation by using algorithms that will be implemented with a mathematical software called Matlab. References to the code will be made constantly in the text, so that the reader can follow the algorithm’s code and its theoretical explanation.

On the other hand, it should be noted that the datasets used for making estimations will be synthetic data (no real data). That is, they will generated by the author and they will have their special characteristics according to the practical case that we are facing. Each practical example will have its mathematical explanation. For this reason, it is necessary to have a statistical knowledge in order to be able to understand the starting point of each algorithm. Some of the statistical models being used here are the binomial model and the Gaussian multivariate model. In each chapter, the starting point will be explained. Also a mathematical introduction will be attached in order to understand the problem.

In the first case, we will estimate a resistor value and the Metropolis Hastings Algorithm will be detailed. This algorithm is widely used in Bayesian inference. For the second case we will face a binomial issue. We will estimate the parameters that characterize that model. Finally, we will analyse a multivariate case. This model will have a problem associated: Bias. The Jackknife method will be explained and will be used to obtain a solution for this problem.

We have to underline that the results obtained are optimal. Making some comparisons to the papers referenced in this problem, we have to stand out the satisfaction with these algorithms.
Capítulo 1. Introducción

1.1 Planteamiento del problema

El propósito general de este proyecto consiste en, mediante herramientas estadísticas, llevar a cabo estimaciones de parámetros en problemas con grandes cantidades de datos. Para llevar a cabo el procesado de esta gran cantidad de datos se usan algoritmos basados en los métodos de Monte Carlo y las cadenas de Markov (métodos MCMC), que utilizan la metodología de estimación Bayesiana para acercarnos asintóticamente a una estimación óptima a medida que el número de muestras crece.

Con el objetivo de reducir costes de comunicación, de computación, y para trabajar a mayor velocidad, dichos algoritmos se llevan a cabo paralelamente en varios nodos sin que estos se comuniquen entre sí (excepto al principio y al final de la estimación). Para ello, previamente se realiza una partición de los datos y cada porción se asigna separadamente a un nodo. La única comunicación que se llevará a cabo será la conexión con el nodo central, que recopilará la información generada por cada nodo, para completar así la estimación global.

Como se ha comentado previamente, el proyecto se basa en los métodos de Monte Carlo, que a su vez utilizan la estimación Bayesiana a partir de funciones de densidad de probabilidad estándar, como la distribución Gaussiana. La metodología de trabajo estará basada en la simulación del algoritmo de Metropolis–Hastings en Matlab, utilizando datos sintéticos correspondientes a diversos ejemplos de interés práctico.

1.2 Objetivos y Especificaciones

1.2.1 Objetivos

En particular, en el presente proyecto se persiguen los siguientes objetivos específicos:

1. Revisión del estado del arte en métodos de Monte Carlo distribuidos [Lee, 2010], que cumplan con las especificaciones detalladas en la siguiente sección.
2. Desarrollo de modelos estadísticos para diversos problemas de interés práctico de acuerdo con la metodología Bayesiana [Stone, 2013]. En la práctica, esto implica la elección de la función de densidad de probabilidad a priori y la verosimilitud apropiadas, y el cálculo de la correspondiente función de densidad de probabilidad a posteriori.
3. Programación y depuración en Matlab de algoritmos de Monte Carlo que permitan llevar a cabo una estimación estadística para los modelos anteriores.
4. Simulación de los algoritmos y comparación de prestaciones para tres casos de interés práctico.
1.2.2 Especificaciones y restricciones de diseño.

A la hora de abordar el proyecto se parte de las siguientes especificaciones y restricciones de diseño:

1. No disponibilidad de estimadores cerrados en la mayoría de problemas prácticos, por lo que hay que recurrir a técnicas de simulación como los métodos de Monte Carlo.
2. Incorporación de información a priori acerca del problema considerado. Dentro de la filosofía Bayesiana, esto supone la elección de la función de densidad de probabilidad apropiada para cada problema considerado.
3. Caracterización probabilística del problema mediante la correspondiente función de verosimilitud.
4. Posibilidad de implementación en paralelo mediante varios agentes de procesado.
5. La comunicación entre agentes no está permitida para evitar los cuellos de botella asociados a la transmisión de información. Dicha comunicación sólo se realizará entre los agentes de procesado y el nodo central (centro de fusión). En la práctica, esto supone restringir la comunicación a la fase inicial (transmisión de los datos) y final (transmisión de los resultados).

Al tratarse de un tema relativamente novedoso, no existe aún ninguna normativa relacionada. Sin embargo, sí que existen una serie de algoritmos (como el “Consensus Monte Carlo” [Scott, 2013] o el “Embarrassingly Parallel Monte Carlo” [Neiswanger, 2013]) que se están convirtiendo en estándares de facto, y que se considerarán a la hora de abordar los problemas planteados.

1.3 Metodología de Trabajo y Cronograma

1.3.1 Metodología de trabajo.

La metodología de trabajo propuesta es la siguiente:

1. Revisión de la metodología de estimación Bayesiana y del estado del arte en métodos de Monte Carlo distribuidos.
2. Programación y depuración de los esquemas elegidos en Matlab.
3. Simulación de los tres ejemplos seleccionados, obtención de resultados y comparación de prestaciones.

1.3.2 Cronograma.

Dado que el alumno se encuentra trabajando, se ha propuesto el desarrollo del proyecto a lo largo de 12 meses con dedicación a tiempo parcial. El desglose de tareas es el siguiente:

1. Tarea 1: Desarrollo de modelos estadísticos basados en la filosofía Bayesiana (Meses 1 – 3).
2. Tarea 2: Revisión del estado del arte en Métodos de Monte Carlo distribuidos y selección de los algoritmos más apropiados (Meses 2 – 4).
3. Tarea 3: Programación y depuración de los esquemas elegidos (Meses 4 – 10).
4. Tarea 4: Simulación de los ejemplos seleccionados y comparación de prestaciones (Meses 7 – 10).
5. Tarea 5: Redacción de la memoria del proyecto y preparación de la presentación (Meses 10 – 12).

1.4 Estructura del proyecto

En primer lugar, el Capítulo 2 hará una introducción al Big Data hoy en día. Además se explicarán sus orígenes, el valor que se puede extraer de él, los problemas que ocasiona y además se verán una serie de aplicaciones reales de este concepto llamado Big Data.

A continuación, en el Capítulo 3 se explicarán las herramientas matemáticas que se utilizarán en este proyecto para trabajar con grandes conjuntos de datos: destacamos el Teorema de Bayes y los métodos de Monte Carlo.

La parte práctica del proyecto se muestra en los Capítulos 4, 5 y 6. En el Capítulo 4 se expondrá el primer ejemplo práctico. En él se llevará a cabo la estimación del valor de una resistencia utilizando varios estimadores. Además se explicará la implementación del algoritmo Metropolis Hastings. Después continuaremos con el Capítulo 5, en el que se llevarán a cabo estimaciones para un caso binomial. Finalmente, en el Capítulo 6, nos adentraremos en el caso multivariable donde seguiremos realizando estimaciones utilizando algoritmos de Monte Carlo a partir de unos datos que cuentan con varias dimensiones y trataremos de encontrar solución a los problemas derivados de la estimación mediante estos algoritmos (el sesgo).

Por último expondremos las conclusiones obtenidas así como líneas futuras de investigación.
Capítulo 2. Big Data

2.1 Introducción

De acuerdo a la información de la Corporación Internacional de Datos (International Data Corporation, IDC), en 2011 la cantidad de datos que había sido creada o copiada en el mundo ascendía a aproximadamente 1,8 ZB (≈10^{21} B) [Gantz, 2011]. Esta cantidad se ha multiplicado por nueve en los últimos cinco años, y se estima que esta cantidad pueda doblarse cada dos años. Por tanto, estamos abocados a que, cada vez más, los datos ganen importancia en nuestro día a día.

El concepto de Big Data ha aumentado su importancia en los últimos años debido a su éxito en campos como los negocios, las ciencias experimentales o la salud. Debido a que es un concepto relativamente reciente hay muchas confusiones acerca de él. Una de ellas es la propia definición de Big Data. El concepto Big Data hace referencia al análisis de grandes conjuntos de datos, el descubrimiento de los valores ocultos en ellos o la manera de organizar y tratar estos conjuntos de datos.

Comúnmente se dice que el concepto Big Data tiene asociadas cuatro V (uves) [Chen, 2014 (p 173)]: (1) Gran volumen de datos; (2) variedad en el tipo de datos; (3) gran velocidad en la generación de datos y en su actualización; y (4) Big Data crea valor.

Debido al potencial que esconden estos conjuntos de datos, cada vez más empresas, universidades, centros de investigación y gobiernos se interesan en el desarrollo de planes de investigación y análisis de Big Data. Este nombre está dejando de ser conocido únicamente en empresas especializadas, cada vez lo podemos escuchar más en nuestra vida diaria debido al trato creciente que los medios de comunicación hacen de él.

Las compañías asociadas a Internet son las que mayor crecimiento de Big Data experimentan: Google procesa cientos de Petabytes (PB), Facebook genera datos entorno a nuevos usuarios de alrededor de 10 PB por mes, Baidu (compañía china) procesa decenas de PB. Sin embargo, la creciente generación de tal cantidad de datos trae asociada una serie de problemas que necesitan solución:

- Los últimos avances en las tecnologías de la información hacen que la generación de datos sea más sencilla. Por ejemplo, cada minuto se suben 72 horas de video a la plataforma YouTube. Por tanto, uno de los desafíos es lidiar con las diferentes fuentes que generan datos y poder llevar un control de las mismas.
- El rápido crecimiento de la nube y del Internet de las Cosas (Internet of Things, IoT) también contribuyen al aumento de la cantidad de datos. Los sensores que hay por todo el mundo recolectan y transmiten datos que son almacenados y procesados en la nube. Tal cantidad de datos sobrepasarán en un futuro cercano la arquitectura e infraestructura actual de las empresas. El crecimiento continuado de este volumen de datos genera el problema de cómo almacenar y manejar tal cantidad heterogénea de datos.
- Considerando la heterogeneidad, la necesidad de procesado en tiempo real, complejidad y privacidad del Big Data, necesitaremos explotar los diferentes conjuntos de datos para extraer sus propiedades intrínsecas y mejorar la toma de decisiones.
Aparte de estos desafíos, la investigación acerca del Big Data debe centrarse en desarrollar una definición apropiada para el término, en extraer su valor, y en cómo transformar un mero conjunto de datos en Big Data. En el resto del capítulo se profundizará en alguno de estos aspectos, así como en el papel del Big Data en este proyecto.

En la Figura 2.1 podemos ver el estado actual del Big Data. Es decir, prácticamente es un gran desconocido. Lo que sabemos acerca del Big data es la punta del iceberg, el verdadero valor que esconden los datos está aún por descubrir.

![Figura 2.1. Big Data hoy día. [Sastri, 2013](image)](image)

### 2.2 El valor del Big Data

McKinsey & Company es una consultora estratégica global que se centra en resolver problemas concernientes a la administración estratégica. Esta compañía observó cómo el Big Data proporcionaba valor después de investigar en profundidad el sistema sanitario de los Estados Unidos, el sector administrativo público, el pequeño comercio, las ventas globales y la localización de las personas en todo el mundo. A partir de la investigación en estos cinco núcleos que representan la economía global, la compañía McKinsey señaló que el Big Data podría ser de gran utilidad en la economía, y serviría para incrementar la productividad y la competitividad entre empresas y el sector público, y finalmente crear enormes beneficios para el consumidor [Manyika, 2011].

McKinsey resumió los importantes beneficios que el Big Data podría crear. Si éste fuese utilizado de forma creativa y efectiva el sistema sanitario americano podría obtener una ganancia de alrededor de 300 billones de dólares, lo que supondría reducir el gasto en salud en un porcentaje del 8%. Además, el pequeño
comercio, utilizando los recursos del Big Data, podría aumentar su productividad en más del 60%. Por otra parte, la potencia de los datos serviría para aumentar la eficiencia de las operaciones estatales, de manera que las potencias económicas europeas podrían ahorrar cerca de 100 billones de euros (excluyendo el efecto de la reducción de fraudes y errores).

Los siguientes hechos nos pueden dar una idea acerca del valor del Big Data. Durante la pandemia gripal del 2009 Google obtuvo información muy importante y oportuna mediante el análisis de datos. De hecho, esta información fue más importante que la proporcionada por los centros de prevención de enfermedades. En ese año, prácticamente todos los países pidieron a los hospitales que informasen a los centros de prevención de esta nueva epidemia gripal. Sin embargo, los pacientes no van al médico inmediatamente después de ser infectados. Además, la comunicación entre los hospitales y los centros de prevención de enfermedades requería tiempo, así como el procesado de toda esta información por parte de los centros de prevención. Por tanto, cuando la sociedad estuviese al tanto del nuevo tipo de gripe, la enfermedad podría haberse extendido ya a nivel global. Google observó que la naturaleza de las consultas en sus sistemas de búsqueda, durante la expansión de la enfermedad, era notoriamente diferente respecto a la de los tiempos ordinarios. También observó que las consultas estaban fuertemente correladas en tiempo y localización. Google recopiló estas consultas relacionadas con el brote de la gripe y las incorporó a modelos matemáticos específicos para pronosticar la difusión de la enfermedad e incluso descubrir los lugares de mayor contagio. Los resultados fueron publicados en la revista Nature [Ginsberg, 2008].

En 2008, Microsoft compró Farecast, una compañía tecnológica investigadora de los Estados Unidos. Farecast tiene un sistema de predicción enfocado a la compra de billetes de avión que informa de las ofertas o las subidas y bajadas de los precios de billetes de avión. El sistema se incorporó al sistema de búsqueda Bing, perteneciente a Microsoft. En el año 2012 el sistema había permitido ahorrar 50 dólares americanos por pasajero [Chen, 2014]. En la Figura 2.2 se ve un ejemplo de cómo funciona esta herramienta: vemos como se muestran varios destinos y a su vez se muestra la variación de los billetes de avión en función del día de vuelo.

![Figura 2.2. Funcionamiento de la herramienta Farecast. [Hip 2009].](image)
En resumen, hoy en día el análisis de los datos se ha vuelto un factor de producción con una importancia comparable a la de los activos materiales o el capital humano. Paralelamente al desarrollo de las tecnologías multimedia y del concepto *social media* (que engloba todas las plataformas en las cuales los usuarios comparten y publican información), las empresas continuarán recolectando datos, llevando hacia un crecimiento exponencial del volumen de los mismos. Esto supondrá un cambio en los negocios y en la relación de ellos con los clientes.

### 2.3 El desarrollo del Big Data

Hemos visto algunos ejemplos de compañías precursoras en el concepto de Big Data. Ahora nos remontamos al comienzo de este concepto tan importante hoy en día. A finales de los años 70 emergió el concepto anglosajón “*database machine*”. Se trataba de una tecnología utilizada particularmente para almacenar y analizar datos. Con el crecimiento del volumen de datos, la capacidad de almacenamiento y procesamiento de una única máquina no era suficiente. En los años 80 se propuso un sistema con máquinas funcionando en paralelo para satisfacer los requisitos asociados al crecimiento del volumen de datos. Este sistema en paralelo contaba con un cluster de máquinas, cada una con su propio procesador y capacidad de almacenamiento. Teradata fue el primer sistema comercial en alcanzar popularidad [Witt, 1992]. En 1986 ocurrió un hecho que marcó un hito: El sistema Teradata fue el primer sistema de funcionamiento en paralelo con una capacidad de un TeraByte, para ayudar a la compañía Kmart en el control logístico de los datos de su almacén. Hacia finales de 1990, las ventajas de las bases de datos en paralelo estaban ampliamente reconocidas en el sector.

Con el desarrollo de los servicios de Internet, los sistemas de búsqueda tenían que enfrentarse a los desafíos de manejar tal cantidad de datos. Google creó GFS (Google File System) [Ghemawat, 2003] y MapReduce [Dean, 2008], dos sistemas de programación enfocados al tratamiento de datos distribuido en el ámbito de Internet. Además, los datos inherentes a los usuarios, así como la información proveniente de la creciente cantidad de sensores, provocaron la necesidad del cambio en la arquitectura existente enfocada al tratamiento de datos a gran escala. En 2007, Jim Gray, un pionero dentro del software de las bases de datos, consideró que la única manera de hacer frente a este cambio pasaba por una revolución en cuanto a las herramientas utilizadas para tratar, visualizar y analizar las bases de datos masivas. En junio de 2011, la publicación del artículo “*Cómo extraer valor del caos*” por parte de EMC/IDC, que introducía el concepto y potencial asociado al Big Data por primera vez, creó un punto de inflexión en este campo [Gantz, 2011].

En los últimos años, prácticamente todas las grandes compañías (entre las que se incluyen Dell EMC, Oracle, IBM, Microsoft, Google, Amazon o Facebook) se han adentrado en diferentes proyectos de Big Data. Por ejemplo, IBM ha invertido 16 billones de dólares en 30 proyectos relacionados con el Big Data desde el año 2005 [Chen, 2014]. Académicamente son muchos los artículos que han sido publicados en los últimos años, por ejemplo el número especial de 2011 de la revista Science [Science, 2011].

El campo del Big Data también ha sido y continúa siendo de interés para los gobiernos. Por ejemplo, en marzo de 2012 el gobierno de Obama anunció el destino de 200 millones de dólares para el plan de desarrollo e investigación del Big Data el cual supuso el segundo mayor plan científico en la historia de Estados Unidos. En 2012, el ministerio de asuntos interiores de Japón resaltó que el desarrollo del Big Data
debería ser parte de la estrategia científica y tecnológica nacional y que debería centrarse en su aplicación en diferentes tecnologías. En julio de 2012 las Naciones Unidas presentó un informe técnico en el que resumía cómo los gobiernos utilizaban el Big Data para servir y proteger a su gente [ONU, 2012].

2.4 Los desafíos del Big Data

Uno de los grandes desafíos del Big Data es el almacenamiento de los datos y el acceso a los mismos. Los sistemas de tratamiento y análisis tradicionales están basados en sistemas de bases de datos relacionales (RDBMS). Estos sistemas funcionan con datos estructurados, no con datos semi-estructurados o no estructurados. Además, los RDBMS utilizan progresivamente hardware más caro. Por otra parte, estos sistemas son incapaces de enfrentarse al volumen y la heterogeneidad del Big Data. La comunidad investigadora ha propuesto algunas soluciones desde diferentes perspectivas. Una es el control en la nube, que cumple los requisitos en infraestructura del Big Data: eficiencia en costes, elasticidad, etc. Para necesidades de almacenamiento permanente y manejo de grupos de datos desordenados a gran escala, los sistemas de archivos distribuidos y las bases de datos NoSQL son una buena opción, aunque no es sencillo desarrollar aplicaciones basadas en los mismos. Aquí hay una lista con algunos inconvenientes y puntos a tener en cuenta y mejorar [Labrinidis, 2012]:

- **Representación de datos**: Muchos conjuntos de datos poseen altos niveles de heterogeneidad en cuando a la estructura, tipo, semántica, organización o accesibilidad. La representación de los datos persigue hacer que los datos sean más significativos para el análisis computacional y la interpretación del usuario. Por tanto, una mala representación de los mismos podría reducir el valor de los datos y suponer un obstáculo en su análisis. Una representación de datos eficiente debe reflejar la estructura de los datos, la clase, el tipo, y las tecnologías integradas para facilitar las operaciones en diferentes conjuntos de datos.

- **Reducción de redundancia y compresión de datos**: normalmente hay una alta redundancia en los conjuntos de datos. Es importante reducirla para abaratar costes sin que el valor de estos datos se vea afectado. Por ejemplo, los sensores de movimiento, temperatura, presión, etc. generan una alta cantidad de datos redundantes que deben ser filtrados y comprimidos.

- **Control del ciclo de vida de los datos**: la velocidad a la que se generan los datos contrasta con el lento avance en el desarrollo de los sistemas de almacenamiento. El valor de los datos depende de la frescura de los mismos. Es importante decidir qué datos deben almacenarse y cuáles no.

- **Mecanismo analítico**: debido a que el sistema analítico referido al Big Data tiene que procesar grandes masas de datos en un tiempo limitado, el sistema RDBMS no cumple con esos requisitos. Las bases de datos no relacionales se acercan más al cumplimiento de estas necesidades y son cada vez más utilizadas en el análisis de datos. Sin embargo también presentan sus problemas particulares. Por tanto es necesario encontrar el equilibrio entre ambas filosofías.

- **Confidencialidad de los datos**: Muchas compañías no pueden analizar sus propios datos debido a carencias en sus infraestructuras y deben confiar en compañías externas, lo que supone un riesgo en la seguridad de sus datos. Antes de llevar a cabo estas operaciones es necesario que la seguridad esté garantizada.
Control energético: el crecimiento en las cantidades de datos a procesar lleva asociado un creciente consumo energético, el cual debe ser controlado.

Escalabilidad: El sistema analítico debe ser flexible para soportar conjuntos de datos tanto presentes como futuros. Es decir, debe ser escalable, con el fin de poder adaptarse a las complejidades que puedan aparecer en un futuro.

Cooperación: El análisis de datos requiere que diferentes expertos cooperen para obtener el máximo beneficio posible del Big Data. Por ello debe establecerse una red que permita la interactividad de científicos e ingenieros para alcanzar los objetivos del análisis.

2.5 Tecnologías relacionadas

Ahora veremos la relación entre Big Data y algunas tecnologías como computación en la nube, IoT (Internet of Things) y data center.

