Notas de clase

APRENDIZAJE DE MÁQUINAS

Esta versión: 17 de junio de 2020

Última versión: `github.com/GAMES-UChile/Curso-Aprendizaje-de-Maquinas`

Felipe Tobar
Centro de Modelamiento Matemático
Universidad de Chile

`ftobar@dim.uchile.cl`
`www.dim.uchile.cl/~ftobar`

Prefacio

Este apunte es una versión extendida y detallada de las notas de clase utilizadas en el curso MA5204: Aprendizaje de Máquinas (ex MA5203) dictado en el Departamento de Ingeniería Matemática, Universidad de Chile. El objetivo principal de este apunte es presentar material autocontenido y original de las temáticas vistas en el curso tanto para apoyar su realización como para estudio personal de quien lo requiera. Debido a que los contenidos del curso van variando año a año, el apunte está en constante modificación, por esta razón hay secciones de este documento que pueden estar incompletas en cuanto a formato, figuras o contenidos. Sin embargo, creo que el hacer disponible este apunte en desarrollo puede ser un aporte para los alumnos del curso MA5204 como a la comunidad en general.

El desarrollo de este apunte solo ha sido posible gracias a la contribución de varios integrantes del cuerpo académico de los cursos MA5203: Aprendizaje de Máquinas Probabilístico (2016-2018), MA5309: Aprendizaje de Máquinas Avanzado (2016, 2018) y MA5204: Aprendizaje de Máquinas (2019-2020). Me gustaría reconocer la indispensable contribución de ayudantes y participantes de estos cursos, tanto en el desarrollo del curso mismo, ideas de tareas y clases auxiliares, producción de figuras, ejemplos, y mucho más: Alejandro Cuevas, Claire Delplancke, Gonzalo Ríos, Alejandro Veragua, Nicolás Aramayo, Lerko Araya, Mauricio Campos, Cristóbal Silva y Cristóbal Valenzuela. Sin la participación de todos ellos, el éxito de los cursos mencionados y la composición de este apunte no habrían sido posible.

Felipe Tobar
17 de junio de 2020
Santiago, Chile

Índice

1. Introducción

El aprendizaje de máquinas (AM) es una disciplina que reúne elementos de ciencias de la computación, optimización, estadística, probabilidades y ciencias cognitivas para construir el motor de aprendizaje dentro de la Inteligencia Artificial. Definido por Arthur Samuel en 1950, el AM es la disciplina que da a las máquinas la habilidad de aprender sin ser explícitamente programadas. Si bien existen enfoques al AM inspirados en sistemas biológicos, esta no es la única forma de construir métodos de aprendizaje: una analogía se puede identificar en los primeros intentos por construir máquinas voladoras, en donde se pretendía replicar el movimiento de las alas de un pájaro, sin embargo, estos intentos no fueron exitosos y no fue sino hasta la invención del primer avión, el cual no mueve sus alas, que el hombre logró construir la primera máquina voladora. Esto sugiere que el paradigma biológico no es exclusivo al momento de construir máquinas inteligentes que utilicen observaciones de su entorno para extraer información útil y generar predicciones.

AM es una disciplina joven que ha experimentado un vertiginoso crecimiento en las últimas décadas, esto ha sido posible en gran parte gracias a los recientes avances computacionales y la cantidad de datos que permite entrenar modelos complejos. El uso masivo de técnicas AM se ha visto reflejado en distintas áreas que incluyen visión computacional, clasificación de secuencias de ADN, marketing, detección de fraude, diagnósticos médicos, análisis financiero y traducción de texto por nombrar algunas. Adicionalmente, si bien el objetivo de AM es desarrollar algoritmos de aprendizaje prescindiendo en gran medida de la intervención humana, otra razón del éxito del AM es su facilidad para acoplarse con otras disciplinas aplicadas, en particular al área que hoy conocemos como *Data Science*.

1.1. Orígenes: Inteligencia Artificial

El hecho de que nuestras habilidades cognitivas nos diferencien fuertemente del resto de las especies del reino animal es la principal prazón del dominio del *Homo sapiens* sobre el planeta: la inteligencia humana superior a la del resto de los animales nos permite adaptarnos a diferentes situaciones ambientales y sociológicas de forma rápida y sin la necesidad de un cambio evolutivo. Por ejemplo, para migrar desde África al norte de Europa y a Oceanía, el *Homo sapiens* no necesitó adaptarse biológicamente a climas distintos, sino que manipuló herramientas y materiales para producir vestimenta adecuada y embarcaciones. Otra característica única del *Homo sapiens* es su habilidad de creer en un imaginario colectivo que permite construir organizaciones con un gran número de individuos, lo cual es exclusivo a nuestra especie y consecuencia de nuestra inteligencia (Harari, 2015).