2.5.1 Big Data y Cloud Computing

Primero hablaremos de la interrelación entre Big Data y la computación en la nube, conocida como cloud computing. La computación en la nube es utilizada por Big Data como un medio de procesar grandes conjuntos de datos distribuidos. El principal objetivo de la computación en la nube es proporcionar un funcionamiento adecuado para las aplicaciones de Big Data. Por esta razón, el desarrollo e investigación en el campo del Big Data acelera el desarrollo de la arquitectura de la nube. En la Figura 2.3 se ilustra el concepto de cloud computing: la interconexión de servicios a través de la nube. La filosofía de almacenamiento distribuido mejora la eficiencia del trato de los datos en las aplicaciones de Big Data, y la capacidad computacional paralela mejora la adquisición y el análisis de los datos.
A pesar de compartir muchas tecnologías, la computación en la nube y Big Data se diferencian en dos aspectos principales. Primero, la nube hace referencia a la arquitectura de almacenamiento de datos, mientras que Big Data se centra en cómo analizar y extraer la información relevante de los datos. En segundo lugar, ambas tecnologías están encaminadas hacia dos clientes finales diferentes. La computación en la nube tiene como cliente final a los responsables de información (CIO, *Chief Information Officers*), cuya función principal es facilitar las tecnologías que permitan que se alcancen los objetivos de la empresa, mientras que en el caso del Big Data estos son los directores ejecutivos (CEO, *Chief Executive Officers*) que se centran en operaciones de negocio.

En definitiva, no sólo Big Data influye en el desarrollo de la nube, sino que ésta por sí misma es una tecnología que ofrece soluciones para las herramientas de Big Data. Ambas tecnologías sirven de estímulo entre ellas para su desarrollo e investigación.

### 2.5.2 Big data e Internet of Things

Ahora veremos la relación entre Big Data e IoT. El concepto llamado Internet de las Cosas hace referencia a la interconexión de objetos, que utilizamos de forma cotidiana, mediante Internet. En la Figura 2.4 se ilustra este concepto. El desarrollo del Internet de las Cosas ha provocado un enorme despliegue de sensores integrados en todo tipo de dispositivos y máquinas: Estos sensores adquieren todo tipo de datos cuyas características principales son su heterogeneidad, su desestructuración, ruido y alta redundancia. Los datos relacionados con IoT tienen tres características principales:

1. Hay una gran cantidad de terminales (sensores) que aportan ingentes cantidades de datos.
2. Los datos suelen estar desestructurados o semi-estructurados.
3. Los datos de IoT únicamente son útiles cuando son analizados.
Cabe destacar que hoy en día los datos correspondientes a IoT no son la parte más amplia del Big Data. Sin embargo, se calcula que hacia el año 2030 la cantidad de sensores alcanzará el número de un trillón, y por entonces IoT será la tecnología que mayor cantidad de datos aportará a Big Data. Por tanto, el desarrollo de IoT y las aplicaciones de Big Data deben ir de la mano.

Figura 2.4. Internet de las Cosas. [Chen, 2014].

2.5.3 Big data y Centros de Datos

Seguidamente hacemos referencia a la relación de Big Data con el centro de datos. El centro de datos no sólo actúa como un lugar de almacenamiento masivo de datos, sino que además supone el núcleo para el desarrollo de las infraestructuras de Big Data. Como tal, las empresas deben centrarse en su desarrollo para así mejorar las arquitecturas de datos presentes en la actualidad. Algunas de las necesidades y funcionalidades asociadas a este centro de datos son:

1. Proporcionar una red con múltiples nodos.
2. Construcción de una red de comunicación interna de alta velocidad.
3. Disipación de energía de forma efectiva.
4. Creación de copias de seguridad de los datos que alberga.

Solo asegurando estas funcionalidades, las operaciones asociadas al Big Data podrán alcanzar la seguridad y fiabilidad necesarias para que se lleven a cabo.

A medida que se desarrollen las aplicaciones y aumente el volumen de datos es necesaria una continua expansión y mejora de los centros de datos. Por ello, es fundamental que los costes asociados a estas mejoras sean minimizados. Por último, los centros de datos no deben funcionar únicamente como soporte
hardware, sino que además deben servir de ayuda a la hora del procesado de datos, contribuyendo a la toma de decisiones en temas de negocio o de desarrollo.

2.6 Aplicaciones Big Data

Antes de centrarnos en aplicaciones más específicas de Big Data haremos un repaso de tres campos que sirven como punto de partida para las diferentes aplicaciones relacionadas con el tratamiento de grandes conjuntos de datos [Chen, 2014]:

- **Aplicaciones comerciales:** Originalmente, los datos asociados a los negocios se caracterizaban por ser unos datos estructurados, que eran recolectados por las compañías y almacenados en bases de datos. El trato que se hacía de ellos era simple, destacándose gráficos, diagramas u hojas de presupuestos. Sin embargo, la aparición de la llamada World Wide Web (WWW) proporcionó una nueva dimensión para las compañías: la interacción directa con el cliente. Algunas de las aplicaciones que comenzaron a desarrollarse a partir de esta nueva filosofía fueron análisis personalizados de la compra de productos, sugerencias en cuanto a compras o análisis económicos adaptados al cliente. Con la invasión de los dispositivos móviles, la información referente a estos campos ha aumentado considerablemente, puesto que las compañías tienen un conocimiento cada vez más exhaustivo de las características de sus clientes.

- **Aplicaciones en red:** Como punto de partida de estas aplicaciones tenemos la World Wide Web y los servicios de correo electrónico. El análisis del contenido de los correos electrónicos, así como de los motores de búsqueda han sido las primeras “víctimas” del Big Data. Hoy en día todas las aplicaciones están basadas en la conexión a la web, sin tener en cuenta el campo a que pertenecen o su objetivo a la hora de diseñarla. Las páginas web están llenas de datos: texto, audio, imágenes, vídeos, contenidos interactivos, etc. Por ejemplo, el análisis de imágenes permite extraer información importante como el reconocimiento de caras. Otro tipo de datos, como vídeos, suponen un pilar fundamental para aplicaciones militares, de negocio o para sistemas de vigilancia. Recientemente, los foros de internet, blogs o distintas redes sociales dan oportunidad a miles de usuarios de compartir información y contenidos entre ellos.

- **Aplicaciones científicas:** En campos como la astrofísica, el estudio de océanos, genómica e investigación medioambiental, la cantidad de sensores y demás instrumentos recopiladores de datos es cada vez mayor. Por ejemplo, La National Science Foundation (NSF) de Estados Unidos puso en marcha el plan BIGDATA para promover proyectos centrados en extraer información de los datos relacionados con los sensores anteriormente comentados [Chen, 2014 (p. 194)]. Un ejemplo de aplicación exitosa en este caso es iPlant [Goff, 2011], cuyas bases de datos ayudan tanto a investigadores como a educadores o estudiantes a enriquecer sus conocimientos en relación con la ciencia de las plantas. En la Figura 2.5 se puede ver una de las pantallas de la aplicación.
2.6.1 Aplicaciones de Big Data en empresas

Hoy en día, los datos vienen de empresas y son principalmente usados por ellas. La aplicación del Big Data en empresas puede mejorar la eficiencia de su producción y competitividad en muchos aspectos. En particular, gracias al Big Data las empresas pueden predecir de manera más precisa el comportamiento del consumidor, y en base a ello encontrar nuevos modelos de negocio y desarrollar nuevas estrategias de marketing. Además, analizando estos datos las empresas pueden optimizar el precio de sus productos. De esta manera se mejora la eficiencia y satisfacción en las operaciones, se evita el exceso en la producción y se reducen los costes de producción. Usando Big Data en la cadena de suministro las empresas consiguen la optimización de su inventario, tanto en aspectos de logística como en la coordinación en el suministro, con el objetivo de reducir el hueco entre suministro y demanda, controlar presupuestos y mejorar su servicio.

2.6.2 Aplicación en IoT del Big Data

IoT no es únicamente una fuente para Big Data, sino que también es uno de los mayores mercados para las aplicaciones de Big Data. Debido a la cantidad de campos que cubre IoT, éste se convierte en uno de os principales entornos para el desarrollo de estas.

Las empresas de logística han sido una de las mayores beneficiadas de estas aplicaciones. Por ejemplo, los camiones de UPS están equipados con sensores, adaptadores inalámbricos y GPS. De esta manera, la oficina
central puede monitorizar la posición de los camiones y prevenir fallos en los motores. Por otra parte, este sistema ayuda a UPS a supervisar a sus empleados y optimizar las rutas de entrega de mercancías. Y a partir de la experiencia pasada de los conductores de UPS se han podido mejorar las rutas. Por ejemplo, en 2011 UPS disminuyó la cantidad de kilómetros recorridos por sus conductores en 48,28 millones de kilómetros [Chen, 2014 (p. 198)].

Otro campo donde se centra el desarrollo de aplicaciones de Big Data es la Ciudad Inteligente (Smart City). Algunos de los resultados de su uso es el ahorro de un millón de dólares en las facturas del agua al poder identificar las fuentes de agua que estaban en funcionamiento y perdiendo agua debido a que estaban deterioradas según el Servicio de Mantenimiento de Parques de Dade County (Miami) [Chen, 2014 (p. 198)]. En la Figura 2.6 se indican algunos de los aspectos que caracterizan a una Ciudad Inteligente.

![Figura 2.6. Características de la Ciudad Inteligente. (Joe, 2016).](image)

2.6.3 Aplicación en el comercio electrónico

La aplicación más clásica del Big Data se da en el comercio electrónico. Decenas de miles de transacciones se llevan a cabo en Taobao (portal web chino de compras por Internet equivalente a eBay en Estados Unidos) y los datos correspondientes a los tiempos de transacción, los precios de los productos y la cantidad de mercancías compradas se registran cada día. Acompañados a estos datos aparecen otros como la edad, género, dirección e incluso hobbies e intereses de los compradores. Toda esta información puede ser consultada en la aplicación Data Cube de Taobao, donde los comerciantes pueden estar al tanto de las transacciones de mercado, las condiciones del mismo y los comportamientos de los clientes para tomar decisiones acerca de la producción o del inventario a partir de estos datos.
2.6.4 Aplicación en las redes sociales

Las redes sociales son estructuras constituidas por personas individuales y conexiones entre ellas. Los datos provenientes de ellas principalmente se deben a mensajes instantáneos, blogs o espacios online compartidos. Algunas aplicaciones clásicas enfocadas en las redes sociales, que principalmente analizan el contenido y la estructura de la información para obtener conclusiones de valor a partir de los datos, son las siguientes [Chen, 2014 (p. 198 – 201)]:

- **Aplicaciones basadas en contenido**: Los contenidos con más presencia en las redes sociales son el lenguaje y el texto. El análisis de los mismos proporciona información acerca de las preferencias de los usuarios, sus emociones e intereses.

- **Aplicaciones basadas en la estructura**: En las redes sociales los usuarios se representan como nodos, y las relaciones, intereses y hobbies forman una estructura de red. Esta estructura posee fuertes relaciones internas debido a la interactividad de los individuos. El análisis de esta estructura ha sido utilizado, por ejemplo, por el departamento de policía de Santa Cruz en Estados Unidos. Mediante el empleo de métodos predictivos y el análisis de las redes sociales, el departamento puede descubrir tendencias y modos criminales e incluso predecir tasas de crimen en determinadas regiones [Chen, 2014].

Por otra parte, el motor de búsqueda Wolfram Alpha se dedicó al estudio del comportamiento de las personas mediante el análisis de datos correspondientes a Facebook [Mayer-Schönberger, 2013]. De este estudio se concluyó que la mayor parte de usuarios de esta red social se enamoran alrededor de los 20, se comprometen a los 27 años de edad y se casan a los 30. Estos datos coincidían con el registro demográfico de los Estados Unidos. Por otra parte, Global Pulse hizo uso de los mensajes públicos pertenecientes a usuarios de Twitter en lengua inglesa, japonesa e indonesa desde julio de 2010 hasta octubre de 2011 con el objetivo de analizar temas relacionados con la alimentación, la gasolina, la vivienda y los créditos del banco [ONU, 2012]. A partir de ahí se buscaba entender los comportamientos y preocupaciones de la sociedad. Un ejemplo del resultado de este estudio fue que los datos acerca del cambio en la inflación del precio de la comida en Indonesia tenían relación con el número de Tweets acerca del precio del arroz. Esta correlación podemos verla en la Figura 2.7.
Hablando de manera general, conocer los comportamientos de los usuarios de las redes sociales puede ayudar en tres aspectos:

1. La detección de situaciones de crisis mediante la detección de un uso anormal de los dispositivos y servicios electrónicos.
2. Proporcionar información precisa para poner en práctica políticas y planes de manera eficiente utilizando los comportamientos, emociones y preferencias de los usuarios a tiempo real.
3. Obtener críticas provenientes de los usuarios acerca de determinadas actividades sociales para poder llevar a cabo las modificaciones pertinentes.

2.6.5 Aplicación en medicina y cuidados sanitarios

Los datos asociados a cuidados sanitarios y a la medicina crecen continuamente y de forma muy rápida. La aplicación del potencial del Big Data influenciará de forma notoria la calidad de los servicios médicos.

Por ejemplo, la compañía de seguros Aetna Life seleccionó 102 personas pertenecientes a un banco de mil pacientes para realizar un experimento con el objetivo de predecir su recuperación con síndrome metabólico. A través de la recopilación de datos de tests de 600.000 laboratorios se llegó a un plan de evaluación de los factores de riesgo que ayudó a los médicos a reducir la obesidad mórbida en un 50% durante los siguientes 10 años [Chen, 2014 (p. 200)].

Otro ejemplo se tiene en el centro médico americano Monte Sinaí, que utiliza las tecnologías de la compañía Ayasdi para analizar las secuencias genéticas del Escherichia Coli pertenecientes a un millón de variaciones de ADN. Las tecnologías pertenecientes a Ayasdi utilizan el análisis de datos de tipo topológico para investigar por qué las cepas bacterianas resisten a los antibióticos [Chen, 2014 (p. 200)].
Por último, HealthVault, que fue creada por Microsoft en 2007, es una plataforma que permite la compartición de historiales médicos online. Los individuos pueden acceder a sus historiales médicos mediante sus dispositivos móviles y estos historiales pueden ser gestionados por el centro médico al que el paciente se encuentra suscrito. En la Figura 2.8 se ve la pantalla de la aplicación donde se controla el ritmo cardiaco de usuario.

![Pantalla de la aplicación HealthVault](image)

**Figura 2.8. Control del ritmo cardiaco por la aplicación HealthVault.** [Ememi, 2011]

### 2.6.6 Inteligencia colectiva

El rápido despliegue de las tecnologías inalámbricas y tecnologías sensoriales ha potenciado la capacidad de los dispositivos móviles en aplicaciones de Big Data. Como resultado de este desarrollo ha aparecido el concepto de *crowdsensing*, consistente en un gran número de usuarios que utilizan sus dispositivos móviles como unidades sensoriales para crear una red de compartición de información [Genti, 2011]. Uno de sus puntos fuertes es que no es necesario que los usuarios tengan conocimientos técnicos de Big Data.

Un ejemplo práctico de esta aplicación es el proyecto Nericell de Microsoft, que permite a los usuarios aportar información acerca del tráfico de las carreteras mediante los dispositivos móviles. Mediante los sensores de un móvil (acelerómetro, Bluetooth, GPS, micrófono…) podemos monitorizar el estado de las carreteras y el tráfico de manera sencilla y en tiempo real.

Otra aplicación desarrollada en la Universidad de California es DietSense, que permite a los usuarios compartir sus hábitos alimenticios [Huang, 2015]. El objetivo de esta aplicación es incentivar los correctos hábitos alimenticios mediante la motivación e información que los distintos usuarios puedan aportar.

Este tipo de aplicaciones resultan atractivas para las compañías porque les aportan datos sin la necesidad de realizar importantes inversiones económicas para conseguirlos.
2.7 Discusión

En este capítulo se ha querido acercar el concepto de Big Data al usuario. Primero se introdujeron las características que lo definen: las cuatro uves (volumen, variedad, velocidad y valor). Seguidamente se comentó el valor que el Big data puede aportar a la sociedad y a los diferentes organismos que la componen. A continuación se revisa brevemente la historia y el surgimiento del Big Data, a la vez que se indican las primeras aplicaciones que hicieron uso del análisis de grandes conjuntos de datos. También se presentan los problemas derivados del Big Data, y que hoy día siguen estando presentes. Por esto mismo se dice que el Big data y la arquitectura que lleva asociada deben seguir un desarrollo paralelo. Por último, se introducen algunas aplicaciones que basan su funcionamiento en el Big Data. Así mismo se explican las tecnologías asociadas a este concepto.

En este proyecto se realizará una primera aproximación al análisis de grandes conjuntos de datos. Para este análisis se contará con conjuntos de datos sintéticos. A partir de ellos se realizarán una serie de estimaciones que nos llevarán a sacar conclusiones acerca de la eficiencia del análisis de Big Data. Se implementarán diferentes estimadores, y diferentes situaciones de uso de los mismos.

El principal desafío del proyecto es el desarrollo de algoritmos eficientes que permitan obtener estimaciones correctas sobre grandes conjuntos de datos. No se introducirán aplicaciones específicas, pero sí se tratará la base de los modelos elegidos y su posible aplicación práctica. Algunas de las herramientas matemáticas que se utilizan a la hora de analizar Big Data se explican en los capítulos posteriores.

Entonces, este Capítulo 2 tiene como objetivo poner en situación al lector y hacerle entender la importancia de los grandes conjuntos de datos y su creciente protagonismo en la vida de las personas. Una vez conocido el origen del Big Data, sus puntos pendientes de mejora, los ámbitos de su utilización y los objetivos del presente proyecto, se llevará a cabo la estimación y obtención de conclusiones a partir de grandes cantidades de datos. Para ello, en el siguiente capítulo se exponen las herramientas estadísticas a utilizar, y cómo estas pueden ayudar a la hora de realizar estimaciones. Posteriormente se expondrán una serie de ejemplos prácticos acerca de la utilización de estas herramientas.
Capítulo 3. Inferencia Bayesiana

3.1 Introducción

En este capítulo se aborda la base teórica de los casos prácticos que se analizarán posteriormente en los Capítulos 4-6. Este trabajo se basa principalmente en poner en práctica conocimientos estadísticos para llevar a cabo una toma de decisiones. Si se parte de un conjunto de datos, la estadística en general, y en particular la estadística Bayesiana, proporciona una herramienta útil para el análisis de datos, así como para llevar a cabo una toma de decisiones (conocida como teoría de decisión) posteriormente.

Se define inferencia como la deducción o conclusión que se obtiene a partir de unos datos. La estadística se basa principalmente en el uso de probabilidades. Se puede entender la probabilidad como una cuantificación de la incertidumbre. Dependiendo del tipo de análisis que se haga de dicha probabilidad se tienen diferentes interpretaciones [Berger, 1985]:

a) **Análisis clásico**: suponiendo un experimento aleatorio, a priori los resultados del mismo son igual de verosímiles. Por tanto, la probabilidad de éxito se mide como el cociente entre casos favorables y casos totales.

b) **Análisis frecuentista**: se lleva a cabo el experimento supuesto un número infinito de veces, idealmente, bajo condiciones similares. La probabilidad hace referencia a la proporción de experimentos en que ocurre el evento de interés.

c) **Análisis subjetivo**: se trata de un juicio personal (aunque fundamentado en la experiencia y conocimientos disponibles) sobre la **probabilidad** de que ocurra un resultado.

La estadística Bayesiana, sobre la cual se va a profundizar, se basa en este último tipo de análisis: el análisis subjetivo.

3.2 Estimación Bayesiana

3.2.1 Teorema de Bayes

Se puede definir el teorema de Bayes como un método para interpretar información teniendo en cuenta un conocimiento previo acerca de esa información. Fue formulado independientemente por Thomas Bayes (c. 1701-1761) y Pierre Simon de Laplace (1749-1827). Desde su formulación hasta la actualidad el teorema de Bayes ha estado sometido tanto a elogios como a críticas. La principal crítica proviene de los defensores del análisis frecuentista, ya que sostienen que basar el análisis en unas creencias subjetivas, de las cuales el resultado del experimento depende de forma decisiva, es algo poco riguroso. Por el contrario, si esta información subjetiva se elige cuidadosamente, la estadística Bayesiana es capaz de proporcionar excelentes resultados en problemas muy complejos.
Una forma general del teorema de Bayes, y que a su vez es fácil de recordar es la siguiente [Stone, 2013]:

\[ \text{posterior} = \frac{\text{verosimilitud} \times \text{priori}}{\text{evidencia}} \] \hspace{1cm} (3.1)

Donde \text{priori} es la información a priori, que indica la información inicial antes de recopilar los datos, \text{verosimilitud} es la llamada función de verosimilitud (que se explicará más adelante) y \text{evidencia} es la verosimilitud marginal, que es la función de probabilidad de los datos.

De una manera más formal, el teorema de Bayes se puede formular como [De la Horra, 2003]:

\[ p(\theta|y) = \frac{p(y|\theta) \times p(\theta)}{p(y)} \propto p(y|\theta) \times p(\theta) \] \hspace{1cm} (3.2)

Donde \( \propto \) indica proporcionalidad.

A continuación se indica la terminología a utilizar durante este y los posteriores capítulos en lo que se refiere al Teorema de Bayes:

- \( y \): Datos.
- \( \theta \): Parámetro a estimar.
- \( p(\theta) \): Probabilidad a priori.
- \( p(y|\theta) \): Verosimilitud.
- \( p(\theta|y) \): Probabilidad a posteriori.
- \( p(y) \): Verosimilitud marginal.

Según avance el capítulo se irá aclarando a qué hace referencia cada una de las anteriores definiciones.