Nuestro interés en entender la inteligencia puede ser identificado desde los comienzos de la Filosofía y Sicología, disciplinas que se han dedicado al estudio de las forma en que entendemos, recordamos, razonamos y aprendemos. La inteligencia artificial (IA) es una disciplina mucho más reciente que las anteriores y va un paso más allá de la mera comprensión de la inteligencia, pues apunta a replicarla e incluso mejorarla (Bostrom, 2014). Hay varias definiciones de IA dependiendo de si (i) adoptamos un punto de vista cognitivo o conductual, o bien si (ii) identificamos las acciones inteligentes como humanas u objetivamente racionales. Una de estas posiciones es la de Alan Turing, el que mediante el juego de la imitación (Turing, 1950), sentencia que una máquina es inteligente si es capaz de desarrollar tareas cognitivas a un nivel "humano" suficiente para engañar a un interrogador (también humano). En este contexto para que una máquina sea inteligente, o equivalentemente, apruebe el test de Turing, es necesario que posea (Russell y Norvig, 2009):

- **representación del conocimiento** para guardar información recibida antes y durante la interrogación,
- **procesamiento de lenguaje natural** para comunicarse con seres humanos, en particular, el interrogador,

- **razonamiento automático** para usar la información guardada y formular respuestas y conclusiones, y
- **aprendizaje de máquinas** para adaptarse a nuevas circunstancias y descubrir patrones.

El test de Turing sigue siendo un tópico de investigación en Filosofía hasta el día de hoy, sin embargo, los avances actuales de la inteligencia artificial no están necesariamente enfocados en diseñar máquinas para aprobar dicho test. Si bien los inicios de la IA están en la Filosofía, actualmente los avances en IA se enfocan en ambientes controlados, donde apuntan a desarrollar metodologías para extraer información de bases de datos en situaciones que el operador humano tiene limitaciones de velocidad, capacidad o eficiencia. Estos esfuerzos de la IA no necesariamente replican el actuar del humano, en particular, el componente de aprendizaje de máquinas en la lista anterior ha jugado un papel fundamental en esta nueva etapa, en donde su aporte en distintas áreas de aplicación es cada vez más evidente.

1.2. Breve historia del aprendizaje de máquinas

En base a las definiciones de la inteligencia de máquinas asentadas por Turing en 1950, las primeras redes neuronales artificiales comenzaron a emerger en los trabajos seminales de (Minsky, 1952) que programó el primer simulador de una red neuronal, (Farley y Clark, 1954) que implementaron un algoritmo de prueba y error para el aprendizaje, y (Rosenblatt, 1958) que propuso el Perceptrón. En los años siguientes la investigación en redes neuronales se vio afectada por las limitaciones de dichas estructuras expuestas en (Minsky y Papert, 1969) dando origen a lo que es conocido como el primer invierno de la inteligencia artificial, en donde el Profesor Sir James Lighthill expuso frente al parlamento inglés que la inteligencia artificial se fijaba objetivos no realistas y solo servía para escenarios básicos (Lighthill, 1973). Esta desconfianza en los alcances de la IA ralentizó su desarrollo, sin embargo, los *conexionistas*[1] seguirían investigando sobre formas de diseñar y entrenar redes neuronales. Específicamente, los resultados de (Werbos, 1974) que culminarían en el algoritmo de *backpropagation* propuesto por (Rumelhart, Hinton, y Williams, 1986) y los avances de (Hopfield, 1982) en redes neuronales recurrentes permitirían terminar con el primer invierno de la inteligencia artificial.

Durante el receso del conexionismo proliferaron los sistemas basados en reglas, en particular, los sistemas experto compuestos por una serie de reglas condicionales "si-entonces" (if-then), las cuales replican el comportamiento de un humano experto, estos métodos se convirtieron en la primera herramienta exitosa de la IA en aplicaciones reales. Sin embargo, los sistemas experto no aprenden por sí solos, en el sentido de que las reglas "si-entonces" deben ser explícitamente programadas por un humano. Este enfoque tiene un costo considerable dependiendo de la complejidad del problema en cuestión, por lo que hacia el comienzo de la década de los 90s los sistemas experto colapsaron debido a que la cantidad de información disponible aumentaba y dicho enfoque no es "escalable". Un sistema basado en reglas que aún se utiliza son los llamado árboles de decisión (Breiman, Friedman, Olshen, y Stone, 1984), los cuales difieren de los sistemas experto en que las reglas no son definidas por un humano sino que descubiertas en base a la elección de variables que mejor segmentan los datos de forma supervisada.

Las redes neuronales vieron un resurgimiento en la década de los 80s con el método de *backpropagation*, el cual permitía entrenar redes neuronales de más de dos capas usando la regla de la cadena. Esto permitió finalmente validar la premisa conexionista en tareas complejas, específicamente en reconocimiento de caracteres usando redes convolucionales como el Neocognitron (Fukushima, 1980) y LeNet-5 (LeCun y cols., 1989), y reconocimiento de voz usando redes neuronales recurrentes (Hopfield, 1982). Posterior a esta validación experimental, vino una segunda caída del conexionismo hacia fines de los 80s, ésta se debió a la inexistencia de una clara teoría que explicara el desempeño de las redes neuronales, su tendencia a sobreajustar y su elevado costo computacional. A principios de los 90s, y basado en la teoría

[1] Es decir, los partidarios del *conexionismo*, enfoque en el que los fenómenos mentales pueden ser explicados mediante la conexión de elementos más simples, e.g., *neuronas*. En el contexto de IA o AM, nos referimos coloquialmente a los *conexionistas* como los partidarios de las redes neuronales.

del aprendizaje estadístico (Vapnik y Chervonenkis, 1971), surgieron los métodos basados en *kernels* (o núcleos), específicamente las máquinas de soporte vectorial (MSV) (Boser, Guyon, y Vapnik, 1992). Esta nueva clase de algoritmos estaba fundamentada en una base teórica que combinaba elementos de estadística y análisis funcional, para caracterizar los conceptos de sobreajuste, optimalidad de soluciones y funciones de costo en el contexto del aprendizaje de máquinas. Además de sus fundamentos teóricos, las MSV mostraron ser una alternativa competitiva a las redes neuronales en cuanto a su desempeño y costo computacional en distintas aplicaciones.

También en la década de los 90s, surgieron nuevos enfoques de AM en donde el manejo de incertidumbre era abordado usando teoría de probabilidades, este es probablemente el punto de mayor similitud entre AM y Estadística (Ghahramani, 2015). Este enfoque procede definiendo una clase de modelos probabilísticos, es decir, se asume que los datos observados han sido generados por un modelo incierto y aleatorio, luego, el aprendizaje consiste en identificar dicho modelo usando Estadística bayesiana. Este es un enfoque elegante y teóricamente sustentado que permitió reinterpretar enfoques anteriores, sin embargo, muchas veces la formulación probabilística no puede ser resuelto de forma exacta y es necesario considerar aproximaciones basadas en Monte Carlo (Neal, 1993) o métodos variacionales (Jordan, Ghahramani, Jaakkola, y Saul, 1999). El enfoque bayesiano permite definir todos los elementos del problema de aprendizaje (modelos, parámetros y predicciones) mediante distribuciones de probabilidad con la finalidad de caracterizar la incertidumbre en el modelo y definir intervalos de confianza en las predicciones, esto incluso permite hacer inferencia sobre modelos con infinitos parámetros (Hjort, Holmes, Müller, y Walker, 2010). En estos casos, el espacio de parámetros pueden ser todas las posible soluciones de un problema de aprendizaje, por ejemplo, el conjunto de las funciones continuas en regresión (Rasmussen y Williams, 2006), o el conjunto de todas las distribuciones de probabilidad en el caso de estimación de densidades (Ferguson, 1973).

Un nuevo resurgimiento de las redes neuronales (RN) se vio en los primeros años de la década del 2000, donde el área se renombró como *deep learning* (Bengio, 2009). Progresivamente, el foco de la comunidad migró desde temáticas probabilistas o basadas en kernels para volver a RN pero ahora con un mayor número de capas. El éxito del enfoque conexionista finalmente logró objetivos propuestos hace décadas, principalmente por dos factores: (i) la gran cantidad de datos disponibles, e.g., la base de datos ImageNet (Deng y cols., 2009), y (ii) la gran capacidad computacional y paralelización del entrenamiento mediante el uso de tarjetas gráficas, lo cual permitió finalmente implementar RNs con billones de parámetros con técnicas para evitar el sobreajuste. El hecho que los parámetros pierdan significado para el entendimiento de las relaciones entre los datos aleja al AM de la Estadística, donde el objetivo es netamente predictivo y no la inferencia estadística: hasta el momento no hay otro enfoque que supere a *deep learning* en variadas aplicaciones. Este fenómeno ha sido principalmente confirmado por los avances de Google, DeepMind y Facebook. De acuerdo a Max Welling, si bien la irrupción de *deep learning* aleja al AM de la Estadística, aún hay temáticas que se nutren de ambas áreas como programación probabilista y computación bayesiana aproximada (Welling, 2015). Adicionalmente, Yoshua Bengio sentencia que aún hay muchos aspectos inciertos de *deep learning* en los cuales los estadísticos podrían ayudar, tal como los especialistas de las ciencias de la computación se han dedicado a los aspectos estadísticos del aprendizaje de máquinas en el pasado (Bengio, 2016).