El uso del teorema de Bayes se aplica en muchos campos diferentes como pueden ser la genética, el procesado de imágenes, la ecología, la percepción visual, etc [Chen 2014]. Para comprender cómo funciona el teorema de Bayes, James V Stone (2013) presenta un ejemplo sencillo que se explica a continuación.

Una mañana un hombre se levanta con la cara llena de manchas y decide ir al médico, asustado por esos síntomas. El médico le dice algo aterrador: el 90% de las personas que tienen viruela (enfermedad mortal) tienen manchas en la cara. Sin embargo, pensando tranquilamente en ese dato, el paciente se da cuenta de que lo que realmente quiere saber es cuál es la probabilidad de tener viruela, no la de que un paciente tenga manchas sabiendo que tiene viruela. Ahora cambia la situación, puesto que la probabilidad de tener viruela teniendo esas manchas es del 1,1%. Este último dato aumenta mucho la tranquilidad del paciente.

El médico sabe que hay dos enfermedades que implican esos síntomas con una alta probabilidad: viruela y varicela. La probabilidad de tener manchas en la cara teniendo varicela es del 80%, mientras que la probabilidad de tenerlas padeciendo viruela es del 90%. Ambas informaciones son conocidas (en estadística...
Bayesiana) como verosimilitud. Como se trata de un médico experimentado, sabe de antemano que la varicela es mucho más común que la viruela. Este conocimiento recibe el nombre de **probabilidad a priori** y resulta fundamental para decirle al paciente que la enfermedad que padece es probablemente varicela.

Para ver el ejemplo de forma más clara, a continuación trabajamos con números. Usando datos sobre estudios médicos acerca de la viruela y la varicela se pueden establecer las siguientes probabilidades (funciones de verosimilitud):

- **Probabilidad de tener manchas padeciendo viruela:**
  \[ p(\text{manchas}|\text{viruela}) = 0.9 \]  
  \[ (3.3) \]

- **Probabilidad de tener manchas padeciendo varicela:**
  \[ p(\text{manchas}|\text{varicela}) = 0.8 \]  
  \[ (3.4) \]

Estas ecuaciones no tienen en cuenta las informaciones previas, sino que están únicamente basadas en los síntomas. Si tomamos una decisión en base a las probabilidades anteriores, estaríamos asumiendo que la probabilidad de padecer viruela y varicela es la misma. Las expresiones (3) y (4) se identifican con \( p(y|\theta) \), tomada de la expresión (2). \( \theta \) hace referencia a la enfermedad (viruela o varicela en cada caso) e \( y \) se identifica con los síntomas (manchas en la cara).

Como la varicela es más común que la viruela, deberíamos ponderar las probabilidades de las ecuaciones antes de tomar una decisión. Por ejemplo, supongamos que las estadísticas de la salud pública pueden indicar que la probabilidad de tener viruela es del 0.1 %. Por tanto:

\[ p(\text{viruela}) = 0.001 \]  
\[ (3.5) \]

Esto es la probabilidad a priori de que un ciudadano tenga viruela. Estos datos nos llevan a formular el teorema de Bayes para este caso particular:

\[ p(\text{viruela}|\text{manchas}) = \frac{p(\text{manchas}|\text{viruela}) \times p(\text{viruela})}{p(\text{manchas})} \]  
\[ (3.6) \]

El denominador hace referencia a la probabilidad de que un individuo de la población tenga manchas. Este es un término que normalmente no se tiene en cuenta, ya que no depende del parámetro a estimar (si el paciente tiene viruela o varicela en este caso). Sin embargo, sí es necesario conocerlo si se quiere hallar la probabilidad a posteriori exacta. En este caso supondremos:

\[ p(\text{manchas}) = 0.0081 \]  
\[ (3.7) \]
Si sustituimos los diferentes valores en la ecuación (3.6) obtenemos que la probabilidad de tener viruela si nuestro síntoma son manchas es la siguiente:

\[ p(\text{viruela}|\text{manchas}) = \frac{0.9 \times 0.001}{0.081} = 0.011 \]  

Esta probabilidad es conocida como probabilidad a posteriori, \( p(\theta|y) \). Si se quiere obtener un diagnóstico acerca de los síntomas (manchas en la cara) debemos conocer las probabilidades de padecer las dos enfermedades que estamos considerando, la viruela y la varicela. Si suponemos que la incidencia de la varicela en la población general es del 10%:

\[ p(\text{varicela}) = 0.1 \]  

Por tanto, recurriendo al teorema de Bayes:

\[ p(\text{varicela}|\text{manchas}) = \frac{p(\text{manchas}|\text{varicela}) \times p(\text{varicela})}{p(\text{manchas})} \]  

Y sustituyendo valores (recordando que \( p(\text{manchas})=0.0081 \)):

\[ p(\text{varicela}|\text{manchas}) = \frac{0.8 \times 0.1}{0.081} = 0.988 \]  

Vemos que la probabilidad de que la enfermedad padecida sea varicela es mucho más alta que la probabilidad de tener viruela. Por tanto, el médico puede indicarle al paciente que su enfermedad es varicela con un 98,8 % de probabilidad. Como se puede apreciar, el hecho diferencial que nos hace decantarnos por una enfermedad u otra es la información a priori. La importancia de la información a priori se verá reflejada en algunos ejemplos del presente trabajo.
Recopilando e identificando la terminología, podemos formular el ejemplo de una manera más genérica:

\[
\begin{align*}
  p(y|\theta) &= p(\text{manchas}|\text{enfermedad}) \\
  p(\theta|y) &= p(\text{enfermedad}|\text{manchas}) \\
  p(\theta) &= p(\text{enfermedad})
\end{align*}
\]  \hspace{1cm} (3.12)

Donde enfermedad en este caso puede ser varicela o viruela.

En definitiva, el teorema de Bayes combina la información a priori con los datos (en este caso los síntomas) para obtener la probabilidad de padecer viruela dados esos síntomas.

En la Figura 3.1 vemos cómo se combinan las diferentes informaciones.

\begin{figure}
  \centering
  \includegraphics[width=\textwidth]{bayes_rule_diagram.png}
  \caption{Figura 3.1. Representación del teorema de Bayes. [Stone, 2013].}
\end{figure}

Es importante recalcar que la inferencia Bayesiana no aporta la respuesta correcta, sino que proporciona la probabilidad de que esa respuesta sea correcta dada la información disponible.
3.2.2 Estimadores MMSE y MAP

Un estimador es una función que nos proporciona una estima del parámetro deseado a partir de los datos observados. Empezaremos comentando algunas de las características deseadas de los estimadores [Candy, 2009]:

1. **Sesgo.** Se denomina sesgo (“bias” en inglés) a la diferencia entre la esperanza del estimador \( \hat{\theta} \), y el verdadero valor del parámetro a estimar, \( \theta \). Matemáticamente:

\[
Bias(\hat{\theta}) = E(\hat{\theta}) - \theta
\]

Donde \( E(\hat{\theta}) \) indica la esperanza matemática del estimador, que es una variable aleatoria (al ser \( \hat{\theta} \) función de los datos, que son variables aleatorias).

A la hora de estimar un parámetro, podemos diferenciar entre estimadores sesgados o estimadores insesgados.

- Estimador sesgado: \( E(\hat{\theta}) \neq \theta \)
- Estimador insesgado: \( E(\hat{\theta}) = \theta \)
- Estimador asintóticamente insesgado: \( \lim_{n \to \infty} E(\hat{\theta}) = \theta \) donde \( n \) es el número de datos disponibles.

2. **Eficiencia.** Se dice que un estimador es más preciso que otro si la varianza del primero es siempre menor que la del segundo. Esto es, dados dos estimadores \( \hat{\theta}_1 \) y \( \hat{\theta}_2 \), se dice que \( \hat{\theta}_1 \) es más eficiente que \( \hat{\theta}_2 \) si \( Var(\hat{\theta}_1) < Var(\hat{\theta}_2) \) para un mismo conjunto de datos, donde \( Var \) hace referencia a la varianza, que matemáticamente se define de la siguiente forma:

\[
Var(\hat{\theta}) = E \left[ (\hat{\theta} - E(\hat{\theta}))^2 \right] = E(\hat{\theta}^2) - (E(\hat{\theta}))^2
\]

Una vez conocidas las dos características deseadas de los estimadores, se define el error cuadrático medio o MSE (Mean Squared Error), que mide el promedio de los errores al cuadrado, es decir, la esperanza de la diferencia entre el valor del estimador y el verdadero valor del parámetro a estimar elevada al cuadrado:

\[
MSE(\hat{\theta}) = E \left[ (\hat{\theta} - \theta)^2 \right]
\]

Desarrollando el MSE es posible demostrar que:

\[
MSE(\hat{\theta}) = Var(\hat{\theta}) + (Bias(\hat{\theta}))^2
\]
Una vez desarrolladas estas relaciones, nos centramos en los estimadores $\text{MMSE}$ y $\text{MAP}$, que son los que se utilizan principalmente en la práctica y serán los únicos considerados a lo largo del proyecto.

El estimador $\text{MMSE}$ (Minimum Mean Squared Error) es aquel que minimiza el $\text{MSE}$. Es decir:

$$\hat{\theta}_{\text{MMSE}} = \arg\min_{\theta} \text{MSE}(\theta)$$

Es bien conocido que este estimador viene dado por la media condicional [Candy, 2009]:

$$\hat{\theta}_{\text{MMSE}} = E(\theta | y) = \int \theta P(\theta | y) d\theta$$

El estimador $\text{MAP}$ (Maximum a Posteriori) es el que maximiza la probabilidad a posteriori, y viene dado por:

$$\hat{\theta}_{\text{MAP}} = \arg\max_{\theta} P(\theta | y)$$

3.3 Métodos de Monte Carlo

3.3.1 Algoritmos de Monte Carlo

Los inicios de los algoritmos se remontan a la aparición del ENIAC (Electronic Numerical Integrator and Computer), que fue el primer ordenador de propósito general. Era una máquina digital capaz de resolver un gran número de problemas numéricos mediante la reprogramación. El matemático Stanislaw Ulam consideró que con el uso de este ordenador las herramientas estadísticas (que habían caído en desuso por el inconveniente de los tediosos y largos cálculos que las acompañaban) podrían renacer. Contempló esta idea con el catedrático de matemáticas del Institute for Advanced Study, John von Neumann. Esta situación desencadenó el surgimiento de los Métodos de Monte Carlo [Metropolis, 1949].

El método de Monte Carlo nace del interés de Stan Ulam por los procesos aleatorios. Stan Ulam fue el creador del concepto “lucky numbers” y era un investigador de los procesos ramificados, que puso en práctica durante la guerra para el estudio de la multiplicación de neutrones en los dispositivos de fisión. El nombre dado al algoritmo (Monte Carlo) proviene del hecho de que un tío de Stan Ulam pedía prestado dinero a sus parientes para poder continuar jugando en el casino de Monte Carlo en Mónaco.
El método de Monte Carlo da lugar a algoritmos no deterministas, es decir, no se puede saber de antemano cuál será el resultado de la ejecución del mismo. Este método es usado para aproximar expresiones matemáticas complejas y costosas de evaluar con exactitud. Como se ha comentado anteriormente, el padre de este método de Monte Carlo es Stan Ulam. Stan Ulam, mientras se recuperaba de una enfermedad en 1946, decidió averiguar cuál era la probabilidad de ganar al solitario. Para ello advirtió que resultaba mucho más simple extraer una conclusión después de jugar múltiples partidas y contar el número de victorias que obtenerla a partir de realizar todas las posibilidades de combinación de cartas de manera formal. Stan Ulam mencionó el método a John von Neumann que, entusiasmado con la idea, decidió aplicarla a su teoría de generación de neutrones.

El algoritmo de Monte Carlo viene descrito en la carta enviada por el catedrático John von Neumann a Robert Richtmyer (líder de la Theroretical Division del Laboratorio Nacional de Los Álamos de Estados Unidos) [Beich, 2000]. Consideremos un núcleo esférico de material fisionable (se define la fisión como la rotura del núcleo de un átomo pesado, generalmente de uranio, mediante el bombardeo con neutrones, produciendo una liberación de energía) rodeado por una carcasa de material anti-forzado. Asumimos una distribución inicial de neutrones en el espacio y su velocidad, ignorando los efectos radiactivos e hidrodinámicos. La idea es seguir el desarrollo de las cadenas de reacción de neutrones individuales como resultado de procesos de dispersión, absorción, fisión y escape. En cada paso se toman una serie de decisiones basadas en probabilidad estadística en torno a factores geométricos y físicos. Se comienza en el instante \( t=0 \), cuando se toma una cierta velocidad y un cierto posicionamiento en el espacio para un neutrón. Como punto de partida, las siguientes decisiones son la posición de la primera colisión y la naturaleza de la misma. Si ocurre una fisión se debe decidir el número de neutrones que emergen y para cada neutrón se seguirá el mismo proceso llevado a cabo desde el principio. Si la colisión da lugar a una dispersión se utilizan las aproximaciones estadísticas adecuadas para determinar el nuevo momento del neutrón. Cuando un neutrón cruza la frontera de un material se tienen en cuenta las características y parámetros del nuevo material. Por tanto se establece un “árbol genealógico” de cada neutrón. Este proceso se repite para cada neutrón hasta que podamos observar una distribución estadística válida.

El uso del ENIAC para llevar a cabo la aplicación de los algoritmos de Monte Carlo en estos experimentos resultó un relativo éxito y supuso la consolidación del método de Monte Carlo en el ámbito de la física. Dicho éxito fue afianzado por siguientes experimentos y pruebas. Paralelamente aparece otro nombre destacado en la historia del algoritmo de Monte Carlo. Este hombre es el italiano Enrico Fermi, quien ideó un método basado en un sistema de dos dimensiones que dibujaba los comportamientos de la población de neutrones en diferentes tipos de experimentos nucleares.

Una vez introducido el contexto histórico bajo el que se desarrollaron los Métodos de Monte Carlo, podemos definir a los mismos como una serie de algoritmos computacionales que basan su funcionamiento en la simulación estadística con el objetivo de obtener unos determinados resultados numéricos. El uso de los Métodos de Monte Carlo para resolver problemas de estimación Bayesiana consiste en sustituir complejas distribuciones estadísticas, asociadas a un problema considerado irresoluble, mediante la generación de una serie de muestras a partir de una distribución conocida. Por tanto, llanamente podríamos decir que los Métodos de Carlo consisten “simplemente” en generar un gran número de muestras a partir de una determinada distribución, para seguidamente realizar una estimación utilizando esas muestras (por ejemplo, mediante un promediado). Por ejemplo, utilizando el método de Monte Carlo, el estimador MMSE de la ecuación (3.18) se puede aproximar como:
\[ \hat{\theta}_{MSE} = \frac{1}{M} \sum_{m=1}^{M} \theta^{(m)} \]

Donde \( \theta^{(m)} \) (para \( m = 1, ..., M \)) son muestras independientes e idénticamente distribuidas de acuerdo con \( P(\theta|y) \).

El problema del método de Monte Carlo es que no seamos capaces de generar muestras a partir de una distribución deseada, bien porque esta distribución no tenga una forma analítica conocida o bien porque sea muy difícil generar muestras de la misma. Es en este momento cuando aparecen en juego los métodos MCMC (Markov Chain Monte Carlo), como el Algoritmo de Metropolis-Hastings.

### 3.3.2 Algoritmo de Aceptación-Rechazo.

Para adentrarnos en el algoritmo de Metropolis-Hastings, primero es necesario conocer la base del mismo: el algoritmo de aceptación y rechazo.

El algoritmo de aceptación y rechazo fue introducido por John von Neumann para la simulación del transporte de los neutrones. Básicamente, si queremos obtener muestras de alguna distribución de probabilidad específica, simplemente lo que debemos hacer es obtener muestras de una distribución sencilla y conocida y quedarnos solo con las “muestras buenas”.

Von Neumann describió el método usando una distribución uniforme (esto es, una distribución que asigna la misma probabilidad a todos los eventos, pero el método se puede generalizar fácilmente para usar cualquier otra distribución.

Supongamos que queremos obtener muestras de una población acorde a una distribución de probabilidad determinada, \( P(\theta|y) \), que en teoría es conocida pero de la cual es difícil obtener muestras en la práctica. Sin embargo, podemos obtener muestras fácilmente de una distribución \( q \) relacionada, tal que \( c_\nu(\theta) \geq q(\theta) \) para algún \( c > 0 \).

El algoritmo de aceptación y rechazo sigue los siguientes pasos:

1. Obtenemos una muestra \( \theta' \) de la distribución \( q \).
2. Evaluamos \( P(\theta'|x) \).
3. Generamos un valor aleatorio \( \rho \) de la distribución uniforme en el intervalo \([0,1)\):
   - Si \( \rho < cP(\theta'|x)/q(\theta') \), entonces aceptamos \( \theta' \).
   - Si no, descartamos \( \theta' \).

**Algoritmo 1. Método de aceptación-rechazo.**

Este proceso se repite hasta que se hayan generado tantas muestras como sea necesario.
3.3.3 Métodos MCMC: Algoritmo de Metropolis-Hastings.

A continuación describimos otro de los pilares del funcionamiento del algoritmo de Metropolis-Hastings: los métodos de Monte Carlo basados en cadenas de Markov (MCMC). Esta técnica consiste en generar muestras de una distribución de interés utilizando una cadena de Markov cuya función de densidad de probabilidad (FDP) estacionaria es la distribución deseada.

Una cadena de Markov es un proceso discreto aleatorio que tiene la propiedad de que la muestra presente generada, dadas todas las muestras pasadas, depende únicamente de la muestra previa [Walpole, 1999]:

\[
\Pr(\theta_t|\theta_{t-1}, \ldots, \theta_0) = \Pr(\theta_t|\theta_{t-1})
\] (3.19)

Una cadena de Markov viene definida por su estado inicial, \( \theta_0 \), y por el llamado núcleo de transición ("transition kernel"): \( \Pr(\theta_t|\theta_{t-1}) \). Bajo ciertas condiciones poco restrictivas [Robert, 2004], una cadena de Markov converge a su distribución estacionaria:

\[
\lim_{t \to \infty} \Pr(\theta_t) \to \Pr(\theta)
\] (3.20)

Por tanto, si queremos obtener valores de una FDP a posteriori, como por ejemplo \( P(\theta|x) \), podemos utilizar una cadena de Markov \( (\theta_1, \theta_2, \ldots) \) cuya distribución estacionaria, de acuerdo con (20), sea la FDP a posteriori bajo estudio, \( P(\theta|x) \).

Si implementamos correctamente el algoritmo, podemos asegurar la convergencia de la cadena hacia la distribución buscada independientemente de los valores iniciales elegidos. Para que esta convergencia se produzca, es necesario que se lleven a cabo un elevado número de iteraciones. Las primeras iteraciones, llamadas iteraciones de “burn-in”, serán eliminadas al no haber alcanzado aún la cadena el estado estacionario.

Una vez conocido el funcionamiento de una cadena de Markov, así como el algoritmo de aceptación y rechazo, podemos describir los pasos del algoritmo de Metropolis-Hastings.

Si \( \theta \) se trata del parámetro que queremos estimar, primero debemos establecer un estado inicial, que designaremos como \( \theta_0 \), así como la distribución que usaremos para generar muestras candidatas, que llamaremos \( q(\theta) \). Como se ha comentado previamente, es conveniente que esta distribución sea sencilla, a pesar de que sus características se alejen de la distribución que en realidad sigue el parámetro.

Los pasos 1, 2 y 3 los llevaremos a cabo un elevado número de veces (T iteraciones), en parte por las razones de convergencia de la cadena de Markov indicadas previamente y en parte porque el rendimiento del algoritmo mejora con el número de muestras generadas.
A continuación se describe el algoritmo, donde $P(\theta | y)$ es la probabilidad a posteriori obtenida con el teorema de Bayes (2). Si $u' \leq \alpha$ entonces aceptamos la muestra $\theta'$. Sin embargo, si $u' > \alpha$ rechazamos la muestra. En ambos casos repetimos los pasos 1-3 hasta obtener el número de muestras deseado.

1. Generamos una muestra $\theta'$ a partir de la distribución $q(\theta)$.
2. Generamos un valor aleatorio $u'$ a partir de la distribución uniforme $U(0,1)$. Este valor será necesario para llevar a cabo el algoritmo de aceptación y rechazo.
3. Aceptar o rechazar $\theta'$ con probabilidad $\alpha$.

$$
\alpha = \min \left( 1, \frac{q(\theta_{t-1}) \times P(\theta'|y)}{q(\theta') \times P(\theta_{t-1}|y)} \right) \quad (3.21)
$$

Si $\theta'$ se acepta, $\theta_t = \theta'$. En caso contrario, $\theta_t = \theta_{t-1}$.

**Algoritmo 2 Algoritmo de Metropolis-Hastings.**

Siguiendo este proceso un número alto de iteraciones, teniendo en cuenta que a más iteraciones mejor funcionará el algoritmo pero mayor coste computacional supondrá, podemos obtener una buena estima del parámetro deseado.

Por tanto, siendo $\theta_t$ las muestras de la cadena de Markov que hemos guardado, podemos obtener una aproximación del estimador MMSE:

$$
\hat{\theta}_\text{MMSE} = \frac{1}{T - T_b} \sum_{t=T_b+1}^{T} \theta_t \quad (3.22)
$$

Donde $T_b$ es el periodo de “burn-in”. Esto es, el número de muestras descartadas para asegurar la convergencia de la cadena de Markov.