1.3. Taxonomía del aprendizaje de máquinas

Las herramientas mencionadas en el apartado anterior (árboles, reglas, redes, kernels, modelos estadísticos), son transversales a los tipos de aprendizaje. A grueso modo, existen tres tipos de AM: supervisado, no-supervisado y reforzados. El aprendizaje supervisado (AS), considera datos en forma de pares (dato, etiqueta) y el objetivo es estimar una función $f(\cdot)$ tal se tiene la siguiente igualdad

$$\text{etiqueta} = f(\text{dato}), \tag{1}$$

o bien lo más cercano posible de acuerdo a una medida de error apropiada. El nombre supervisado viene del hecho que los datos disponibles están "etiquetados", y por ende es posible supervisar el entrenamiento (o ajuste)del método. Ejemplos de AS son la identificación de *spam* en correos electrónicos (clasificación), como también la estimación del precio de una propiedad en función de su tamaño, ubicación y otras características (regresión). Ambos casos requieren de un conjunto de entrenamiento (datos etiquetados) construido por un humano. La segunda categoría es el aprendizaje no supervisado (AnS), en donde los datos no están etiquetados y el objetivo es encontrar estructura entre ellos, esto puede ser agrupar subconjuntos de datos que tienen algún grado de relación o propiedades en común, como realizar *clustering* de artículos en distintas grupos basado en su frecuencia aparición de palabras en un texto (Salakhutdinov, 2006). La tercera categoría del AM es el aprendizaje reforzado (AR), en la cual un agente (sintético) aprende a tomar decisiones mediante la maximización de un funcional de recompensa, este es probablemente el tipo de aprendizaje más cercano a la forma en que los animales aprendemos, mediante prueba y error. Un ejemplo de AR es en el cual entrenamos un perro para que aprenda algún truco recompensándolo con comida cada vez realiza la tarea correctamente. El reciente resultado de DeepMind donde una máquina aprendió a jugar Go usando una búsqueda de árbol y una red neuronal profunda es uno de los ejemplos más exitosos de este aprendizaje reforzado (Silver y cols., 2016). Es posible identificar categorías o subdivisiones de AM adicionales a las anteriormente descritas, estas incluyen aprendizaje continuo, activo, semi-supervisado, multi-tarea, etc.

1.4. Relación con otras disciplinas

AM y Estadística. En el análisis de datos es posible identificar dos extremos: el aprendizaje inductivo en donde los datos son abundantes, los supuestos a priori sobre su naturaleza son vagos y el objetivo es realizar predicciones con métodos sofisticados que no necesariamente expliquen los datos. En el otro extremo está el aprendizaje deductivo, donde los datos no son masivos, se adoptan supuestos sobre su naturaleza y el objetivo es aprender relaciones significativas entre los datos usando modelos simples que posteriormente permiten realizar predicciones. Si bien en general al analizar datos nos movemos entre estos dos extremos, podemos decir que la estadística es más cercana al segundo extremo, mientras que el AM contiene componentes que son de enfoque deductivo y muy relacionado a estadística (inferencia bayesiana), pero al mismo tiempo considera otros que son de enfoque inductivo (redes neuronales). Una consecuencia de esto es la importancia que tienen los parámetros en distintos métodos de AM: En los métodos deductivos (estadísticos) los parámetros tienen un rol explicativo de la naturaleza del fenómeno en cuestión que es revelado por los datos, mientras que los métodos inductivos se caracterizan por tener una infinidad de parámetros sin una explicación clara, pues el objetivo muchas veces es hacer predicciones directamente. La relación entre estadística y algunos enfoques a AM es tan cercana que Robert Tibshirani se ha referido a AM como "estadística pretenciosa" (*glorified statistics*).

AM y Programación Clásica. La programación clásica construye algoritmos basados en reglas, es decir, una lista de instrucciones que son ejecutadas en una máquina, lo cual requiere que el programador conozca de antemano el algoritmo a programar, e.g., calcular la transformada de Fourier rápida (FFT) de una grabación de audio. En AM, por el contrario, los algoritmos son precisamente lo que buscamos, por lo tanto, el enfoque que adopta AM es diferente al de la programación clásica en el sentido de que se busca programar la máquina para que aprenda la lista apropiada de instrucciones (en vez de programar la máquina para implementar dichas instrucciones). Tomemos el caso del ajedrez, de acuerdo a (Shannon, 1950) el número de combinaciones posibles para el juego de ajedrez es del orden de 10^40, esto significa que usando programación clásica un programa ingenuo para jugar ajedrez tendría que tener al menos esa cantidad de instrucciones de la forma

"para la combinación C, ejecutar la acción A".

Si todos los humanos sobre la faz de la tierra se uniesen para programar dicha rutina y cada uno pudiese

escribir 10 de estas instrucciones por segundo, nos tomaría 4×10^{21} años, esto es casi un billón (10^{12}) de veces la edad de la tierra (4.54×10^{9}), lo cual hace impracticable adoptar un enfoque clásico de programación. Una alternativa basada en AM es un programa simple en el cual la máquina explora distintos posibles escenarios del tablero e inicialmente toma decisiones aleatorias de qué acción ejecutar para luego registrar si dicha movida llevó a ganar o perder el juego; este enfoque de programación no pretende programar la máquina para "jugar ajedrez" sino para "aprender a jugar ajedrez". Una implementación exitosa de este concepto usando aprendizaje reforzado y redes neuronales profundas para el juego de Go puede verse en (Silver y cols., 2016).