Y de manera similar,

$$
\hat{\theta}_\text{MAP} = \arg\max_{\theta_t} P(\theta_t | y) \quad (3.23)
$$

En el capítulo posterior veremos la aplicación de este algoritmo y comprobaremos su funcionamiento.
3.4 Métodos de Monte Carlo distribuidos

El uso de la inferencia Bayesiana para el análisis de modelos estadísticos complejos ha aumentado notablemente estos últimos años, en parte gracias al poder actual de cálculo computacional disponible. Hay muchos tipos de algoritmos y técnicas disponibles para llevar a cabo problemas de inferencia Bayesiana. Sin embargo, la mayor parte conllevan una demanda muy alta de recursos computacionales. El único modo práctico de llevar a cabo estos análisis estadísticos es usando las técnicas MCMC (“Markov chain Monte Carlo”), pero su coste computacional implica la necesidad de introducir la computación en paralelo.

Existen dos estrategias diferentes utilizadas para paralelizar las técnicas MCMC [Wilkinson, 2006]:

- Múltiples cadenas MCMC en paralelo.
- Paralelización de una única cadena MCMC.

La paralelización de los cálculos añade una capa más de complejidad a los algoritmos MCMC. Uno de los mayores problemas de estos sistemas es el coste que supone la comunicación entre las máquinas: cuantas más máquinas participan en el cálculo mayor es dicho coste.

Hay dos filosofías principales a la hora de dividir los datos: la primera es dividir el trabajo en varios núcleos de un mismo chip, como puede ser una unidad de procesamiento multi-core (CPU) o una unidad de procesamiento gráfico en paralelo (GPU). El procesamiento en varios núcleos puede ser muy efectivo, pero aparecen dos limitaciones. La primera es que los cuellos de botella ligados al espacio en memoria no desaparecen. El segundo problema es la programación: los códigos con varios hilos son difíciles de desarrollar, ya que pueden aparecer problemas asociados durante la ejecución de los mismos que son difíciles de detectar. Además, el trabajo con GPUs requiere un manejo de memoria muy pequeño, lo cual es difícil de abstraer para trabajar con lenguajes de programación de alto nivel.

La alternativa a trabajar con unidades multi-core es el uso de varias máquinas. En este caso los datos se reparten entre las diferentes máquinas, como indicábamos antes. Un aspecto favorable de este modo de trabajo es la desaparición de los cuellos de botella anteriores, pero el coste de comunicación entre las máquinas es muy alto. Cabe destacar que ambos modos de trabajo son complementarios. Por ejemplo, utilizar máquinas multi-core en un sistema con varias máquinas aceleraría el procesado en cada máquina.

La principal diferencia entre ambas filosofías es la comunicación. En los sistemas multi-core la velocidad de comunicación es apreciablemente mayor que en los sistemas con varias máquinas, independientemente del tamaño del mensaje que se transmite. Por lo tanto, para que un algoritmo de Monte Carlo sea efectivo en un entorno con varias máquinas es necesario evitar la comunicación entre las mismas en la medida de lo posible. Para minimizar la comunicación, cada máquina genera muestras a partir de la información local de la que dispone y las estimaciones de cada máquina son combinadas después. Este método de trabajo es conocido como Embarrassingly parallel [Wilkinson, 1999] y uno de los principales algoritmos de este tipo es el “Consensus Monte Carlo” [Scott, 2013], que es el que se ha usado en el PFG y se describe a continuación.
3.4.1 Consensus Monte Carlo

La idea fundamental sobre la que se apoya el Consensus Monte Carlo es dividir los datos en grupos (llamados generalmente shards), proporcionar cada shard a un worker (que llevará a cabo una estimación de Monte Carlo local) y finalmente combinar las estimaciones de cada worker para dar lugar a una estimación global.

Si $x$ representa la totalidad de los datos, $x_s$ hace referencia a los datos de cada shard ($s$). Siendo $\theta$ el parámetro del modelo que deseamos estimar y, asumiendo que los datos son independientes:

$$p(\theta \mid y) = \prod_{s=1}^{S} p(\theta \mid y_s) \propto \prod_{s=1}^{S} p(y_s \mid \theta) p(\theta)^{1/S} \quad (3.24)$$

Vemos que la distribución a priori, $p(\theta) = \prod_{s} p(\theta)^{1/S}$, se divide en $S$ componentes (una componente por cada uno de los $S$ shards). En los siguientes capítulos veremos cómo esta información que se aporta a cada worker influye en el resultado final.

Si cada worker devolviese su distribución a posteriori como una función matemática, simplemente habría que multiplicar las mismas para obtener el resultado final, ya que asumimos que cada worker trabaja de manera independiente. El problema es que estos workers devuelven el resultado como un conjunto de muestras de Monte Carlo, lo cual complica la manera de combinar estas muestras para obtener el resultado final.

La manera más simple de combinar estos resultados es simplemente promediarlos, pero esta no es una solución óptima. Una alternativa mejor consiste en asignar a cada worker una matriz de pesos, $W_s$. Por tanto, si para cada uno de los workers tenemos $G$ estimaciones de Monte Carlo, la combinación entre workers utilizando una matriz de pesos se hará de la siguiente manera [Scott, 2013]:

$$\theta_g = \left( \sum_s W_s \right)^{-1} \sum_s W_s \theta_{sg} \quad para \ g = 1, \ldots, G \quad (3.25)$$

De este modo obtenemos $G$ muestras distribuidas aproximadamente de acuerdo con $p(\theta \mid y)$ combinando las $G$ muestras obtenidas a partir de $p(\theta \mid y_s)$ de acuerdo con su “calidad”.

A continuación se proporciona una síntesis de los términos utilizados en este algoritmo:

- Promedio: $\theta_{g} = \frac{1}{S} \sum_{s=1}^{S} \theta_{sg} \ buscado para \ g = 1, \ldots, G$
- $\theta_{sg}$: Muestra $g$-ésima del “shard” $s \ (s = 1, \ldots, S)$, distribuido de acuerdo a $p(\theta \mid y_s)$
- $\theta_{g}$: Muestra g-ésima de la posterior global.
- $W_s = Var(\theta \mid y_s)$: Peso óptimo para el Caso Gaussiano.
Cuando cada $p(\theta|y_s)$ es una distribución Gaussiana, la distribución a posteriori conjunta, $p(\theta|y)$, también es Gaussiana y la ecuación (3.25) da un resultado idéntico al que obtendríamos si estimásemos $\theta$ de manera centralizada a partir de la distribución total, $p(\theta|y)$.

En resumen, el algoritmo de Consensus Monte Carlo sigue los siguientes pasos:

1. Dividir $y$ en shards $y_1, \ldots, y_S$
2. Realizar $S$ algoritmos de Monte Carlo en paralelo obteniendo $\theta_{sg} \sim p(\theta|y_s)$ para $g = 1 \ldots G$ en cada shard usando la información a priori fraccionada $p(\theta)^\frac{1}{S}$.
3. Combinar las muestras de los shards usando promediado con pesos como en la ecuación (3.25).

Aunque el algoritmo es óptimo para un modelo Gaussiano, veremos en ejemplos de capítulos posteriores que también funciona bien para otro tipo de distribuciones. Esto es porque las distribuciones a posteriori tienden a una forma Gaussiana cuando el conjunto de muestras tiende a infinito debido al Teorema Central del Límite [Walpole, 1999].

3.4.2 El problema del sesgo

Cuando tenemos una gran cantidad de datos con la que trabajar, al dividir estos en shards y combinarlos posteriormente puede aparecer un desplazamiento de la distribución a posteriori del parámetro, tal como se muestra en la Figura 3.2.

Figura 1.2. Ejemplo de sesgo. En negro la estimación global y en línea roja discontinua la estimación local.
Vemos como la estimación procedente del consensus Monte Carlo (línea de puntos) aparece desplazada hacia la derecha. La línea más fina que aparece por debajo de la distribución a posteriori (línea continua) es la que se obtiene de resultado al realizar la corrección Jackknife [Miller 1974]. Esta corrección se utiliza para mitigar el sesgo cuando este es un problema. En el Capítulo 6 veremos cómo se implementa.

### 3.5 Discusión

En este capítulo se ha introducido el concepto de estadística Bayesiana. Se ha visto como esta es una herramienta importante para la teoría de decisión e inferencia estadística. Mediante un ejemplo sencillo hemos explicado cómo el teorema de Bayes es utilizado para llevar a cabo una predicción. Partiendo del concepto de probabilidad a priori, y la verosimilitud, nos permite hallar la probabilidad a posteriori que se utiliza para llevar a cabo la estimación de un parámetro de interés dada una serie de datos.

Para llevar a cabo estas estimaciones se presentaron los estimadores MMSE y MAP, que minimizan el error cuadrático medio y maximizan la probabilidad a posteriori respectivamente. Debido a la dificultad de encontrar expresiones cerradas para estos estimadores se desarrollan los métodos de Monte Carlo.

Se ha realizado una introducción histórica del método de Monte Carlo y seguidamente la definición del mismo. Se explicó que, debido a la dificultad de generar muestras a partir de una distribución, se debía recurrir a los métodos MCMC, entre los cuales el más utilizado es el algoritmo de Metropolis-Hastings.

El objetivo de este capítulo es la explicación de estas herramientas. Es decir, se quiere hacer saber que para llevar a cabo una toma de decisión acerca de un modelo matemático no es necesario realizar su completo y complejo análisis, sino que mediante el uso de los modelos explicados, se pueden abordar problemas matemáticos complejos desde un punto de vista más sencillo y cómodo.

Por último, al llevar estos algoritmos a la práctica se encuentra la necesidad de agilizar los costes computacionales. Para ello se explica el método de Consensus Monte Carlo, que permite dividir el trabajo de estimación en diferentes grupos de trabajo sin que ello suponga una pérdida en la precisión de la estimación en muchos casos de interés.

En capítulos posteriores se lleva a cabo una demostración de las herramientas aquí explicadas en tres casos de interés práctico, así como la solución de los diferentes problemas que las mismas presentan.
4.1 Introducción

En este primer ejemplo realizaremos una introducción al tratamiento de datos y presentaremos un problema práctico de cómo trabajar con las herramientas que se describieron de forma teórica en el Capítulo 3. Se llevará a cabo una primera aproximación a los métodos de Monte Carlo y al algoritmo de Metropolis-Hastings. Para ello haremos una analogía sencilla referida al proceso de medición de una resistencia.

Supóngase el siguiente ejemplo. Una empresa fabricante de resistencias lanza una tirada de resistencias cuyo valor nominal es $R_0 = 100 \, \Omega$. El proceso de fabricación no es exacto, es decir, al valor nominal de la resistencia el fabricante añade una tolerancia del 10%. Por tanto, $(1 - \delta)R_0 \leq R \leq (1 + \delta)R_0$ siendo $R$ el valor real de la resistencia y $\delta = 0,1$ en nuestro caso. El fabricante aporta como dato que el valor de las resistencias (el parámetro $R$) sigue una distribución uniforme centrada en el valor nominal y con rango desde $(1 - \delta)R_0$ hasta $(1 + \delta)R_0$.

Si quisiéramos obtener el valor real de la resistencia, llevaríamos a cabo una medición. El proceso de medida es inexacto, ya que depende de factores ambientales y del ruido en el sistema de medida. Por ejemplo, uno de los factores ambientales que influye en el valor de la resistencia eléctrica es la temperatura, tal y como se muestra en la Figura 4.1.

![Figura 4.2. Influencia de la temperatura en la resistencia eléctrica del cobre. [UPV]](image-url)

Figura 4.2. Influencia de la temperatura en la resistencia eléctrica del cobre. [UPV]
Para obtener un valor preciso que indique el valor que tienen las resistencias que nos da ese fabricante lo más intuitivo, en un principio, sería medir cada resistencia un gran número de veces y realizar un promediado de todas las medidas que se han realizado. Con el objetivo de evitar este laborioso trabajo utilizaremos datos sintéticos que se basarán en un modelo matemático determinado.

### 4.2 Generación de los datos

Como hemos indicado anteriormente, las medidas que obtendríamos en el proceso de medida serían inexactas. Debemos simular este proceso de medida inexacto. Una forma de simular este proceso es añadir un ruido al valor nominal indicado por el fabricante. La distribución del ruido depende de muchos factores, y por tanto es difícil conocer la misma. Sin embargo, el teorema central del límite nos dice que, cuando el ruido es la suma de múltiples factores independientes, su distribución tiende a ser Gaussiana [De La Horra, 2003]. Consecuentemente, en este ejemplo supondremos que el ruido sigue una distribución normal con varianza igual a $100 \, \Omega^2$, lo que supone una desviación típica de 10 $\Omega$ con respecto al valor medio. Así mismo tomaremos como centro de esta distribución normal (esto es, como su media) el valor de 0 $\Omega$.

Escribiendo estos datos en notación matemática:

\[
\sigma_w^2 = 100 \, \Omega^2 \quad (4.1)
\]
\[
\sigma_w = 10 \, \Omega \quad (4.2)
\]

Por tanto, considerando que el valor real de la resistencia es $R$, y que el ruido $w$ simula el proceso inexacto de medida, podemos escribir los valores medidos de esta forma. En la siguiente expresión, $Y[n]$ hace referencia a los valores que hemos obtenido tras la medición con un multímetro (continuando con la analogía indicada anteriormente).

\[
Y[n] = R + w[n] \quad n = 1, \ldots, N \quad (4.3)
\]

siendo $w \sim N(0, \sigma_w^2)$. Donde $N$ indica el número de veces que medimos esta resistencia y $w[n] \sim N(0, \sigma_w^2)$ indica que $w[n]$ sigue una distribución Gaussiana con media nula y varianza $\sigma_w^2$. Intuitivamente, cuantas más veces se realice esta medida, mejor podremos conocer el valor real de la resistencia. Por ejemplo, si promediámos todas las medidas obtenidas, cuanto mayor sea la población a promediar mayor será el nivel de confianza de dicho promedio, gracias a la ley de los grandes números [Walpole, 1999].
4.3 Estimación de R

Como se ha dicho antes, el objetivo es estimar el valor del parámetro R. Hemos comenzado generando una serie de datos $Y[n]$ que se identifican con las medidas hechas por un multímetro sobre una resistencia.

Veamos ahora cómo podemos simular este proceso con la que será nuestra herramienta de trabajo: Matlab. Matlab es una herramienta de software matemático que permite la manipulación de datos, así como la representación de funciones, entre un conjunto muy extenso de funcionalidades.

Las siguientes líneas de código pertenecen al script utilizado para la generación de los datos $Y[n]$:

```matlab
function [Y,R] = GenDatos(R0,delta,N,sigma2_w)

%Suponemos que el valor a estimar R (el valor de la resistencia) toma una distribución uniforme
%R0: valor nominal de la resistencia.
%N: número de datos a generar.
%delta: tolerancia del fabricante.
%sigma2_w: varianza del ruido.

R = rand(1)*((R0+delta)-(R0-delta))*R0 + R0*(1-delta);
w = randn(1,N);
Y = R + sqrt(sigma2_w)*w;
```

$R$ hace referencia al valor de la resistencia que nos proporciona el fabricante. Este se encuentra en un intervalo centrado en el valor nominal $R_0$ y delimitado por la tolerancia facilitada también por el fabricante. Nótese que, de acuerdo con las especificaciones del fabricante, $R$ se genera de manera uniforme entre $(1-\delta)R_0$ y $(1+\delta)R_0$. Esto se puede conseguir generando una variable aleatoria uniforme entre 0 y 1, $u \sim U(0,1)$, con el comando `rand(1)` y haciendo $R = [R_0(1+\delta) - R_0(1-\delta)] \times u + R_0(1-\delta) = 2\delta R_0 u + R_0(1-\delta)$. Seguidamente, simulando el proceso de medida, añadimos el ruido. Su función de densidad estará centrada en 0 y tendrá una varianza indicada por $\sigma^2_w$. Para generarla, obtenemos primero $w[n] \sim N(0,1)$ con el comando `randn(1,N)` y luego multiplicamos $w[n]$ por $\sigma_w$ para conseguir que tenga una varianza igual a $\sigma^2_w$. Como se ha dicho antes, el objetivo es estimar el valor de la resistencia $R$. Para ello, inicialmente vamos a presentar tres estimadores heurísticos. El término heurístico hace referencia a la capacidad práctica o informal de resolver un problema, sin atender a ningún criterio de optimidad fijado a priori.
4.3.1 Estimadores Heurísticos.

El primer estimador se define en la siguiente ecuación:

\[ \hat{R}_1 = \frac{1}{N} \sum_{n=1}^{N} Y[n] \]  

(4.2)

como se ve, simplemente consiste en la media aritmética de todos los datos.

Se van a implementar en Matlab los diferentes estimadores aquí expuestos. Como primer algoritmo tenemos el estimador \( \hat{R}_1 \):

```matlab
function R1 = Estimador1(Y)
  R1 = mean(Y);
```

En la Figura 4.2 se puede ver la comparación entre las diferentes medidas que se han realizado, \( Y[n] \), y el estimador \( \hat{R}_1 \) para un ejemplo con \( N = 10 \).

![Figura 4.2. Estimador R1](image_url)
Con este estimador estamos incluyendo valores que el fabricante asegura que no pueden ocurrir. Es decir, previamente se indicó que los valores de las resistencias estarían comprendidos entre 90 y 110 Ω, según el fabricante. Por tanto, como el ruido puede dar lugar a valores fuera de ese rango, se tendrán que modificar todos los valores que estén fuera de los límites. Por ejemplo, si se obtiene un valor $R = 120 \, \Omega$ este valor se podría tener en cuenta como un valor de $R = 110 \, \Omega$, que es el valor máximo de resistencia posible de acuerdo con las especificaciones del fabricante.

Por tanto se aplicará el siguiente sistema:

$$T(Y[n]) = \begin{cases} 
R_0(1 + \delta), & Y[n] \geq R_0(1 + \delta) \\
Y[n], & R_0(1 - \delta) < Y[n] < R_0(1 + \delta) \\
R_0(1 - \delta), & Y[n] \leq R_0(1 - \delta)
\end{cases} \quad (4.3)$$

Entonces, el segundo estimador:

$$\hat{R}_2 = \frac{1}{N} \sum_{n=1}^{N} T(Y[n]) \quad (4.4)$$

Este es el código empleado para simular el segundo estimador:

```matlab
function R2 = Estimador2(Y,R0,delta)

Y(Y<R0*(1-delta))=R0*(1-delta);
Y(Y>R0*(1+delta))=R0*(1+delta);

R2 = mean(Y);
```

Al igual que en el primer estimador, la Figura 4.3 muestra un ejemplo de los valores de $Y[n]$ usados para realizar la estimación junto con el valor de $\hat{R}_2$ obtenido.
El último estimador heurístico contemplado aquí es una variación del anterior. A diferencia del segundo, se descartan las observaciones fuera del rango en vez de forzarlas a estar dentro del mismo.

El código del estimador es el siguiente:

```matlab
function R2b = Estimador2b(Y,R0,delta)
Y1 = Y(Y<=R0*(1+delta) & Y>=R0*(1-delta));
R2b = mean(Y1);
```

En la Figura 4.4 se puede ver que no aparece ningún valor de $R$ fuera del rango de $90 \, \Omega$ a $110 \, \Omega$ y sólo se tienen en cuenta 7 de las 10 muestras disponibles en este ejemplo.
Definiendo $I = \{n: R_0(1 - \delta) \leq Y[n] \leq R_0(1 + \delta)\}$, podemos definir este estimador como:

$$\hat{R}_{2b} = \frac{1}{|I|} \sum_{n \in I} Y[n]$$

(4.5)

Donde $|I|$ denota la cardinalidad del conjunto $I$, esto es, su número de elementos.

### 4.3.2 Estimador MMSE

Una vez introducidos los tres estimadores heurísticos anteriores, se pasa ahora a analizar estimadores más complejos. Un estimador con un buen funcionamiento es el llamado estimador MMSE (Minimum Mean Squared Error). Como se explicó en el Capítulo 3, este estimador es el que minimiza el error cuadrático medio. El estimador MMSE viene dado por la siguiente expresión:

$$\hat{R}_{MMSE} = \int_{-\infty}^{\infty} R \cdot P(R|Y) \, dR$$

(4.8)
Donde \( P(R|Y) \) es la función de probabilidad a posteriori, que indica la probabilidad de obtener el valor del parámetro \( R \) conocido \( Y \) (conjunto de datos).

Según el teorema de Bayes, la probabilidad a posteriori se puede expresar de la siguiente manera (como vimos en el Capítulo 3):

\[
P(R|Y) = \frac{P(Y|R)P(R)}{P(Y)} \propto P(Y|R)P(R) \tag{4.9}
\]

Donde \( P(R) \) es la función de probabilidad a priori. Se ha comentado al principio que el fabricante asegura que el valor de las resistencias que se fabrican se encuentra en un rango definido por una tolerancia del 10% y el valor nominal \( R_0 = 100 \Omega \). Por tanto, la probabilidad a priori de este ejemplo es la que se indica en la siguiente expresión:

\[
P(R) = \begin{cases} 
\frac{1}{2\delta R_0}, & R_0(1 - \delta) \leq R \leq R_0(1 + \delta) \\
0, & \text{resto}
\end{cases} \tag{4.10}
\]

En este caso particular, siendo \( R_0 = 100 \) y \( \delta = 0,1 \), se tiene:

\[
P(R) = \begin{cases} 
\frac{1}{20}, & 90 \leq R \leq 110 \\
0, & \text{resto}
\end{cases} \tag{4.11}
\]

\( P(Y|R) \) hace referencia a la verosimilitud. Dado que el ruido es Gaussiano, para este ejemplo la verosimilitud viene descrita en la siguiente expresión:

\[
P(Y|R) = \prod_{n=1}^{N} N(Y[n]|R, \sigma_w^2) = \prod_{n=1}^{N} \frac{1}{\sqrt{2\pi \sigma_w^2}} \exp \left( -\frac{(Y[n] - R)^2}{2\sigma_w^2} \right) \tag{4.12}
\]

La ecuación anterior indica cómo se distribuyen los datos para un valor de \( R \) concreto. Como vimos anteriormente, cada medida sigue una distribución normal centrada en el valor real de la resistencia (que desconocemos) y con una varianza \( \sigma_w^2 \) que viene dada por el ruido. En cuanto al operador producto: puesto
que cada medida es independiente de la anterior, la teoría estadística señala que para varios eventos independientes su probabilidad conjunta es el producto de las probabilidades individuales de cada uno [De La Horra, 2003].