AM, Knowledge Discovery in Databases (KDD) y minería de datos. KDD (Fayyad, Piatetsky-Shapiro, y Smyth, 1996) es "el proceso no trivial de identificar patrones potencialmente válidos, novedosos, útiles y explicativos en datos", y consta de 5 etapas: selección, preparación, transformación, minería e interpretación de datos. La etapa de minería de datos consiste en extraer información desde datos disponibles, por ejemplo, agruparlos en subconjuntos afines o identificar situaciones anómalas. Para esto generalmente se consideran herramientas de AM, especialmente de aprendizaje no-supervisado debido a la cantidad de los datos que deben ser analizados en aplicaciones de KDD, los cuales en general no están no etiquetados y existe poco conocimiento *a priori* de su naturaleza.

AM y Data Science. En línea con sus orígenes en la inteligencia artificial, el objetivo del AM es encontrar relaciones entre datos y hacer predicciones prescindiendo de la intervención humana, es decir, con poco o nada de conocimiento *a priori*. Sin embargo, las técnicas de AM pueden ser complementadas con el conocimiento de un problema específico, en donde especialistas en AM colaboran con especialistas de (i) el área en cuestión, (ii) minería de datos, y (iii) visualización. Esto es *Data Science*, una disciplina colaborativa donde especialistas de variadas áreas trabajan de forma conjunta para hacer un análisis detallado de los datos, con la finalidad de resolver un problema en particular. El perfil del *Data Scientist* es muy completo, pues debe tener conocimiento de AM, estadística, programación, minería de datos, interactuar con especialistas y entender el impacto comercial del análisis. De hecho, la posición de *Data Scientist* ha sido considerada la más sexy del siglo XXI según *Harvard Business Review* (Davenport y Patil, 2012).

1.5. Estado del aprendizaje de máquinas y desafíos

El aprendizaje de máquinas es una reciente disciplina que provee de herramientas a una gran cantidad de disciplinas, pero también es un área de investigación en sí misma con una activa comunidad y muchas preguntas abiertas. Desde el punto de vista algorítmico es posible identificar en primer lugar el desafío del entrenamiento en línea, es decir, cómo ajustar un algoritmo/modelo cada vez que se dispone de un nuevo dato para operar continuamente (e.g. análisis de series de tiempo). Este es un concepto fundamental en procesamiento adaptativo de señales y no ha sido tomado en cuenta satisfactoriamente aún por la comunidad de AM. Otro desafío que tiene relación con la implementación de algoritmos es la capacidad de escalar el entrenamiento de AM para bases de datos masivas y no estructuradas (i.e., *Big Data*). Esto último porque en general los métodos de AM son costosos de implementar y entrenar, pues se enfocan en descubrir estructura y no en procesar datos de alta dimensión *per se*. Sin embargo, esta habilidad es cada vez más necesaria en la era de la información donde, los conceptos que actualmente se usan para lidiar con grandes volúmenes de datos son básicos comparados con es estado del arte en AM.

También es posible identificar desafíos en un plano conceptual, como por ejemplo la "transferencia de aprendizaje", en donde la experiencia adquirida en la realización de una tarea (e.g., reconocimiento de automóviles) sirve como punto de partida para una tarea relacionada (e.g., reconocimiento de camiones). Este concepto es encontrado en el aprendizaje humano también, por ejemplo cuando un físico o matemático migra al mundo bancario sin tener conocimiento *a priori* de finanzas pero aposentador exitosamente en dicha área en poco tiempo. Por otro lado, un desafío ético son los riegos del uso de AM: el avance de AM parece a veces descontrolado y abre interrogantes con respecto de la legislación sobre el actuar de máquinas inteligentes. Por ejemplo, ¿quién es responsable en un accidente en el que está involucrado un

automóvil autónomo? Estos últimos dos desafíos, el conceptual y el ético, revelan que hay una dimensión importante que no hemos explorado y que, a pesar de los avances teóricos y sobretodo aplicados del AM, estamos lejos de entender la inteligencia. Como ha sido expuesto en (Gal, 2015), nuestra relación con el entendimiento de la inteligencia mediante el uso del AM puede ser entendido como el *Homo erectus* hace 400.000 años frotando dos ramas para producir fuego. Ellos usaron el fuego para abrigarse, cocinar y cazar, sin embargo, esto no quiere decir que entendían por qué al frotar dos ramas generaban fuego o, peor aún, qué es el fuego. Estamos en una etapa temprana del entendimiento del aprendizaje, en la que usamos estas herramientas "inteligentes" para nuestro bienestar, sin embargo, estamos lejos de entender la ciencia que hay detrás.

2. Regresión Lineal

El problema de regresión busca determinar la relación entre una variable *independiente* (entrada, estímulo o característica; usualmente denotada por x) y una variable *independiente* (salida, respuesta o etiqueta; usualmente denotada y). Intuitivamente, un modelo de regresión permite entender cómo cambia la variable dependiente cuando la variable independiente es modificada. Esta relación entre ambas variables es representada por una función, consecuentemente, el problema de regresión es equivalente a encontrar una función definida desde el espacio de la entrada x al de la salida y. De esta forma, en base a (i) el espacio de posible funciones donde se busque dicha relación, e.g., los polinomios de grado menor o igual a 5, y a (ii) el criterio de búsqueda que se aplique, e.g., mínimos cuadrados, podemos obtener distintas soluciones para el problema de regresión.