Es decir, supóngase que por algún motivo se conoce el valor de $R$. Si se realizan varias medidas con un multímetro se obtendrán una serie de valores ($Y[n]$). Como se comentó anteriormente, las medidas llevarán asociadas un ruido. Por consiguiente, la probabilidad de obtener el conjunto de medidas será el producto de las probabilidades de obtener el valor de cada medida, ya que se trata de sucesos independientes. El ruido seguirá una distribución normal, y puesto que se conoce el parámetro $R$, la función de probabilidad será una distribución normal centrada en el valor del parámetro $R$ y con varianza $\sigma^2_w$.

Una vez conocidas la probabilidad a priori y la verosimilitud se desarrolla la expresión (4.9), obteniéndose la probabilidad a posteriori:

$$P(R|Y) \propto \prod_{n=1}^{N} \frac{1}{\sqrt{2\pi \sigma^2_w}} \exp \left( -\frac{(Y[n] - R)^2}{2\sigma^2_w} \right) \times \frac{1}{2\delta R_0} \times I[R_0(1 - \delta) \leq R \leq R_0(1 + \delta)]$$  \hspace{1cm} (4.13)

Donde:

$$I[c] = \begin{cases} 1, & c \text{ true} \\ 0, & c \text{ false} \end{cases}$$

$I[c]$ se conoce con el nombre de función indicadora. Y consecuentemente, en este caso particular $I[c] = I[R_0(1 - \delta) \leq R \leq R_0(1 + \delta)]$.

4.3.3 Estimador de Monte Carlo

El estimador MMSE de la expresión (4.8) hace uso de $P(R|Y)$. Sin embargo, resolver la integral no es siempre posible, ya que $P(R|Y)$ puede ser una distribución que no sigue un modelo matemático que nos permita calcular la integral de manera analítica. El concepto de la estimación de Monte Carlo visto en el Capítulo 3 nos indica que una posible solución en estos casos es obtener muestras de esta distribución para posteriormente llevar a cabo una integración numérica. Como ejemplo de ello tenemos el siguiente estimador:

$$\hat{R}_3 = \frac{1}{T} \sum_{t=1}^{T} R_t$$  \hspace{1cm} (4.14)

Donde $R_t$ son muestras obtenidas directamente de $P(R|Y)$, dada por la ecuación (4.13).
La teoría de los métodos de Monte Carlo establece que, si se obtiene un número de muestras suficientemente grande, se cumple lo siguiente:

\[
\lim_{{T \to \infty}} \frac{R}{\hat{R}} = R_{\text{MMSE}}
\]

(4.15)

La dificultad de los métodos de Monte Carlo reside en generar las muestras \(R_t\). En este ejemplo concreto podemos hacerlo desarrollando la expresión (12) de tal modo que obtengamos una FDP de \(R\) de la que sea fácil muestrear. Este método es conocido por el nombre de completar cuadrados. Si definimos \(v(R) = \frac{1}{2\delta R_0} I[R_0(1-\delta) \leq R \leq R_0(1+\delta)]\), tenemos:

\[
P(R|Y) \propto \prod_{{n=1}}^{N} \frac{1}{\sqrt{2\pi\sigma^2_w}} \exp\left(\frac{(Y[n] - R)^2}{2\sigma^2_w}\right) \times v(R)
\]

Si desarrollamos el producto, obtenemos una suma de exponentes:

\[
P(R|Y) \propto \frac{1}{(2\pi\sigma^2_w)^\frac{N}{2}} \exp\left(-\frac{1}{2\sigma^2_w} \sum_{{n=1}}^{N} (R - Y[n])^2\right) \times v(R)
\]

Tomando únicamente el sumatorio dentro de la exponencial y desarrollándolo:

\[
\sum_{{n=1}}^{N} (R - Y[n])^2 = \sum_{{n=1}}^{N} (R^2 - 2Y[n]R + Y[n]^2) = NR^2 - 2 \left(\sum_{{n=1}}^{N} Y[n]\right) R + \sum_{{n=1}}^{N} Y[n]^2
\]

Renombrando algunos términos:

\[
N\mu_y = \sum_{{n=1}}^{N} Y[n]
\]
\[
N\sigma^2_y = \sum_{{n=1}}^{N} Y[n]^2
\]

\[
\sum_{{n=1}}^{N} (R - Y[n])^2 = NR^2 - 2N\mu_y R + N\sigma^2_y = (R^2 - 2\mu_y R + \sigma^2_y) \times N
\]

\[
(R - \mu_y)^2 = R^2 - 2\mu_y R + \mu^2_y + \sigma^2_y - \sigma^2_y
\]
Y por tanto (teniendo en cuenta que es una proporción):

\[
P(R|Y) \propto \frac{1}{(2\pi \sigma_w^2)^N} \times \exp\left(-\frac{(R - \mu_y)^2}{2 \sigma_w^2 N}\right) \times \exp\left(-\frac{(-\mu_y^2 + \sigma_y^2)}{2 \frac{\sigma_w^2}{N}}\right) \times v(R)
\] (4.16)

De la anterior expresión se pueden obtener muestras \( R_t \), ya que

\[
\frac{1}{(2\pi \sigma_w^2)^N} \times \exp\left(-\frac{(R - \mu_y)^2}{2 \sigma_w^2 N}\right) = N \left(\mu_y, \sigma_w^2 \frac{N}{N}\right)
\] (4.17)

Y por tanto:

\[
P(R|Y) \propto N \left(\mu_y, \sigma_w^2 \frac{N}{N}\right) \times v(R)
\] (4.18)

Así pues, tenemos que

\[
R_t \sim N \left(\mu_y, \sigma_w^2 \frac{N}{N}\right) \times u \left(R_0(1 - \delta), R_0(1 + \delta)\right)
\] (4.19)

El código del estimador recientemente expuesto es el que se muestra a continuación. Nótese que para generar los valores de \( R_t \) se usa el algoritmo de aceptación – rechazo explicado en el Capítulo 3: se generan muestras de \( N \left(\mu_y, \sigma_w^2 \frac{N}{N}\right) \) y se rechazan todas aquellas que están fuera del rango. Este procedimiento se repite hasta obtener el número de muestras deseado, N.

```matlab
function R3 = Estimador3(Y,sigma2_w,R0,delta)

[r,N]=size(Y);
T = 10000; %Numero de iteraciones
uy = sum(Y);
Rn = zeros(1,T);
for t = 1:T
    repeat=true;
    while repeat==true
```

Proyecto Fin de Grado

57
\[ R_n(t) = \frac{u_y}{N} + \sqrt{\sigma_2 w/N} \cdot \text{randn}(1); \]

\[
\text{if (ge}(R_n(t), R_0(1-\delta)) \text{&& le}(R_n(t), R_0(1+\delta)))
    \text{repeat=false;}
\text{end}
\text{end}
\]

\[ R_3 = \text{mean}(R_n); \]

En la Figura 4.5 se muestra su funcionamiento para \( N = 1000 \) muestras.

![Comparación de las muestras con el estimador R3](image)

Figura 4.5. Estimador R3.

Como se aprecia en la figura anterior, el número de muestras es mayor que en los casos anteriores ya que así lo hemos elegido. Como se indicó previamente, cuanto mayor sea el número de iteraciones del algoritmo mejor será su funcionamiento.
4.3.4 **Algoritmo de Metropolis-Hastings**

En muchos casos no es posible obtener muestras de la distribución de probabilidad a posteriori directamente. Utilizando el algoritmo de Metropolis-Hastings podremos obtener muestras válidas partiendo de otra distribución más sencilla.

En el Capítulo 3 podemos ver una explicación genérica del algoritmo aquí usado. En el presente Capítulo se explica paso por paso cómo se utiliza el algoritmo particularizado para este ejemplo.

1. Se elige un estado inicial $R_0$.
2. Se toma una distribución del parámetro a estimar, $q(R)$, para generar parámetros candidatos. Por ejemplo, una distribución normal centrada en $R_0$ y de varianza $\sigma_R^2$:
   \[
   q(R) = N(R|R_0, \sigma_R^2) = \frac{1}{\sqrt{2\pi\sigma_R^2}} \exp\left(-\frac{(R - R_0)^2}{2\sigma_R^2}\right)
   \]

En la iteración $t$ ($t = 1, 2, ..., T$) se realizan los siguientes pasos:
3. Se genera una muestra de la distribución supuesta en el punto 2:
   
   $R' \sim q(R)$
4. Se genera un valor aleatorio entre 0 y 1 que servirá para comparar con la probabilidad que se obtiene posteriormente y permitirá aceptar o rechazar el valor estimado en el punto 3.
   
   $u' \sim U(0,1)$
5. Se calcula la probabilidad de aceptación:
   
   $\alpha = \min\left(1, \frac{q(R_{t-1})P(R'|Y)}{q(R')P(R_{t-1}|Y)}\right)$
6. Si $u' \leq \alpha$ se acepta $R'$, y por tanto $R_t = R'$ (t hace referencia a la iteración). En caso de que $u' > \alpha$ se rechaza $R'$. Entonces $R_t = R_{t-1}$.
7. Finalmente, una vez se han llevado a cabo todas las iteraciones se tiene un conjunto de valores $R_t$. A partir de ellos se calcula el estimador:
   
   $\hat{R}_4 = \frac{1}{T - T_b} \sum_{t=T_b+1}^{T} R_t$  \hspace{1cm} \text{(4.20)}

Donde $T_b$ es el periodo de “burn-in”.

**Algoritmo 4. Algoritmo de Metropolis-Hastings**
Se muestra el código empleado:

```matlab
%Algoritmo M-H
function R4 = Estimador4(Y,R,R0,delta,sigma2_r,sigma2_w)

T = 10000; %Número de iteraciones del algoritmo
Tburnin = 0.9*T;
Rt=zeros(1,T); %Array donde guardaremos los valores del parámetro
Rt(1)=R0;
Rmed = mean(Y);

%Creamos una función indicador
f = (@(x) (R0*(1-delta) <= x) & (x <= R0*(1+delta)));

for t = 2:T+1
    Rtemp = R0 + randn(1)*sqrt(sigma2_r);
    utemp = rand(1);
    num = exp(-((Rt(t-1)-R0)^2)/(2*sigma2_r)) * ...
         exp((-1/(2*sigma2_w))*sum((Y-Rtemp).^2)) * f(Rtemp);
    den = exp(-((Rtemp-R0)^2)/(2*sigma2_r)) * ...
         exp((-1/(2*sigma2_w))*sum((Y-Rt(t-1)).^2));
    n = log(num);
    d = log(den);
    alfa = exp(n-d);
    if alfa > utemp
        Rt(t)=Rtemp;
    else
        Rt(t)=Rt(t-1);
    end
end
R4 = mean(Rt(Tburnin + 1:T+1));
```

En la Figura 4.6 se muestran las muestras generadas junto al estimador \( \hat{R}_4 \) y el valor real \( R \). Se han descartado las muestras del tiempo de “burn-in”.
A partir del valor $R$, que es el valor real, y los diferentes estimadores vistos se puede calcular el error cuadrático medio o MSE ("mean squared error"). Calculando los estimadores $M = 1000$ veces (para que el valor del MSE se asemeje más a la realidad) se pueden obtener 4 conjuntos de valores asociados a los mismos:

$$MSE1(m) = (R - \hat{R}_1)^2$$
$$MSE2(m) = (R - \hat{R}_2)^2$$
$$MSE2b(m) = (R - \hat{R}_2b)^2$$
$$MSE3(m) = (R - \hat{R}_3)^2$$
$$MSE4(m) = (R - \hat{R}_4)^2$$
Nótese que $R$ también varía en cada prueba, ya que se obtiene uniformemente entre $R_0(1 - \delta)$ y $R_0(1 + \delta)$. En la Figura 4.7 se muestra una comparación de los errores cuadráticos medios:

![Figura 4.7. Comparación de los errores de cada algoritmo.](image)

Se aprecia en la figura que, excepto el estimador 2b que tiene un comportamiento ineficiente, el resto de algoritmos funcionan de manera muy similar.

El comportamiento exageradamente malo del segundo estimador se puede explicar por la cantidad de valores que se descartan para hacer el promedio final. El algoritmo que mejor debe funcionar es el algoritmo 3, que se identificaba con el algoritmo de Monte Carlo. Si se tiene la posibilidad de extraer muestras de la distribución que sigue el parámetro podremos tener una estimación bastante cercana al valor real.

Hemos comentado anteriormente que el algoritmo de Metropolis-Hastings funciona mejor si se descartan las muestras correspondientes al tiempo de burning. En la Figura 4.8 ilustramos la comparación cuando tenemos tiempo de burning frente a cuando no lo tenemos.
Figura 4.8. Comparaciones entre errores cuadráticos medios en M-H con y sin tiempo de burn-in.

Otra representación interesante es la del error cuadrático medio en función de la cantidad de tiempo de “burn-in”. Hasta ahora, el tiempo de “burn-in” era el 90% de las muestras. En la Figura 4.9 se muestra el error cuadrático medio (MSE) en función del tiempo de “burn-in”. En el eje x se representa el tiempo de “burn-in” en intervalos del 10%.
Como puede verse, un período de “burn-in” del 30% de las muestras parece ser óptimo en este caso, aunque tampoco hay realmente una gran diferencia en los resultados.

Por último, tenemos la comparación entre los sesgos de cada estimador (Figura 4.10). Como vimos en el Capítulo 3, el sesgo es la diferencia entre la esperanza del estimator y el verdadero valor del parámetro a estimar.
Los estimadores $MSE_2$ y $MSE_2b$ introducen sesgo. El estimador $MSE_2$ compensa este sesgo reduciendo la varianza, a diferencia del $MSE_2b$. En la Figura 4.11 se representa esta varianza de los estimadores.
4.4 Discusión

En este primer caso práctico se ha llevado a cabo un primer acercamiento a las herramientas presentadas de forma teórica en el Capítulo 3. Se ha partido de un ejemplo sencillo, cercano al campo de la electrónica: la estimación del valor de una resistencia a partir de un conjunto de medidas ruidosas.

Primero, se han tenido en cuenta las especificaciones del fabricante, las cuales se han introducido en nuestro modelo matemático de partida. Seguidamente se han tenido en cuenta las limitaciones del proceso de medida, para lo cual se ha añadido una distribución de ruido. Teniendo en cuenta las características anteriores, se han podido generar una serie de datos sintéticos. Partiendo de los mismos se han podido llevar a cabo las primeras estimaciones utilizando para ello estimadores heurísticos.

Por otra parte, conociendo la información a priori y la verosimilitud podemos obtener el modelo matemático que identifica a la función de probabilidad a posteriori. A partir del mismo se han podido obtener muestras y llevar a cabo un promediado de ellas, lo que se conoce como método de Monte Carlo.

Figura 4.11. Varianza de los estimadores.
Seguidamente se contempló la posibilidad de que la generación de muestras fuese inviable. Esto nos llevó a abordar el algoritmo de Metropolis-Hastings, que hace uso del algoritmo de aceptación-rechazo que se vio en el Capítulo 3.

Por último, se ha llevado a cabo una comparación de todos los estimadores empleados. Cabe destacar cómo los dos últimos estimadores (estimadores no heurísticos) ofrecían un mejor comportamiento que el de los estimadores heurísticos.
Capítulo 5. Distribución binomial

5.1 Introducción teórica.

Mostraremos el funcionamiento del algoritmo Consensus Monte Carlo visto en el Capítulo 3. El primer ejemplo parte de unos datos. Éstos consisten en 1000 experimentos de Bernoulli con un único éxito. En teoría estadística, un experimento de Bernoulli es un suceso aleatorio que tiene dos posibles resultados: éxito o fracaso. Cada vez que se lleva a cabo un experimento de Bernoulli tenemos la misma probabilidad de éxito.

Por tanto, definimos una distribución de Bernoulli como una distribución de probabilidad discreta que viene caracterizada por una probabilidad de éxito $p$. Una distribución de Bernoulli tiene la siguiente notación matemática [De La Horra, 2003]:

$$y \sim \text{Be}(p) \quad (5.1)$$

Un ejemplo sencillo y conocido de experimento de Bernoulli es el lanzamiento de una moneda. Hay dos posibles resultados, que salga cara (éxito) o que salga cruz (fallo), y una probabilidad de éxito que suponemos del 50% si la moneda no tiene más peso por alguna de sus partes (no está trucada).

La distribución binomial es una distribución de probabilidad discreta que cuenta el número de éxitos en una secuencia de $n$ ensayos de Bernoulli independientes entre sí, con una probabilidad fija de ocurrencia del éxito $p$ para cada ensayo.

La distribución de Bernoulli es un caso particular de la distribución Binomial en la que el número de ensayos es igual a uno. La distribución Binomial se representa de esta manera [De La Horra, 2003]:

$$y \sim \text{B}(n, p) \quad (5.2)$$

Donde $n$ indica el número de ensayos y $p$ la probabilidad de los mismos.

Tomamos como información a priori una distribución uniforme. Es decir, nuestra información antes de llevar a cabo una estimación es que el éxito y el fracaso son igual de probables. Nótese que esta información a priori contradice el resultado del experimento, ya que, a pesar de tener 1000 datos de los cuales solo hay un éxito, decimos que la probabilidad de éxito es igual a la probabilidad de error.
El objetivo de este ejemplo será mostrar el funcionamiento del algoritmo Consensus Monte Carlo y determinar qué parámetros hacen que funcione de manera correcta. Como vimos anteriormente en la teoría, el procedimiento general del algoritmo Consensus Monte Carlo sigue los siguientes pasos:

1. Generar/obtener los datos globales, \( y \)
2. Dividir \( Y \) en shards (grupos de datos): \( y_1, \ldots, y_S \)
3. Ejecutar G algoritmos de Monte Carlo independientes para cada uno de los shards, obteniendo:

\[
\theta_{sg} \sim p(\theta|y_s) \propto p(y_s|\theta)p(\theta)^{1/s}
\]

Para \( s = 1, \ldots, S \) y \( g = 1, \ldots, G \)
4. Combinar las estimas de los shards usando una mezcla lineal promediada:

\[
\theta_g = \left( \sum_{s=1}^{S} w_s \right)^{-1} \left( \sum_{s=1}^{S} w_s \theta_{sg} \right)
\]

donde \( w_s \) hace referencia a la matriz de pesos.

**Algoritmo 5. Consensus Monte Carlo**

*Cada shard es asignado a un worker.* Cabe destacar que si cada worker devolviese una función matemática (esto es, una función de densidad de probabilidad) simplemente las combinaríamos mediante el producto de las mismas. La dificultad aparece porque cada worker devuelve un conjunto de estimaciones de Monte Carlo, las cuales tienen que ser combinadas.

La forma de combinar las estimaciones de los workers del paso 4 es una combinación óptima para el caso gaussiano. Otra forma más fácil de realizar la combinación sería simplemente promediando:

\[
\theta_g = \frac{1}{S} \sum_{s=1}^{S} \theta_{sg} \quad (5.3)
\]

En este ejemplo veremos que la combinación para el caso gaussiano funciona bien también para casos no gaussianos.

Cuando cada \( p(\theta|y_s) \) es Gaussiana, la combinación posterior \( p(\theta|y) \) también es Gaussiana. De la expresión del paso 4 del algoritmo podemos obtener estimaciones iguales a las que obtendríamos si lo realizásemos directamente desde \( p(\theta|y) \). Para ver esto suponemos \( S = 2 \) (número de shards) y que \( p(\theta|y_1) \sim N(\mu_1, \Sigma_1) \) y \( p(\theta|y_2) \sim N(\mu_2, \Sigma_2) \).
Siguiendo los cálculos de los estándares Bayesianos obtendríamos lo siguiente:

\[ p(\theta|y) \propto p(\theta|y_1)p(\theta|y_2) \propto N(\theta|\bar{\mu}, V) \]  \hspace{1cm} (5.4)

Donde \( V^{-1} = \Sigma_1^{-1} + \Sigma_2^{-1} \) y \( \bar{\mu} = V(\Sigma_1^{-1}\mu_1 + \Sigma_2^{-1}\mu_2) \). Ahora, si \( \theta_1 \sim N(\mu_1, \Sigma_1) \) y \( \theta_2 \sim N(\mu_2, \Sigma_2) \) se verifica que:

\[ V(\Sigma_1^{-1}\theta_1 + \Sigma_2^{-1}\theta_2) \sim N(\bar{\mu}, V) \]  \hspace{1cm} (5.5)

Podemos ver la siguiente relación:

\[ w_s = \Sigma_s^{-1} \]  \hspace{1cm} (5.6)

La matriz de pesos de la expresión (5.6) es óptima para los modelos gaussanos, donde \( \Sigma_s = Var(\theta|y_s) \).

### 5.2 Ejecución práctica: Primer Caso

Una vez explicado cómo será el procedimiento para realizar el Consensus Monte Carlo en este caso proseguimos con el ejemplo. Nos encontrábamos en la siguiente situación:

- Teníamos 1000 pruebas de Bernoulli con un solo éxito (estos eran nuestros datos).
- Nuestra información a priori era una distribución uniforme.

Primero llevaremos a cabo la estimación global. Esta nos servirá para compararlo con la que obtendremos al distribuir los datos en varios workers y así ver si el funcionamiento del Consensus Monte Carlo es correcto.

#### 5.2.1 Estima global

Para realizar la estima global se generan \( G \) muestras de Monte Carlo a partir de la distribución a posteriori:

\[ \theta_g \sim p(\theta|y) \propto p(y|\theta)p(\theta) \]  \hspace{1cm} (5.7)
Para obtener la distribución a posteriori utilizaremos una herramienta de la teoría Bayesiana, conocida como *conjugate prior*. Esta herramienta establece que, si la distribución a priori es la conjugada de la función de verosimilitud, entonces la distribución a posteriori pertenecerá a la misma familia que la distribución a priori. [Raudenbush, 2000].

En este ejemplo sabemos que la distribución a posteriori es una distribución Beta. Una distribución Beta está definida en el intervalo [0,1] y viene caracterizada por dos parámetros: $\alpha$ y $\beta$ [Walpole, 1999]. Como caso particular hay que destacar que para valores de $\alpha$ y $\beta$ igual a 1, la distribución Beta se corresponde con una distribución uniforme. En este ejemplo se demostrará el uso de la distribución Beta como distribución a posteriori.

La distribución a priori es una distribución uniforme. Es decir: podemos afirmar que la función de distribución de probabilidad a priori es la siguiente:

$$p(\theta) = Beta(1,1)$$  \hspace{1cm} (5.8)

Por tanto, escribiendo la expresión (5.8) de forma genérica:

$$p(\theta) = Beta(\alpha, \beta) = \theta^{\alpha-1}(1-\theta)^{\beta-1}$$  \hspace{1cm} (5.9)

Donde $\alpha = \beta = 1$ en nuestro caso.

Por otra parte, como datos teníamos 1000 pruebas de Bernoulli con un solo éxito. La función de probabilidad para una única prueba de Bernoulli viene expresada de la siguiente manera:

$$P\{Y_i = K|\theta\} = \begin{cases} \theta, & k = 1 \\ 1 - \theta, & k = 0 \end{cases}$$  \hspace{1cm} (5.10)

Donde $\theta$ hace referencia a la probabilidad de éxito y $k$ es el resultado del experimento, el cual puede ser éxito (1) o fracaso (0).

En la expresión (5.10) se hacía referencia a un experimento individual, es decir, una prueba de Bernoulli. En el ejemplo, se ha explicado que como datos se toman 1000 experimentos (intentos) de Bernoulli. Por tanto, la siguiente expresión hace referencia a la función de probabilidad de un conjunto ($n$) de experimentos:

$$P(y = [y_1, ..., y_n]|\theta) = \prod_{i=1}^{n} \theta^{y_i}(1-\theta)^{(1-y_i)}$$  \hspace{1cm} (5.11)
Recordamos ahora el teorema de Bayes, que permite obtener la distribución a posteriori:

\[
p(\theta|y) = \frac{P(y|\theta) \times p(\theta)}{p(y)} \propto P(y|\theta) \times p(\theta)
\] (5.12)

Por tanto, conociendo la distribución a priori y la verosimilitud, operamos:

\[
p(\theta|y) \propto \left( \prod_{i=1}^{n} \theta^{y_i}(1 - \theta)^{(1-y_i)} \right) \theta^{\alpha-1}(1 - \theta)^{\beta-1}
\]

\[
= \theta^{\alpha + \sum_{i=1}^{n} y_i - 1} \times (1 - \theta)^{\beta + n - \sum_{i=1}^{n} y_i - 1}
\] (5.13)

Si comparamos la expresión (5.13) con la (5.9) se puede concluir que la distribución a posteriori se identifica con una distribución Beta:

\[
p(\theta|y) = Beta \left( \alpha + \sum_{i=1}^{n} y_i, \beta + n - \sum_{i=1}^{n} y_i \right)
\] (5.14)

Una vez demostrada la relación *conjugate prior* (si la FDP a priori es del tipo Beta y la verosimilitud sigue una distribución binomial, la FDP a posteriori también es Beta), abordamos el primer objetivo: realizar la estimación global. Observando la expresión (5.14) y teniendo en cuenta que \(\sum_{i=1}^{n} y_i\) hace referencia al número de éxitos de nuestros datos (1, en nuestro caso), \(\alpha = \beta = 1\) en nuestro caso y \(n\) es el número de intentos de Bernoulli (1000, para este caso particular) podemos formular la distribución a posteriori de este ejemplo:

\[
p(\theta|y) = Beta \left( \alpha + \sum_{i=1}^{n} y_i, \beta + n - \sum_{i=1}^{n} y_i \right) = Beta(2, 1000)
\] (5.15)
En la Figura 5.1 se dibuja la distribución de la expresión (5.15).

![Figura 5.1. Distribución Beta (α=2 y β=1000)](image)

Una vez conocida la distribución a posteriori, procedemos a aplicar el método de Monte Carlo: se generan G muestras de Monte Carlo, y seguidamente se realiza un promediado de las mismas. Como resultado, se obtiene la estima global del parámetro $\theta$. El valor estimado se identifica mediante $\hat{\theta}$. En la siguiente expresión se indica el estimador de forma analítica:

$$\hat{\theta} = \frac{1}{G} \sum_{g=1}^{G} \theta_g$$

(5.16)

Al igual que en el ejemplo del Capítulo anterior, se utilizará Matlab como herramienta de simulación. En las siguientes líneas de código se muestra la generación de las 1000 muestras de Bernouilli con un único acierto, que son los datos de los cuales partimos. A continuación, se generan G muestras (en este caso particular G = 1000) a partir de la distribución Beta (5.15), y se lleva a cabo el promediado, como indica la expresión (5.16).
Generamos los datos: 1000 resultados de Bernoulli con un solo acierto
Y = [1 zeros(1,999)];
Y = Y(randperm(size(Y,2)));

%ESTIMACION GLOBAL
Pg = betarnd(2,1000,1,10000);
Pgf = mean(Pg);

De esta manera se lleva a cabo la estima global usando G = 1000 muestras de Monte Carlo. Ahora se llevará a cabo la estima local. Esta estima local se puede llevar a cabo en diferentes máquinas o núcleos.

5.2.2 Estima local

Como se ha indicado anteriormente, la estima local consiste en dividir la estima global en diferentes centros de cálculo, que pueden ser máquinas o núcleos computacionales. En el Capítulo 3 se puede encontrar una explicación más extensa acerca de estos detalles.

En el presente ejemplo es importante, para la comprensión del lector, saber que se dispone de varias máquinas. Para este caso particular, las mismas reciben el nombre de workers.

Una vez se han generado los datos, seguimos los pasos del Algoritmo 1 detallado anteriormente. Para este caso particular se dividen los datos en 100 grupos (shards) de 10 muestras cada uno, y se asigna 1 shard a cada worker. Nótese que 99 workers recibirán 10 muestras idénticas e iguales a cero, y sólo 1 worker recibirá una de sus 10 muestras no nula.

En las siguientes líneas de código se muestra cómo se ha llevado a cabo la generación de los datos y su división en shards.

Y = [1 zeros(1,999)];
Y = Y(randperm(size(Y,2)));%

%Dividimos los datos en 100 shards, con 10 datos cada uno
Ys = buffer(Y,10);

Seguidamente, a partir de cada shard se generan G muestras de Monte Carlo. Como se ha comentado anteriormente G = 1000. Se considera que este es un valor suficiente que se encuentra entre los márgenes de coste computacional y eficiencia del algoritmo de Monte Carlo.

Como se indicó en la introducción teórica, el objetivo es valorar qué parámetros intervienen en el rendimiento del algoritmo Consensus Monte Carlo. Por tanto, partiendo del teorema de Bayes (expresión 5.12), podemos obtener fácilmente la distribución de la que tenemos que generar las muestras de cada shard para nuestro caso:

\[ \theta_{sg} \sim p(\theta | y_s) \propto p(y_s | \theta)p_s(\theta) \] (5.17)
Donde $p_s(\theta)$ es la FDP a priori de cada shard $e y_s$ (para $s = 1, ..., S$) indica el número de datos disponibles en el mismo.

A la hora de generar la distribución a posteriori se tenían en cuenta los datos y la distribución a priori. En el caso de la estimación local, se parte de un grupo de datos más pequeño, lo que influirá en la distribución a posteriori. Cabe destacar que carecemos de libertad a la hora de decidir sobre esta información puesto que los datos son los que son.

En el caso de la estimación local se probará con dos distribuciones a priori diferentes:

- $p_s(\theta) = Beta(1,1)$
- $p_s(\theta) = Beta(0.01, 0.01)$

Frente a la no libertad de poder elegir la verosimilitud, se tiene la posibilidad de tomar diferentes distribuciones a priori. Uno de los objetivos del presente ejemplo es analizar cuál es la distribución a priori más apropiada para acercarnos al resultado de la estima global.

En la Figura 5.2 dibujamos la primera de ellas.

![Figura 5.2. Distribución Beta ($\alpha=1$ y $\beta=1$).](image)
Por tanto, tenemos la misma probabilidad de obtener cualquier valor entre 0 y 1, lo cual es altamente incorrecto, puesto que en 1000 experimentos se ha obtenido un solo éxito, lo cual parece indicar que la probabilidad de fracaso es mucho mayor que la de éxito.

Con la primera distribución se está asegurando que tanto el éxito como el fracaso tienen la misma probabilidad, lo cual, observando los datos, es incorrecto. Por tanto, la estimación a partir de esa información a priori debería ser incorrecta, tal y como se afirma en [Scott, 2013].

La FDP a posteriori en este caso será:

\[
p(\theta|y_s) = Beta\left(1 + \sum_{i=1}^{n_s} y_s(i), 1 + n_s - \sum_{i=1}^{n_s} y_s(i)\right)
\]  (5.18)

Donde \( n_s \) es el número de muestras asignadas a cada shard (\( n_s = 10 \) en nuestro caso), \( \alpha = \beta = 1 \) (distribución a priori uniforme como se muestra en la Figura 5.1), y \( \sum_{i=1}^{n_s} y_s(i) = 0 \) para 99 de los 100 shards y \( \sum_{i=1}^{n_s} y_s(i) = 1 \) para el otro shard. Por tanto,

\[
p(\theta|y_s) = Beta(1,11)
\]

Para 99 shards, y

\[
p(\theta|y_s) = Beta(2,10)
\]

Para el shard restante.

A continuación explicamos el uso de \( p_s(\theta) = Beta(0.01,0.01) \) como distribución a priori. Partimos de la siguiente igualdad:

\[
p(\theta|y) = p(\theta|y_1, ..., y_S) \propto p(y_1, ..., y_S|\theta)p(\theta)
\]  (5.19)

Como cada lote de observaciones, dado \( \theta \), son condicionalmente independientes entre sí:

\[
p(y_1, ..., y_S|\theta) = \prod_{s=1}^{S} p(y_s|\theta)
\]  (5.20)

Por tanto:

\[
p(\theta|y) \propto \left(\prod_{s=1}^{S} p(y_s|\theta)\right)p(\theta) = \prod_{s=1}^{S} p(y_s|\theta)p(\theta)^{\frac{1}{S}}
\]  (5.21)
Con las anteriores expresiones se ha demostrado el porqué acerca del uso de \( p_s(\theta) = Beta(0.01, 0.01) = p(\theta)^{\frac{1}{S}} \) como distribución a priori. En este ejemplo \( S = 100 \), por tanto se tiene \( p(\theta)^{\frac{1}{S}} = Beta(0.01, 0.01) \). En la Figura 5.3 se representa dicha FDP a priori.

![Figura 5.3. Distribución Beta (α=0.01 y β=0.01)](image_url)

Con la segunda distribución se les sigue asignando la misma probabilidad al éxito y al fracaso, pero ahora se favorecen las probabilidades extremas (bien el éxito muy probable o bien el fracaso muy probable), lo cual concuerda mejor con los datos observados. En este caso, la distribución se acerca más a la realidad, puesto que muestra que el 0 y el 1 son los resultados más probables. La información a priori es muy importante, ya que dar información de éxito adicional tiene una influencia muy notoria en la estimación de la distribución a posteriori.

Se obtiene la distribución a posteriori cuando se toma \( p_s(\theta) = Beta(0.01, 0.01) \) como distribución a priori:

\[
p(\theta | y_s) = Beta \left( \frac{1}{100} + \sum_{i=1}^{n_s} y_s(i), \frac{1}{100} + n_s - \sum_{i=1}^{n_s} y_s(i) \right)
\]

(5.22)
Una vez establecidas las distribuciones a posteriori se generan las muestras de Monte Carlo para las dos distribuciones a priori consideradas. En las siguientes líneas de Matlab se muestra como lleva a cabo:

```matlab
for i=1:100
    %Estimamos 1000 muestras de Monte Carlo
    P1(:,i) = betarnd(1+sum(Ys(:,i)),1+10-sum(Ys(:,i)),1,1000);
    P2(:,i) = betarnd(0.01+sum(Ys(:,i)),0.01+10-sum(Ys(:,i)),1,1000);
    Wsp1(i) = var(P1(:,i)).^-1;
    Wsp2(i) = var(P2(:,i)).^-1;
end
```

Se pasa ahora a analizar las dos últimas líneas del bucle FOR. El siguiente paso, tal y como describe el algoritmo de consensus Monte Carlo, es llevar a cabo la combinación de las estimas de los workers para obtener la estima global. Para ello se utiliza la siguiente fórmula:

\[
\theta_g = \left( \sum_{s=1}^{S} w_s \right)^{-1} \left( \sum_{s=1}^{S} w_s \theta_{sg} \right)
\]

(5.23)

Dentro de la anterior expresión, \(w_s\) hace referencia a la matriz de pesos. Para el caso Gaussiano, tal y como se vio en el Capítulo 3 y se recordó anteriormente, la matriz de pesos viene determinada de la siguiente forma:

\[
w_s = Var(\theta | y_s)^{-1}
\]

(5.24)

A pesar de tratarse de un caso no gaussiano, analizaremos el rendimiento del algoritmo utilizando las matrices de peso óptimas para el caso gaussiano. En las dos últimas líneas del bucle (Wsp1 y Wsp2) se ha llevado a cabo la definición de la matriz de pesos (expresión 5.24).

Además, se utilizará otra matriz de pesos alternativa conocida como Equal Weights Fusion Matrix, y cuyos elementos toman todos el siguiente valor:

\[
w_s = \frac{1}{S} = \frac{1}{100}
\]

(5.25)

A continuación, una vez llevada a cabo la combinación de las estimas de los workers, se procede a la estimación del parámetro \(\theta\). Se obtienen \(G = 1000\) muestras de Monte Carlo y se realiza un promediado:

\[
\hat{\theta} = \frac{1}{G} \sum_{g=1}^{G} \theta_g
\]

(5.26)
En las siguientes líneas de código se muestra cómo se genera la matriz de pesos de la expresión 5.25, la combinación de las estimas de los shards, y el promediado de las muestras (expresión 5.26):

\[ W_{s1} = \frac{1}{100}; \]

\[ P_a = (\text{sum}(W_{s1})^{-1}) \times (\text{sum}(W_{s1} \times P_1')); \]
\[ P_b = (\text{sum}(W_{s1})^{-1}) \times (\text{sum}(W_{s1} \times P_2')); \]
\[ P_c = (\text{sum}(W_{sp1})^{-1}) \times (\text{sum}(W_{sp1}(1,i) \times P_1')); \]
\[ P_d = (\text{sum}(W_{sp2})^{-1}) \times (\text{sum}(W_{sp2}(1,i) \times P_2')); \]

\[ P_{af} = \text{mean}(P_a); \]
\[ P_{bf} = \text{mean}(P_b); \]
\[ P_{cf} = \text{mean}(P_c); \]
\[ P_{df} = \text{mean}(P_d); \]

Se procede ahora a la ejecución del algoritmo en Matlab. Se lleva a cabo una representación en forma de histograma de los diferentes resultados obtenidos.

En el primer caso se utiliza la distribución a priori uniforme, mostrada en la Figura 5.2 y la matriz de pesos *Equal Weights Fusion*:

- \( p_s(\theta) = \text{Beta}(1,1) \)
- Utilizando \( w_s = \frac{1}{s} \) para \( s = 1, ..., 100 \).
Como se aprecia en la Figura 5.4, la estimación es errónea: el valor del parámetro estimado tiende a valores entre 8 y 11 cuando debería ser aproximadamente igual a 0. Como se vio en los datos, los valores deberían ser 0 ó 1. Consecuentemente, esta estimación no puede tomarse como correcta.

En la siguiente ejecución se toma una distribución a priori más adecuada, tal y como se explicó anteriormente. La información a priori tomada puede verse representada en la Figura 5.3. En este caso se indica a la estimación Bayesiana que los valores con más probabilidad serán aquellos cercanos a 0 y 1. En este caso se utiliza una matriz de pesos iguales, es decir cada worker aporta una estimación igual de valiosa que la de los otros. Por tanto:

- Utilizando $p_s(\theta) = Beta(0.01, 0.01)$
- Utilizando $w_s = \frac{1}{S}$

![Figura 5.4. Histograma caso 1.](image)
Como se aprecia en la Figura 5.5, ahora los valores más frecuentes están más cercanos al 0. Esta distribución a posteriori es mucho más razonable que la anterior, ya que se encuentra mucho más cercana a los datos de los que partimos.

Utilizamos ahora de nuevo la distribución a priori errónea, sin embargo se prueba a realizar la combinación de las estimas con una matriz de pesos adecuada al caso gaussiano. Aunque se trata, claramente, de un caso no gaussiano, se comprueba el buen funcionamiento de esta matriz. Por tanto, se utiliza:

- \( p_θ(θ) = Beta(1,1) \)
- Utilizando \( w_s = Var(θ | y_s) \)

En la Figura 5.6 se muestra el resultado.
El valor del parámetro estimado es 100 veces menor, aunque la forma de la distribución es similar a la del caso 1. Esto nos lleva a destacar la importancia e influencia de la matriz de pesos a la hora de realizar la combinación de las estimaciones.

Por último, utilizando la FDP a priori y los pesos más apropiados:

- \( p_s(\theta) = Beta(0.01, 0.01) \)
- Utilizan \( w_s = Var(\theta|y_s) \)

El resultado se representa en la Figura 5.7.
Una vez obtenidos los histogramas para cada caso, se puede proceder a compararlos con el histograma de la distribución general. En la Figura 5.8 se representa el histograma de la estimación general.
Podemos ver que tanto los histogramas del caso 2 y 4 presentan una distribución similar en cuanto a forma comparando con el histograma de la estimación general. Sin embargo el valor del parámetro estimado se acerca mucho más al valor obtenido con la estimación general cuando se utilizan la información a priori adecuada junto con la matriz de pesos adecuada al caso gaussiano (caso 4). Como conclusión destacamos que tanto la información a priori como la matriz de pesos utilizada son importantes a la hora de realizar la estimación.

En la Figura 5.9 se comparan las funciones de densidad de la estimación general y de la estimación utilizando el método Consensus Monte Carlo empleando una matriz de pesos según el caso Gaussiano.
Con esta última representación vemos el buen funcionamiento del algoritmo, ya que ambas son similares. Por tanto, como conclusión, se destaca la necesidad de realizar una buena elección de la información a priori y de la matriz de pesos para que la estimación sea correcta.

5.3 Ejecución práctica: Segundo caso

Se aborda ahora el ejemplo anterior con dos modificaciones. En el caso anterior cada worker tenía la misma carga de trabajo. Esta vez se contará con 5 workers, y a cada uno de ellos se le asignará un número de muestras diferente para llevar a cabo la estimación local. A su vez, en el caso anterior se partía de un conjunto de datos a partir de los cuales se realizaba una estimación global y otra local. Ahora se parte de una distribución de Bernoulli cuya probabilidad de éxito es 0,01. A partir de ella se generan 710 muestras que serán repartidas entre los workers. Se asignan 100, 20, 20, 70 y 500 datos a cada worker respectivamente.

Al conocer el parámetro a estimar ($\theta = 0,01$), esta vez se podrá calcular tanto el error cuadrático medio de la estimación como la varianza.
Se comienza generando los datos que se asignarán a los workers. En las siguientes líneas de Matlab se lleva a cabo la generación de los datos:

%Generamos los datos
N = 1; %nº de intentos de la binomial
p = 0.01; %probabilidad de éxito de la binomial
Y = binornd(N,p,1,710); %Generamos las 710 observaciones binomiales
G = 1000; %nº de muestras de Monte Carlo que vamos a generar
P = zeros(G,5); %Matriz donde juntaremos las estimaciones de los trabajadores

En esta ocasión se generan 710 muestras de Bernoulli. Como anteriormente, cada worker generará $G = 1000$ muestras de Monte Carlo. La matriz P será utilizada para almacenar las muestras y operar con ellas.

En el caso anterior contábamos con 100 workers. En este caso son 5 los workers que participan en la estimación. Por tanto, la distribución a priori será la siguiente:

$$p_s(\theta) = p(\theta)^{\frac{1}{5}} = \text{Beta}\left(\frac{1}{5}, \frac{1}{5}\right)$$

(5.27)

Ya que en esta ocasión $S = 5$, es decir: los datos se dividen en 5 grupos (como se indicó anteriormente).