El escenario básico de regresión, y que sirve de base para casos más complejos, es el de regresión lineal. En este caso, el espacio de funciones donde se busca la relación entre las variables dependientes e independientes es el de las funciones lineales afines. Específicamente, para un conjunto de entrenamiento D que contiene $N \in \mathbb{N}$ observaciones de entrada y salida, respectivamente $\{x_i\}_{i=1}^N$ y $\{y_i\}_{i=1}^N$, de la forma

$$D = \{(x_i, y_i)\}_{i=1}^N \subset \mathbb{R}^M \times \mathbb{R}, \tag{2}$$

la regresión lineal busca encontrar un modelo lineal, es decir, una función $f(\cdot)$ definida por

$$\begin{aligned} f\colon \mathbb{R}^M &\to \mathbb{R} \\ x &\mapsto f(x) = a^\top x + b, \quad a \in \mathbb{R}^M, b \in \mathbb{R}, \end{aligned} \tag{3}$$

que *mejor represente* la forma en que la variable y depende de la variable x, en base a las las observaciones contenidas en el conjunto D en la ec. (2). Antes de proceder a definir un criterio de *mejor representación*, el siguiente recuadro justifica la elección de modelos lineales.

¿Por qué consideramos el caso lineal en particular?

Existen distintas razones para estudiar los modelos lineales. En primer lugar, con el criterio de mínimos cuadrados que veremos a continuación, el modelo lineal es el único que admite resolución de forma explícita (o, como diremos alternativamente, *tiene forma cerrada*). Además de calcular dicha solución, la existencia de su forma cerrada nos permite interpretar las propiedades de dicha solución y en qué casos ésta tiene sentido. En segundo lugar, los resultados que obtendremos a continuación requieren linealidad solo en los parámetros—ver ec. (3)—y no necesariamente en la variable independiente x. Por esta razón, el estudio del modelo lineal también incluye modelos no lineales del tipo

$$\begin{aligned} f\colon \mathbb{R}^M &\to \mathbb{R} \\ x &\mapsto f(x) = \theta^\top \phi(x), \quad \theta \in \mathbb{R}^{M'}, \end{aligned} \tag{4}$$

donde $\phi\colon \mathbb{R}^M \to \mathbb{R}^{M'}$ es una función no lineal sin parámetros libres. Es decir, estrictamente hablando deberíamos referirnos a los modelos lineales como *lineales en los parámetros* y no necesariamente *lineales en la entrada*.

Finalmente, cuando los parámetros a determinar en el problema de regresión afectan de forma no lineal la relación entre las variables dependiente e independiente, el análisis presentado a continuación no es válido y en general la solución óptima de mínimos cuadrados no tiene forma cerrada.

2.1. Mínimos cuadrados

En el contexto recién presentado, aflora naturalmente la siguiente pregunta: *¿qué es una buena función $f(\cdot)$?* o, equivalentemente, *¿cómo cuantificar la bondad de un modelo de regresión lineal?* Una práctica ampliamente utilizada es elegir la función $f(\cdot)$ en la ec. (3) de acuerdo al criterio de **mínimos cuadrados**. Es decir, elegir la función $f(\cdot)$ que minimiza la suma de los cuadrados de las diferencias entre las observaciones $\{y_i\}_{i=1}^N$ y las predicciones calculadas por la función $\{f(x_i)\}_{i=1}^N$ de acuerdo al siguiente costo:

$$J(D, f) = \frac{1}{2}\sum_{i=1}^{N}(y_i - f(x_i))^2, \tag{5}$$

donde hemos sido enfáticos en que el costo depende del conjunto de entrenamiento $D = \{(x_i, y_i)\}_{i=1}^N$ y la función f, sin embargo, cuando estas cantidades son claras, nos referiremos al costo simplemente como J. Además, denotamos la (o las) funciones que satisfacen el criterio de mínimos cuadrados mediante

$$f^\star = \underset{f \text{ es lineal}}{\arg\min}\, J. \tag{6}$$

Debido a la forma lineal de $f(\cdot)$, resolver este problema de optimización es equivalente a encontrar los parámetros a y b en la ec. (3). Es decir:

$$a^\star, b^\star = \underset{a,b}{\arg\min}\, \frac{1}{2}\sum_{i=1}^{N}(y_i - a^\top x_i - b)^2. \tag{7}$$