En las siguientes líneas de Matlab se indica cómo se produce la asignación de datos a cada worker:

%Asignamos los datos a cada trabajador
Y1 = Y(1:100);
Y2 = Y(101:120);
Y3 = Y(121:140);
Y4 = Y(141:210);
Y5 = Y(211:710);

Siguiendo el procedimiento del ejemplo anterior, y sustituyendo en la expresión 5.14 se genera la distribución a posteriori para cada worker:

%Parámetros de la prior
alfa = 1/5;
beta = 1/5;
%Para cada shard generamos G muestras de Monte Carlo.
P(1,:) = betarnd(alfa + sum(Y1),beta + N*size(Y1,2) - sum(Y1),1,G);
P(2,:) = betarnd(alfa + sum(Y2),beta + N*size(Y2,2) - sum(Y2),1,G);
P(3,:) = betarnd(alfa + sum(Y3),beta + N*size(Y3,2) - sum(Y3),1,G);
P(4,:) = betarnd(alfa + sum(Y4),beta + N*size(Y4,2) - sum(Y4),1,G);
P(5,:) = betarnd(alfa + sum(Y5),beta + N*size(Y5,2) - sum(Y5),1,G);
Seguidamente se definen los pesos que se van a utilizar:

1. **Pesos iguales.**

   \[
   w_{s1} = \frac{1}{5}
   \]  

   \(5.28\)

2. **Pesos óptimos para el caso Gaussiano:**

   \[
   w_{sg} = Var(\theta|y_s)^{-1}
   \]  

   \(5.29\)

3. **Pesos ponderados en función de las muestras disponibles en cada shard:**

   \[
   w_{s2} = \frac{\text{muestras de cada shard}}{\text{muestras totales}}
   \]  

   \(5.30\)

En las próximas líneas de Matlab se definen las matrices de pesos a utilizar en el ejemplo:

```matlab
%Generamos las diferentes matrices de pesos
M1 = 1/5.*ones(1,5);
M2 = [var(P(1,:)).^-1 var(P(2,:)).^-1 var(P(3,:)).^-1 var(P(4,:)).^-1 ...
     var(P(5,:)).^-1];
M3 = [size(Y1,2)/710 size(Y2,2)/710 size(Y3,2)/710 size(Y4,2)/710 ...
     size(Y5,2)/710];
```

donde M1, M2 y M3 se corresponden con \(w_{s1}\), \(w_{sg}\) y \(w_{s2}\) respectivamente.

Por último, sólo queda combinar las estimas entre *workers*, utilizando para ello la matriz de pesos según lo indicado en la expresión 5.23, la cual se recuerda aquí:

\[
\theta_g = \left( \sum_{s=1}^{S} w_s \right)^{-1} \left( \sum_{s=1}^{S} w_s \theta_{sg} \right)
\]

En Matlab, utilizando los datos para este caso particular:

```matlab
Pa = M1*P;
Pb = (sum(M2).^(-1) * (M2*P));
Pc = (sum(M3).^(-1) * (M3*P));
```
Donde \( Pa, Pb \) y \( Pc \) se corresponden con las matrices que almacenan las \( G = 1000 \) muestras de Monte Carlo. Hay tres debido a las tres matrices de pesos diferentes utilizadas. A partir de ellas calculamos el MSE (error cuadrático medio) y la varianza.

\[
\begin{align*}
\text{MSE}_a &= \text{mean}((p-Pa).^2); \\
\text{MSE}_b &= \text{mean}((p-Pb).^2); \\
\text{MSE}_c &= \text{mean}((p-Pc).^2); \\
\text{Var}_a &= \text{mean}((Pa-\text{mean}(Pa).^2)); \\
\text{Var}_b &= \text{mean}((Pb-\text{mean}(Pb).^2)); \\
\text{Var}_c &= \text{mean}((Pc-\text{mean}(Pc).^2));
\end{align*}
\]

En esta ocasión, al haber utilizado la misma información a priori en cada una de las tres estimaciones, podremos concentrarnos en analizar el funcionamiento de las matrices de pesos.

Representamos los valores obtenidos del error cuadrático medio para cada uno de los casos en la Figura 5.10.

![Figura 5.10. Errores cuadráticos medios.](image)
una matriz de pesos iguales estamos dando igual importancia a todos los workers. El caso más razonable sería ponderar la estimación en función de las muestras que tiene cada worker. Según este razonamiento, la matriz \( w_{sg} \) (expresión 5.30) debería dar un mejor rendimiento que la matriz \( w_{sg} \) (expresión 5.29), puesto que la primera tiene en cuenta el número de muestras de cada worker, mientras que la segunda se trata de la matriz óptima para el caso Gaussiano (siendo este un caso no Gaussiano).

En el primer ejemplo se mostraba el buen funcionamiento de la matriz óptima para el caso Gaussiano en un caso claramente no Gaussiano. Aquí los pesos Gaussianos óptimos siguen funcionando bien, pero hemos encontrado unos pesos que funcionan aún mejor.

Se muestra la varianza en la Figura 5.11.

![Figura 5.11. Varianza de las estimaciones.](image-url)
Como puede verse, los pesos Gaussianos son los mejores desde el punto de vista de minimizar la varianza. Sin embargo, dan lugar a estimas con un mayor sesgo, lo que lleva a un $MSE$ mayor.

5.4 Discusión.

Este Capítulo tenía como principal objetivo mostrar el funcionamiento del algoritmo Consensus Monte Carlo en un segundo ejemplo basado en una distribución binomial. Para ello se ha partido de un ejemplo mostrado en [Scott, 2013]. Además de seguir los pasos indicados en [Scott, 2013], se ha buscado explicar el porqué de los datos utilizados. Para ello se ha mostrado el uso de una herramienta llamada conjugate prior, que permitía obtener de forma directa una distribución a posteriori conocida a partir de los datos que se tenían. A continuación, se procedió a la división de los datos en workers. Esto supone la base del algoritmo. A partir de esta situación se derivaban dos discusiones: la información a priori asignada a cada worker y la combinación de las estimas realizadas por cada worker.

Principalmente, nuestro objetivo es poner a prueba el algoritmo Consensus Monte Carlo. Para ello se parte de una distribución a priori que poco tiene que ver con los datos, mientras que por otro lado se hace uso de otra que sí que se corresponde con ellos. De este modo, se ilustró cómo influyó la elección de la distribución a priori y se recalció la importancia de que esta elección fuese la correcta.

Seguidamente se mostró cómo funcionaba la combinación de las estimas y se valoraron los diferentes pesos a utilizar. Se demostró que los pesos óptimos para el caso gaussiano proporcionan un buen funcionamiento para casos no gaussianos.

Por último, se presentó una variación del ejemplo anterior: la cantidad de muestras dadas a cada worker era distinta. El principal objetivo de esta variación era evaluar las diferentes posibilidades en cuanto a pesos. Se observó que unos pesos que tengan en cuenta la cantidad de muestras de cada worker son los que ofrecen un mejor rendimiento.
Capítulo 6. Caso Multivariable

6.1 Introducción teórica.

En este capítulo se aborda el último caso de estimación Bayesiana considerado: la inferencia en un modelo Gaussiano multivariable. La particularidad destacable de este último caso es que nos enfrentamos a un modelo multivariable. Hasta ahora los modelos expuestos en este proyecto contaban con una única dimensión. Sin embargo, en la práctica todos los ejemplos acerca de la estimación Bayesiana con grandes conjuntos de datos siguen un modelo multidimensional.

El ejemplo descrito en este capítulo se corresponde con un modelo Gaussiano. Como en los casos anteriores, se busca estimar una serie de parámetros, y para ello contaremos con un conjunto numeroso de datos. Para realizar dicha estimación, se hace uso de herramientas presentadas de forma teórica en el Capítulo 3 de este proyecto, titulado Inferencia Bayesiana.

Como se ha dicho, se busca la estimación de unos parámetros: el vector de medias y la matriz de covarianzas en este caso. Partiendo de esta base, el objetivo principal del capítulo es comparar la estimación contando con una única máquina con el caso en el que disponemos de un algoritmo distribuido en varios núcleos de trabajo. Unos resultados satisfactorios serían aquellos en los que se apreciase que la estimación “multi máquina” es muy cercana a la estimación llevada a cabo por una sola máquina. Si estos resultados satisfactorios se alcanzasen, nos encontraríamos con un gran punto de partida para demostrar que el procesado y estimación multi-core puede proporcionar tan buenos resultados como el uni-core, y con la sabida reducción de tiempo de cálculo. Se ha comentado anteriormente que, si se cuenta con varias máquinas, el procesado de datos y la estimación de parámetros es mucho más eficaz.

Como en ocasiones anteriores, se hace uso de la herramienta Matlab. A medida que se explica la implementación del algoritmo utilizado para replicar el algoritmo, se mostrarán las líneas de código implicadas.
6.2 Modelo Gaussiano Multivariable.

Como ejemplo práctico de modelo Gaussiano multivariable, vamos a partir del ejemplo 4.2 de [Scott 2013].

En este artículo se estudia un modelo Gaussiano multivariable que viene definido por el vector de medias y la matriz de covarianzas que se indican a continuación:

$$\mu = (1, 2, 3, 4, 5) \quad (6.1)$$

$$\sum = \begin{pmatrix}
1.00 & 0.99 & 0.98 & 0.00 & -0.70 \\
0.99 & 1.00 & 0.97 & 0.00 & -0.75 \\
0.98 & 0.97 & 1.00 & 0.00 & -0.60 \\
0.00 & 0.00 & 0.00 & 1.00 & 0.00 \\
-0.70 & -0.75 & -0.60 & 0.00 & 1.00 \\
\end{pmatrix} \quad (6.2)$$

Como se puede extraer de la matriz de covarianzas, el modelo tiene tres variables con una alta correlación (indicado por los valores 0.99, 0.98 y 0.97 en la matriz de covarianzas), otra que es independiente de las demás (correlación cruzada nula) y por último una que es anticorrelada con el resto (correlaciones cruzadas de valores -0.7, -0.75 y -0.6 con las tres primeras variables, respectivamente).

A partir de las bases descritas en (6.1) y (6.2), generamos una serie de muestras con las que trabajaremos. En nuestro caso, generamos $N = 5000$ observaciones.

Recapitulando, se cuenta con una serie de observaciones que se generan a partir de un modelo Gaussiano multivariable con media y varianza descritas en (6.1) y (6.2). Matemáticamente:

$$y \sim N (\mu, \Sigma) = N (y | \mu, \Sigma) \quad (6.3)$$

Tanto $\mu$ como $\Sigma$ son desconocidos. Únicamente se han utilizado para generar los datos con los que trabajar, que es la única información (a parte del conocimiento de la forma de la FDP de $x$) que se utilizará para llevar a cabo la inferencia.

Como en anteriores ocasiones, se utiliza el teorema de Bayes para estimar el modelo:

$$p(\theta | y) = \frac{p(y | \theta) \times p(\theta)}{p(y)} \propto p(y | \theta) \times p(\theta) \quad (6.4)$$
Si particularizamos para nuestros parámetros:

\[
p(\mu, \Sigma | y) = \frac{p(y|\mu, \Sigma) \times p(\mu, \Sigma)}{p(y)} \propto p(y|\mu, \Sigma) \times p(\mu, \Sigma)
\] (6.5)

Donde:
- \( p(\mu, \Sigma) \) es la información a priori.
- \( p(y|\mu, \Sigma) \) es la verosimilitud.
- \( p(\mu, \Sigma | y) \) es la información a posteriori.
- \( p(y) \) es la probabilidad marginal.

Por otra parte, si se quiere calcular la probabilidad conjunta de dos parámetros dependientes entre sí, se puede hacer uso de la regla de la cadena. Si se tienen dos variables \( y_1 \) y \( y_2 \), la regla de la cadena establece la siguiente relación [De la Horra, 2003]:

\[
p(y_1, y_2) = p(y_1 | y_2) \times p(y_2) = p(y_2 | y_1) \times p(y_1)
\] (6.6)

Por tanto, haciendo uso de la regla de la cadena podemos reescribir la información a priori, \( p(\mu, \Sigma) \), de esta manera:

\[
p(\mu, \Sigma) = p(\mu | \Sigma) \times p(\Sigma)
\] (6.7)

A su vez, la probabilidad a posteriori, \( p(\mu, \Sigma | y) \), se reescribe de la siguiente manera utilizando la regla de la cadena:

\[
p(\mu, \Sigma | y) = p(\mu | \Sigma, y) \times p(\Sigma | y)
\] (6.8)

La expresión (6.8) será el punto de partida para calcular la probabilidad a posteriori. Una vez descrito el punto de partida llevamos a cabo un análisis de los términos que componen esa expresión.
6.2.1 FDP a posteriori de la matriz de covarianzas, \( p(\Sigma | y) \)

Utilizando el teorema de Bayes (expresión (6.4)), tenemos lo siguiente:

\[
p(\Sigma | y) = \frac{p(y|\Sigma) \times p(\Sigma)}{p(y)}
\]  
(6.9)

Según la teoría estadística Bayesiana, la distribución de probabilidad \( \text{Inverse Wishart} \) es la \text{conjugate prior} para la estimación de la matriz de covarianza de una distribución normal multivariable [Haines, 2011]. Por tanto, utilizaremos para la probabilidad a priori, \( p(\Sigma) \), una distribución \( \text{Inverse Wishart} \) caracterizada por dos parámetros: \( N_0 \) y \( \Psi' \). \( N_0 \) hace referencia a los grados de libertad. En nuestro caso \( N_0 = 5 \), ya que la dimensión (esto es, el número de variables) es 5. \( \Psi' \) es una matriz definida positiva (diremos que es definida positiva si \( y^T Ay > 0 \) para todo vector \( y \neq 0 \)). En nuestro caso, la matriz elegida es la matriz identidad, con el objetivo de que no influya en gran medida en el resultado final del ejemplo, y tomamos \( N_0 = 5 \).

Analíticamente:

\[
p(\Sigma) = W^{-1}(\Psi, N_0) = W^{-1}(I, 5)
\]  
(6.10)

Por otra parte, \( p(y|\Sigma) \) hace referencia a la distribución de los datos con \( \Sigma \) conocido. Puesto que estamos usando el modelo Gaussiano multivariable, los datos siguen una distribución Gaussiana. Y puesto que \( p(\Sigma) \) es la conjugate prior de \( p(y|\Sigma) \), la distribución a posteriori pertenecerá a la misma familia que la distribución a priori. Por tanto, la distribución a posteriori resultante será una \( \text{Inverse Wishart} \):

\[
p(\Sigma | y) = W^{-1}(\Psi + (\bar{y} - \mu_0)^T (\bar{y} - \mu_0), N + N_0)
\]  
(6.11)

Donde:

- \( \Psi \equiv I \) (matriz identidad)
- \( N \equiv \) número de observaciones
- \( \bar{y} = \frac{1}{N} \sum_{n=1}^{N} y_n \) (estima empírica de la media)
- \( N_0 \equiv \) número de grados de libertad
- \( \mu_0 \equiv \) media a priori
6.2.2 FDP a posteriori del vector de medias, $p(\mu|\Sigma, y)$

Este término se calcula de forma posterior a $p(\Sigma|y)$, ya que necesitamos haber hecho una estimación de $\Sigma$. Siguiendo el teorema de Bayes:

$$p(\mu|\Sigma = \hat{\Sigma}, y) = \frac{p(y|\mu, \Sigma = \hat{\Sigma}) \times p(\mu|\Sigma = \hat{\Sigma})}{p(y)} \quad (6.12)$$

$p(y|\mu, \Sigma = \hat{\Sigma})$ es la función de densidad de probabilidad a posteriori de las observaciones. Se trata de una distribución Gaussiana, según se ha comentado en la introducción.

En cuanto al otro término, $p(\mu|\Sigma = \hat{\Sigma})$, elegimos también una distribución Gaussiana:

$$p(\mu|\Sigma = \hat{\Sigma}) = N\left(\mu_0, \frac{1}{M} \hat{\Sigma}\right) \quad (6.13)$$

$\mu_0$ hace referencia a la media a priori. Al igual que con $\Psi$, se ha elegido un valor no informativo que no condicione las estimaciones posteriores. En nuestro caso: $\mu_0 = (0, 0, 0, 0, 0)$.

$M$ es un valor que hace referencia al llamado número de pseudo – observaciones. Este parámetro indica la importancia que le damos a la información a priori, y se puede interpretar como el número de observaciones virtuales utilizadas para obtener la FDP a priori.

De este modo, al ser ambas distribuciones Gaussanas, el producto de las mismas será a su vez una distribución Gaussiana, caracterizada por los parámetros que se indican a continuación:

$$p(\mu|\Sigma, y) = N\left(\frac{N\bar{y} + M\mu_0}{N + M}, \frac{1}{N + M} \hat{\Sigma}\right) \quad (6.14)$$

6.2.3 Objetivo

Una vez introducidas las herramientas y expresiones que se van a utilizar, establecemos claramente el objetivo del ejemplo: se persigue obtener una estimación lo más precisa posible de la matriz de covarianzas, representando la función de densidad de probabilidad de dicha estimación. Para representar dicha función de densidad de probabilidad se seguirá la metodología de Monte Carlo. Es decir, se generarán una serie de muestras para, a partir de ellas, realizar una estimación aproximada.
Se recuerda que, cuanto mayor sea el número de muestras disponibles mejor aproximaremos el valor real de los parámetros a estimar. Analíticamente:

\[
\lim_{T \to \infty} \frac{1}{T} \sum_{t=1}^{T} \hat{\theta}_t = \hat{\theta}_{\text{MMSE}} = \int_{-\infty}^{\infty} \theta \ p(\theta | y) d\theta
\] (6.15)

En la siguiente sección se realizará una estimación global y seguidamente se hará una estimación local, para finalmente comparar las dos estimaciones. Por último, se implementará un algoritmo de corrección del sesgo (algoritmo Jackknife) y se observará su eficiencia.

### 6.3 Implementación práctica del ejemplo.

#### 6.3.1 Generación de los datos

En primer lugar, se definen los parámetros que se van a utilizar para implementar el ejemplo práctico. Estos han sido definidos previamente en la parte teórica de este Capítulo. En las siguientes líneas de código Matlab quedan definidos:

```matlab
%Primero generamos los datos
T = 100; %Número de iteraciones
N = 5000; %Número de observaciones
N0 = 5;
mu0 = [0 0 0 0 0];
M = 10;
mu = [1 2 3 4 5];
cov = [1 0.99 0.98 0 -0.7; 0.99 1 0.97 0 -0.75; 0.98 0.97 1 0 -0.6;...
    0 0 0 1 0; -0.7 -0.75 -0.6 0 1];
```

Observamos que el número de muestras generadas por el algoritmo de Monte Carlo será igual a 100, ya que este coincide con el número de iteraciones del algoritmo. A su vez, como se ha indicado en la introducción, el número de muestras (datos) de las que partimos es igual a 5000. Por último, cabe destacar el valor que se le ha dado al número de pseudo-observaciones (M = 10). Este valor indica que la importancia de la información a priori es pequeña (nótese que \( \frac{M}{N} = \frac{1}{500} \)). Como objetivo posterior se tiene el analizar la influencia directa de este valor en el algoritmo, ya que en [Scott 2013] no se especifica el valor tomado.

Una vez expuestos los parámetros base de nuestro algoritmo, se generan las 5000 muestras que se corresponderán con nuestros datos. Estas muestras siguen una distribución Gaussiana multivariable. En Matlab:

```matlab
y = mvnrnd(mu,cov,N);
```
Una vez se dispone de los datos, es posible comenzar la estima de los parámetros que desconocemos. Es decir: la matriz de covarianzas ($\Sigma$) y el vector de medias ($\mu$).

### 6.3.2 Estima global

Previamente se calculan los parámetros necesarios para la estima: $\bar{x}$ y $S$.

```matlab
%Primero definimos los parámetros de la posterior
psi = eye(5);
ym = (mean(y)'*(ones(1,N)))';
S = (1/N)*((y-ym)'*(y-ym));
covt = zeros(size(cov,1),size(cov,2),T);
mut = zeros(size(mu,1),size(mu,2),T);
```

En las dos últimas líneas se definen las matrices donde se guardarán las muestras generadas mediante el método de Monte Carlo.

Seguidamente, se muestra el bucle donde se lleva a cabo la generación de muestras de $\Sigma$ y $\mu$ de acuerdo con sus respectivas FDPs a posteriori. Nótese que se utilizan las expresiones (6.11) y (6.14) respectivamente:

```matlab
for t=1:T
    covt(:,:,t) = iwishrnd(psi+N*S+(N*M/(N+M))*(mean(y)-mu0)'*(mean(y)-mu0),N+N0);
    mut(:,:,t) = mvnrnd(((N*mean(y)+M*mu0)/N+M),covt(:,:,t)/(N+M));
end
```

Finalmente se promedian las T muestras generadas para obtener las estimas globales:

```matlab
%Estimaciones globales
cov_estg = mean(covt,3);
mu_estg = mean(mut,3);
```

### 6.3.3 Estimación local

Una vez se ha realizado la estimación global, se prosigue con la estimación local (dividiendo las observaciones en shards). Como se vio en capítulos anteriores, Capítulo 3 y Capítulo 5, cada uno de estos shards se asignará a un worker. Cada worker hará su propia estimación a partir de sus datos. Y finalmente se combinarán dichas estimas. Para más información, véase el Algoritmo Consensus Monte Carlo (Algoritmo 1 del Capítulo 5).

Se ha elegido, inicialmente, un número de workers igual a 100. A cada uno de ellos se le asignará un shard de 50 observaciones. En la parte final del capítulo se compararán los resultados teniendo 100 workers frente a tener 50.

En Matlab:

```matlab
%Dividimos el trabajo en workers
W = 100; %Número total de workers
Nw = N/W; %Número de observaciones por worker
```
Al igual que se hizo antes para la estima global, se siguen los mismos pasos para la estimación local de cada worker, generando muestras de sus respectivas FDPs a posteriori locales (que son distintas de la FDP a posteriori global, ya que cada shard sólo dispone de parte de los datos) mediante el método de Monte Carlo.