Observemos que el costo en la ec. (7) es cuadrático en a y b, por lo qur el problema de optimización tiene un único mínimo que puede ser encontrado explícitamente. Para esto, como la función f en la ec. (3) no es lineal sino que *afín*, hacemos el siguiente cambio de variable:

$$\begin{bmatrix} x \\ 1 \end{bmatrix} \mapsto \tilde{x} \in \mathbb{R}^{M+1}, \quad \begin{bmatrix} a \\ b \end{bmatrix} \mapsto \theta \in \mathbb{R}^{M+1}, \tag{8}$$

con lo cual el funcional cuadrático a minimizar se convierte en

$$J = \frac{1}{2}\sum_{i=1}^{N}(y_i - \theta^\top \tilde{x}_i)^2, \tag{9}$$

y el parámetro θ de mínimos cuadrados puede ser encontrado aplicando las condiciones de primer orden de la siguiente forma:

$$\begin{aligned}
\nabla_\theta J = 0 \Leftrightarrow & \sum_{i=1}^{N}(y_i - \theta^\top \tilde{x}_i)\tilde{x}_i^\top = 0 && \text{def. } J \\
\Leftrightarrow & \sum_{i=1}^{N} y_i \tilde{x}_i^\top = \sum_{i=1}^{N} \theta^\top \tilde{x}_i \tilde{x}_i^\top && \text{ordenar} \\
\Leftrightarrow & \theta^\top = \sum_{i=1}^{N} y_i \tilde{x}_i^\top \left(\sum_{i=1}^{N} \tilde{x}_i \tilde{x}_i^\top\right)^{-1} && \text{despejar } \theta^\top \\
\Leftrightarrow & \theta = \left(\sum_{i=1}^{N} \tilde{x}_i \tilde{x}_i^\top\right)^{-1} \sum_{i=1}^{N} \tilde{x}_i y_i && \text{transponer} \\
\Leftrightarrow & \theta = \left(\tilde{X}^\top \tilde{X}\right)^{-1} \tilde{X}^\top Y, && \text{def. } \tilde{X} \text{ y } Y
\end{aligned} \tag{10}$$

donde $\tilde{X}$ y Y son las matrices de datos definidas por

$$\tilde{X} = \begin{bmatrix} \tilde{x}_1^\top \\ \vdots \\ \tilde{x}_N^\top \end{bmatrix} \in \mathbb{R}^{N\times(M+1)}, \quad Y = \begin{bmatrix} y_1 \\ \vdots \\ y_N \end{bmatrix} \in \mathbb{R}^N. \tag{11}$$

Con los parámetros del modelo regresión lineal encontrados con el criterio de mínimos cuadrados, es posible implementar la solución y comprarla visualmente con los datos. La Fig. 1 muestra la regresión lineal correspondiente a chirridos de grillos por segundo en función de la temperatura (Pierce, 1949).

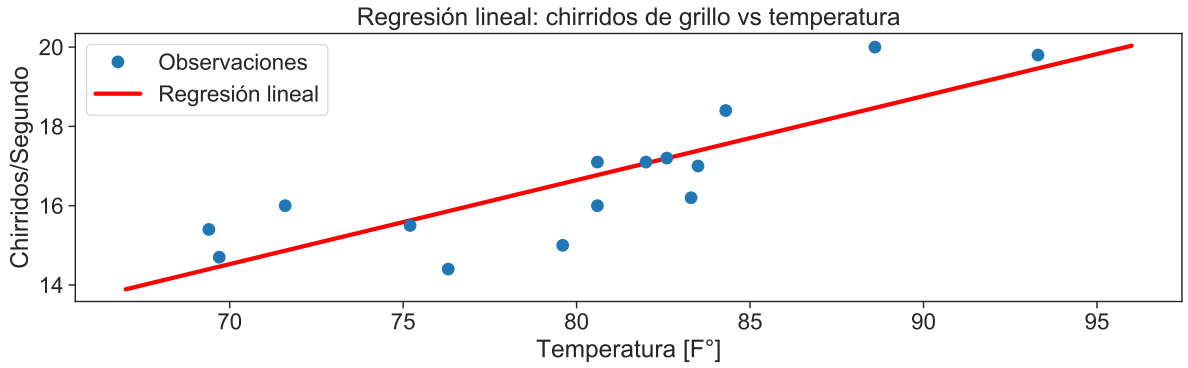


Fig. 1. Ejemplo de regresión lineal usando mínimos cuadrados sobre la base de datos de chirridos versus temperatura.

La expresión $\left(\tilde{X}^\top\tilde{X}\right)^{-1}\tilde{X}^\top$ en la ec. (10) es conocida como la pseudo-inversa de Moore-Penrose (Ben-Israel y Greville, 2006, p. 7). Observemos que una condición **necesaria** para que esta pseudo-inversa esté bien definida es que la cantidad de observaciones (N) sea mayor o igual que la cantidad de dimensiones ($M+1$). Esto es porque la matriz $\tilde{X}^\top\tilde{X}$ es de tamaño $(M+1)\times(M+1)$ y su rango es mín$\{N, M+1\}$, consecuentemente, para que $\tilde{X}^\top\tilde{X}$ tenga rango completo (y por ende sea invertible), se debe cumplir al menos que $N \geq M+1$. Adicionalmente, una condición **necesaria y suficiente** para la existencia de la solución de mínimos cuadrados en la ec. (10), requiere que las $N \geq M+1$ observaciones[2] sean linealmente independientes, pues de esta forma los términos que componen la pseudo-inversa son efectivamente linealmente independientes y ésta tiene rango completo. Es claro que para el caso de variables continuas es muy poco usual que dos observaciones sean perfectamente colineales, sin embargo, en el caso de variables categóricas donde las observaciones son asignadas a un número finito de símbolos es probable que dos o más valores para la variable dependiente sean exactamente iguales.