%Creamos las matrices donde guardaremos los datos
covtw = zeros(size(cov,1),size(cov,2),T,W);
mutw = zeros(W,size(mu,2),T);

for w=1:W
    for t=1:T
        yw = y(Nw*(w-1)+1:Nw*w,:);
        ymw = (mean(yw) +* (ones(1,Nw))))';
        covtw(:,:,t,w) = iwishrnd(psi+Nw+...
            (mean(yw)-mu0)'*(mean(yw)-mu0),Nw+N0);
        mutw(w,:,t) = mvnrnd(((Nw*mean(yw)... +M*mu0)/Nw+M),covtw(:,:,t,w)/(Nw+M));
    end
end

Destacamos que se ha particularizado el parámetro \( \bar{y} \) para el caso de \( N_w \) muestras.

Seguidamente, es necesario combinar las muestras. Para ello se utilizan las matrices de pesos definidas en el Algoritmo 1 del Capítulo 5. A modo de recordatorio:

\[
\theta_g = \left( \sum_{s=1}^{S} w_s \right)^{-1} \left( \sum_{s=1}^{S} w_s \theta_{sg} \right)
\]

Donde:

\[
w_s = Var(\theta | y_s)^{-1}
\]

En las siguientes líneas de código se refleja la definición de las matrices y la combinación de las estimas de los workers:

%Definimos las matrices de pesos
MA_cov = reshape(var(covtw,0,3).^-1,[size(cov,1) size(cov,2) W]);
MA_mu = var(mutw,0,3).^-1;

cov_estw = zeros(size(cov,1),size(cov,2),W,T);
mu_estw = zeros(size(mu,1),size(mu,2),T);

for t=1:T
    for i=1:size(mu,2)
        mu_estw(:,:,i,t) = sum(MA_mu(:,:,i))^-1*MA_mu(:,:,i)'*mutw(:,:,i,t);
    end
    for w=1:W
        cov_estw(:,:,w,t) = sum(MA_cov(:,:,w)*covtw(:,:,t,w));
    end
end
Por último, solo queda promediar las $T$ muestras generadas para obtener la estima deseada. De esta manera, obtenemos la estimación de los parámetros $\Sigma$ y $\mu$:

\[
\text{cov\_est} = \text{mean}(\text{reshape}(\text{sum(cov\_estw,3),[size(cov,1) size(cov,2) T]}),3);
\]
\[
\text{mu\_est} = \text{mean}(\text{mu\_estw},3);
\]

### 6.4 Resultados.

Una vez se han implementado las líneas de código necesarias para llevar a cabo la estimación, se muestran algunos resultados. Se dibujará la función de densidad de probabilidad de cada componente de la matriz de covarianzas ($\Sigma_{ij}$). En un principio se cuenta con 100 workers, y a cada uno de ellos se le asigna un shard de 50 muestras. En la Figura 6.1 se representan gráficamente las funciones de densidad de probabilidad, tanto para la estima global como para la estima local.

![Figura 6.1. Estimación de $\Sigma_{ij}$. Estimación local frente a estimación global usando 100 workers. La línea de color negro representa la estima global y la línea roja discontinua la estimación local.](image)

En este caso podemos tomar como “ground truth” a la estimación global, dado el elevado número de muestras usado.

Téngase en cuenta que el número de iteraciones (número de muestras generadas por el algoritmo de Monte Carlo) es igual a 100. Se considera que este valor es aceptable y equilibra la balanza entre la precisión de la estimación y la velocidad de computación.
Si ahora se eligen 50 workers con 100 muestras asignadas a cada uno, se obtienen los resultados representados en la Figura 6.2

**Figura 6.2.** Estimación de $\Sigma_{ij}$. Estimación local frente a estimación global usando 50 workers. La línea negra corresponde a la estimación global y la línea roja discontinua a la estimación local.

Vemos que ambos resultados tienen un pequeño sesgo. El sesgo se define como la desviación de la esperanza matemática en la estimación de un parámetro frente a su valor real. El sesgo cuando se tienen 50 workers es ligeramente inferior. Esto es razonable, puesto que el sesgo suele ser inversamente proporcional al número de datos, de modo que el sesgo será mayor cuantas menos muestras sean asignadas a cada worker.

Aparte de aumentar el número de muestras, una manera de reducir el sesgo consiste en utilizar el estimador Jackknife. En este caso particular el sesgo es muy bajo, pero aun así explicamos este estimador a continuación y comprobamos su eficacia.
6.5 Reducción del sesgo: Estimador Jackknife.

Este estimador se aplicará sobre cada worker. Aclaremos el punto de partida: assumimos que se tienen $N$ muestras en cada shard ($y_1, \ldots, y_N$). Recordamos que a cada worker se le asigna un shard. El parámetro que queremos estimar se va a identificar de forma genérica como $\hat{\theta}$. A continuación se detallan los pasos que sigue el algoritmo [Miller, 1974]:

1. Se dividen las $N$ observaciones en $G$ grupos de tamaño $H = \frac{N}{G}$.
2. Para $i = 1, \ldots, G$: Se obtiene $\hat{\theta}_{yi}$ utilizando todos los grupos de datos excepto el $i$-ésimo.
3. Se obtiene el estimador final como:

   $$
   \begin{align*}
   \tilde{\theta}_i &= G\hat{\theta} - (G - 1)\hat{\theta}_{yi} \\
   \hat{\theta} &= \frac{1}{G} \sum_{i=1}^{G} \tilde{\theta}_i \\
   \Rightarrow \hat{\theta} &= G\hat{\theta} - \frac{G - 1}{G} \sum_{i=1}^{G} \hat{\theta}_{yi}
   \end{align*}
   $$

   Donde $\hat{\theta}$ es el estimador local que utiliza todos los datos del worker.


En el Consensus Monte Carlo se hace un subsample del 20%. Esto significa que se fija el número de grupos, $G$, a 5. Por tanto, el número de muestras por grupo, $H$, será el número de muestras que se le asigna a cada worker dividido por el número de grupos. Analíticamente:

$$
H = \frac{N_w}{5} = \frac{N_w}{5}
$$

(6.18)

En primer lugar se definen los parámetros que hemos comentado y se implementan las matrices donde se guardarán los datos:

%Implementación del Jackknife
%Creamos las matrices donde guardaremos los datos
G = 5; %número de grupos
H = Nw/G; %número de muestras de los grupos
covtw_j = zeros(size(cov,1),size(cov,2),T,W,G);
mutfw_j = zeros(W,size(mu,2),T,G);

Comparando este caso particular con el algoritmo genérico, vemos que $covtw_j$ hace referencia al parámetro $\hat{\theta}_{yi}$. 

Proyecto Fin de Grado 103
Nótese que la estimación Jackknife se lleva a cabo para las $T$ muestras de Monte Carlo. En las siguientes líneas se indica el código utilizado para la implementación del algoritmo:

```matlab
for w=1:W
    ytemp = y((w-1)*Nw+1:w*Nw,:);
    for i=1:G
        y_j = ytemp(1:H*G,:); %Cojo H*G muestras
        y_j((i-1)*H+1:i*H,:) = []; %Quito un grupo por cada iteración
        ymw_j = (mean(y_j,1)'*ones(1,H*G-H))';
        for t=1:T
            covtw_j(:,:,t,w,i) = iwishrnd(psi+(mean(y_j,1)-mu0)'*...
                (mean(y_j,1)-mu0),(H*G-H)+N0);
            mutw_j(w,:,t,i) = mvnrnd(((H*G-H)*mean(x_j)... +M*mu0)/((H*G-H)+M)),covtw_j(:,:,t,w,i)/((H*G-H)+M));
        end
    end
end

cov_j = mean(G*reshape(mean(covtw,4),[size(cov,1) size(cov,2) T]) -...
        ((G-1)/G)*sum(reshape(mean(covtw_j,4),[size(cov,1) size(cov,2) T G]),4),3);
```

Una vez implementado el algoritmo se observan los resultados. Veremos la efectividad para el primer caso (en el que contábamos con 100 *workers*). En la Figura 6.3 se puede ver el resultado.

![Figura 6.3. Efectividad del algoritmo Jackknife contando con 100 workers. En negro la estimación global, en línea roja discontinua la estimación local y en azul la estimación corregida por Jackknife.](image)

En color azul vemos la distribución resultante habiendo utilizado el algoritmo Jackknife. Por lo tanto vemos que para este caso la efectividad del algoritmo es muy alta: en general se consigue disminuir el sesgo, ya que
la distribución de las estímas locales se desplaza hacia la del estimador global. Se recuerda que en color negro se muestra la distribución de la estimación global y con la línea naranja discontinua se indica la estimación local. Apenas se puede distinguir entre el color negro y el azul, esto supone un indicador de la eficacia del estimador ya que el sesgo se ha corregido prácticamente en su totalidad.

Veamos ahora el resultado cuando se utilizan 50 workers. Vemos estos resultados en la Figura 6.4.

Figura 6.4. Efectividad del algoritmo Jackknife contando con 50 workers. En negro la estimación global, en línea roja discontinua la estimación local y en azul la estimación corregida por Jackknife.

El sesgo es ligeramente inferior que para el caso de 100 workers. Aun así vemos que el algoritmo Jackknife sigue realizando correctamente su función y a simple vista el sesgo ha desaparecido completamente.

Si disminuimos el número de workers drásticamente (por ejemplo, contando con 5 workers) presumiblemente tendremos un sesgo muy bajo. Visualizamos ahora el comportamiento del algoritmo para este caso particular (Figura 6.5).
Figura 6.5. Efectividad del algoritmo Jackknife (5 workers). En negro la estimación global, en línea roja discontinua la estimación local y en azul la estimación corregida por Jackknife.

En la Figura 6.6 se muestra la función de densidad de un único componente de la matriz de covarianzas para tener una vista con más zoom de la comparación entre la estima global y la estima local. Se puede apreciar que el sesgo es muy bajo.

Figura 6.6. Comparación de la estima global y la estima local de la función de densidad de un coeficiente de la matriz de covarianzas. En línea roja discontinua la estima local y en negro la estima global.
Podemos apreciar en la Figura 6.6 que efectivamente el sesgo es nulo a simple vista, pero lo más destacable es que el comportamiento del algoritmo Jackknife es óptimo: el sesgo ha desaparecido por completo. Por tanto, podemos decir que, para este caso particular, el funcionamiento del algoritmo Jackknife no nos va a llevar a resultados incorrectos. Es decir: el algoritmo Jackknife nunca va a afectar negativamente a la estimación para este caso concreto.

Por último, probemos otro caso extremo: dividimos el trabajo en 1000 workers. Disponer de un número tan alto de workers supone un alto coste computacional. A cada uno de ellos le corresponderán 5 muestras. En principio esto debe suponer un sesgo muy alto. Se aplicará también la corrección Jackknife. En la Figura 6.7 se muestran los resultados.

Efectivamente vemos un sesgo muy alto, pero de nuevo el rendimiento del algoritmo Jackknife es muy bueno y permite una reducción notoria del sesgo.
6.6 Discusión.

En este capítulo se ha considerado un modelo Gaussiano multivariable, replicando uno de los ejemplos propuestos en [Scott, 2013]. La particularidad de este ejemplo es que nos adentrábamos por primera vez en el ámbito multivariable. Contando con esta particularidad, se utilizaron las herramientas matemáticas usadas en anteriores capítulos, para generar las correspondientes estimaciones a posteriori. La mayor dificultad a la hora de trabajar en un espacio multivariable fue la implementación del código de los algoritmos y el manejo de las matrices que ello conllevaba.

En este capítulo se utilizó de nuevo el algoritmo Consensus Monte Carlo para comparar las estimaciones globales con las estimaciones locales. A la hora de realizar las estimaciones, se llevaron a cabo 100 iteraciones, lo que implicaba generar 100 muestras de Monte Carlo. Se consideró que este era un valor apropiado que oscilaba entre dos máximas: un coste computacional aceptable y una estimación precisa.

Se vio que la división del trabajo de estimación en workers trajo asociado el problema del sesgo. Se comprobó que este era mayor cuanto más numerosos eran los workers (esto es, cuanto menor era el número de muestras por worker). Para solucionar este inconveniente se presentó el algoritmo Jackknife. Este algoritmo consiste en dividir el número de muestras asociadas a cada worker en grupos y aplicar una estimación sobre esos grupos. Como resultado, vimos que la efectividad de este algoritmo es óptima: en general consigue reducir el sesgo de forma notable y apenas da resultados negativos.

En relación a esta última afirmación se presentó un caso extremo: se dividió el trabajo en 1000 workers. El sesgo en este caso fue notablemente mayor, pero vimos que, en líneas generales, el resultado de aplicar el algoritmo Jackknife también fue satisfactorio.

Sin embargo, esto no quiere decir que el algoritmo Jackknife tenga un buen rendimiento en otro tipo de distribuciones. Uno de los próximos retos podría ser el de evaluar su rendimiento en otro tipo de distribuciones.

Como conclusión, hay que destacar que tanto la estimación multivariable y su corrección de sesgo han sido altamente satisfactorias siendo el resultado incluso mejor que el que se muestra en [Scott, 2013].
Capítulo 7. Conclusiones y líneas futuras

En este proyecto se ha considerado el problema del análisis de grandes conjuntos de datos. Para abordar el análisis de dichos conjuntos de datos se han introducido las herramientas matemáticas que nos permitan entender tal análisis. La principal herramienta utilizada es la inferencia Bayesiana y los métodos de Monte Carlo. Se han elaborado tres casos prácticos: el primer caso abordaba la estimación de la medida de una resistencia. El segundo caso trataba las distribuciones binomiales y presentaba la división del trabajo en diferentes núcleos computacionales, que en el presente proyecto se conocen como workers. Por último, se abordó un caso multivariable, y al igual que en el segundo caso, se realizó una estima global y una estima local. Las conclusiones generales obtenidas han sido las siguientes:

- El hecho que hace que el Big Data sea un concepto tan importante hoy día radica en el análisis que se hace sobre estos conjuntos de datos y en la extracción de valor de los mismos. Sin este análisis el concepto de Big Data carece de valor.
- La metodología Bayesiana nos permite combinar información a priori e información contenida en los datos de manera que, mediante la combinación de estas dos informaciones, podamos obtener información a posteriori y así poder tomar decisiones basadas en ella. Hay que destacar que esta metodología Bayesiana nos da el grado de certidumbre, es decir, nos indica cómo de seguros podemos estar ante la toma de una decisión.
- Los métodos de Monte Carlo son muy importantes en la estimación Bayesiana. Son la solución para poder realizar estimaciones sobre conjuntos de datos en los que no es posible encontrar estimadores cerrados.
- Los métodos de Monte Carlo distribuidos suponen una solución para realizar las estimaciones a mayor velocidad, ya que permiten trabajar en paralelo.
- Es necesario llegar a un equilibrio entre el rendimiento y la velocidad. Cuantas más máquinas estén trabajando en paralelo menor será el rendimiento, pero mayor la velocidad. El rendimiento óptimo es el que se obtiene con una sola máquina, pero trabajar de esta manera supone una velocidad muy baja.
- Las estimaciones mediante las herramientas anteriores se pueden extender al caso multidimensional. Cabe destacar que los casos multidimensionales son los que se acercan más a los casos reales, de ahí su importancia.
- El sesgo puede ser un problema a la hora de realizar estimaciones. Sin embargo, hemos visto que el algoritmo Jackknife proporciona muy buenos resultados a la hora de corregirlo, al menos en el caso Gaussiano multivariable.

A continuación indicamos alguna de las líneas futuras a tratar:

- La continuación del desarrollo de los algoritmos aquí descritos aplicándolos a casos más complejos (distribuciones con más dimensiones, menos convencionales, etc.).
- Utilización de datos reales a la hora de realizar las estimaciones (recordemos que en este proyecto se han utilizado datos sintéticos).
- Desarrollo de una aplicación real que haga uso de las ideas y conceptos expuestos en este proyecto.
- Análisis de métodos de Monte Carlo que permitan la comunicación entre agentes.
- Uso de otros métodos de Monte Carlo. Por ejemplo, el muestreador de Gibbs para problemas multidimensionales o técnicas de muestreo enfatizado (“importance sampling”) [Neal, 2001].
Referencias


Índice de Figuras

Figura 2.5 – [LLC, 2016] – La funcionalidad de búsqueda en la aplicación iPlant.
Figura 2.6 – [Joe, 2016] – Características de la Ciudad Inteligente.
Figura 2.7 – [ONU, 2012] – Correlación entre el número de tweets y el precio del arroz.
Figura 2.8 – [Ememi, 2011] – Control del ritmo cardiaco por la aplicación HealthVault
Figura 3.2 – Ejemplo de sesgo. En negro la estimación global y en línea roja discontinua la estimación local.
Figura 4.1 – [UPV] - Influencia de la temperatura en la resistencia eléctrica del cobre.
Figura 4.2 – Estimador R1.
Figura 4.3 – Estimador R2.
Figura 4.4 – Estimador R2b.
Figura 4.5 – Estimador R3.
Figura 4.6 – Estimador R4.
Figura 4.7 – Comparación de los errores de cada algoritmo.
Figura 4.8 – Comparaciones entre errores cuadráticos medios en M-H con y sin tiempo de “burn-in”.
Figura 4.9 – Error cuadrático medio en función del tiempo de “burn-in”.
Figura 4.10 – Comparación entre el sesgo de los estimadores.
Figura 4.11 – Varianza de los estimadores.
Figura 5.1 – Distribución Beta (α=2 y β=1000).
Figura 5.2 – Distribución Beta (α=1 y β=1).
Figura 5.3 – Distribución Beta (α=0.01 y β=0.01).
Figura 5.4 – Histograma, caso 1.
Figura 5.5 – Histograma, caso 2.
Figura 5.6 – Histograma, caso 3.
Figura 5.7 – Histograma, caso 4.
Figura 5.8 – Histograma estimación general.
Figura 5.9 – Comparación entre las funciones de densidad.

Figura 5.10 – Errores cuadráticos medios.

Figura 5.11 – Varianza de las estimaciones.

Figura 6.1 – Estimación de $\Sigma_{ij}$. Estimación local frente a estimación global usando 100 workers. La línea de color negro representa la estima global y la línea roja discontinua la estimación local.

Figura 6.2 – Estimación de $\Sigma_{ij}$. Estimación local frente a estimación global usando 50 workers. La línea nera corresponde a la estimación global y la línea roja discontinua a la estimación local.

Figura 6.3 – Efectividad del algoritmo Jackknife contando con 100 workers. En negro la estimación global, en línea roja discontinua la estimación local y en azul la estimación corregida por Jackknife.

Figura 6.4 – Efectividad del algoritmo Jackknife contando con 50 workers. En negro la estimación global, en línea roja discontinua la estimación local y en azul la estimación corregida por Jackknife.

Figura 6.5 – Efectividad del algoritmo Jackknife (5 workers). En negro la estimación global, en línea roja discontinua la estimación local y en azul la estimación corregida por Jackknife.

Figura 6.6 – Comparación de la estima global y la estima local de la función de densidad de un coeficiente de la matriz de covarianzas. En línea roja discontinua la estima local y en negro la estima global.

Figura 6.7 – Efectividad del algoritmo Jackknife (1000 workers). En negro la estimación global, en línea roja discontinua la estimación local y en azul la estimación corregida por Jackknife.
Apéndice A. Presupuesto

Para la elaboración del presupuesto necesario para llevar a cabo el presente proyecto se han tenido en cuenta tanto los recursos humanos como los medios materiales.

A.1 Coste de los medios materiales.

En la Tabla A.1 se indican los recursos materiales necesarios para la correcta realización del proyecto. Además de estos materiales se necesita disponer de un lugar de trabajo debidamente acondicionado para que el trabajador desarrolle el proyecto con la mayor comodidad posible. Esto supone una correcta iluminación, temperatura adecuada, limpieza, mesas, sillas y tomas de corriente y de Internet. El alquiler de dicho emplazamiento se estima en 600 euros por mes. Se estima que el tiempo necesario para realizar el proyecto son 6 meses contando con una jornada laboral de 40 horas semanales. Por tanto, el coste del lugar de trabajo supondrá un total de 3.600 euros.

<table>
<thead>
<tr>
<th>Concepto</th>
<th>Coste por unidad</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ordenador personal</td>
<td>1.000 euros</td>
</tr>
<tr>
<td>Licencia de Matlab</td>
<td>2.300 euros</td>
</tr>
<tr>
<td>Microsoft Office 2016</td>
<td>150 euros</td>
</tr>
<tr>
<td>Impresiones y fotocopias</td>
<td>100 euros</td>
</tr>
<tr>
<td>Material de oficina</td>
<td>100 euros</td>
</tr>
<tr>
<td>Encuadernación e impresión final</td>
<td>200 euros</td>
</tr>
<tr>
<td><strong>TOTAL</strong></td>
<td><strong>3.850 euros</strong></td>
</tr>
</tbody>
</table>

Sumando el coste de los materiales al coste del lugar de trabajo se obtiene un total de 7.450 euros necesarios para cubrir los costes materiales del proyecto.

A.2 Coste de los recursos humanos.

Se ha estimado que el sueldo medio del desarrollador del proyecto sea de 18.500 euros brutos al año divididos en 14 pagas. Como el proyecto se va a desarrollar durante 6 meses, se tendrá un sueldo de 1.270 euros netos al mes. Lo que supone un total de 7.620 euros, correspondientes a los 6 meses necesarios para llevar a cabo el proyecto.

Será necesaria llevar a cabo una supervisión mensual del trabajo, la cual será efectuada por el director del proyecto. La compensación económica correspondiente a esta dirección será de un 10% del salario neto del netos por la duración total del proyecto.

En resumen, el presupuesto reservado a los recursos humanos asciende a un total de 8.820 euros.
A.3 Coste total.

Sumando el presupuesto necesario para los medios materiales y el destinado a los recursos humanos, el presupuesto total del proyecto supone un total de 12.670 euros.