En la práctica, generalmente tendremos más observaciones que parámetros al considerar un modelo lineal y éstas serán linealmente independientes. Sin embargo, es posible que las observaciones sean tal que la inversión de la matriz $\tilde{X}^\top\tilde{X}$ sea numéricamente inestable. Esto ocurre fundamentalmente en dos casos ilustrados en el siguiente recuadro.

¿Matriz cuasi-singular o incorrectamente escalada?

Al tratar de invertir una matriz de forma computacional, probablemente hemos obtenido un mensaje de la forma `matrix is singular, close to singular or badly scaled`. Veremos dos ejemplos para entender de dónde viene esta advertencia.

[2]Recordemos que nos referimos a las observaciones aumentadas $\tilde{x}$

Caso 1: Consideremos la matriz

$$A = \begin{bmatrix} 10^{50} & 1 \\ 10^{50} & 2 \end{bmatrix} \tag{12}$$

dicha matriz es claramente invertible y su inversa puede ser calculada mediante

$$A^{-1} = \frac{1}{10^{50} \cdot 2 - 10^{50} \cdot 1} \begin{bmatrix} 2 & -1 \\ -10^{50} & 10^{50} \end{bmatrix} = \begin{bmatrix} 2 \cdot 10^{-50} & -10^{-50} \\ -1 & 1 \end{bmatrix}, \tag{13}$$

donde cuyos elementos difieren en 50 órdenes de magnitud. Sin embargo, la representación usual que consideramos cuando programamos es la de punto flotante de precisión simple, la cual considera el menor valor (de magnitud mayor que cero) de $2^{-127} \approx 10^{-38}$. Consecuentemente, los valores más pequeños que este límite serán aproximados por el elemento más cercano, es decir, cero. Utilizando la inversa aproximada, denotada $\tilde{A}^{-1}$, resulta en errores como el siguiente:

$$A\tilde{A}^{-1} = \begin{bmatrix} 10^{50} & 1 \\ 10^{50} & 2 \end{bmatrix} \begin{bmatrix} 0 & 0 \\ -1 & 1 \end{bmatrix} = \begin{bmatrix} -1 & 1 \\ -2 & 2 \end{bmatrix} \tag{14}$$

Caso 2: Consideremos

$$A = \begin{bmatrix} a & a \\ b & b+\epsilon \end{bmatrix} \tag{15}$$

la cual también es invertible para $a, \epsilon > 0$, pues su determinante está dado por

$$\det A = a(b+\epsilon) - ab = a\epsilon > 0, \tag{16}$$

sin embargo, si $\epsilon \ll 1$ entonces el cálculo de la inversa puede sufrir inestabilidades numéricas como en el caso anterior. Sin embargo, observe para un $\eta > 0$ suficientemente grande, la matriz $A + \eta I$ puede tener un determinante arbitrariamente grande (ver Sección 2.1.1).

Es relevante reflexionar por qué consideramos mínimos cuadrados como la métrica de error relacionada al problema de regresión. Existen varias razones por que lo hacemos, tanto técnicas como conceptuales, como también diversas desventajas de este criterio que es importante identificar. Desde del punto de vista técnico, el costo convexo de un modelo lineal (en los parámetros) define un problema de optimización que también es convexo y por ende tiene una solución única. Además, en el caso particular del costo cuadrático, este óptimo puede ser determinado de forma explícita (lo cual es fuertemente deseado), pues está dado únicamente por la inversión de una matriz y no mediante, e.g., una búsqueda iterativa.

Desde un punto de vista conceptual, otra justificación para usar la medida del error cuadrático es que éste representa la varianza muestral. Es decir, si considerásemos que x_i e y_i son observaciones iid de variables aleatorias (VAs) x e y respectivamente, entonces el error cuadrático (relacionado con la función f) definido por

$$e = \sum_{i=1}^{N} (y_i - f(x_i))^2, \tag{17}$$

es la varianza muestral de la variable aleatoria $y - f(x)$, definida como el *error* de estimación. De igual forma, la varianza de la suma de múltiples variables aleatorias (pensemos en errores acumulados, los cuales son independientes) corresponde a la suma de las varianzas de dichas VAs. Esto ocurre precisamente cuando usamos el exponente igual a 2, y no si usáramos 1.95 o 2.05.

Podemos además justificar el uso del error cuadrático con una motivación geométrica. Recordemos que el problema de regresión (lineal) requiere encontrar una solución aproximada de un sistema lineal sobredeterminado definido por

$$\tilde{X}\theta = Y, \tag{18}$$

donde la cantidad de incógnitas $(M+1)$ es ampliamente superada por el número de ecuaciones (N). Como

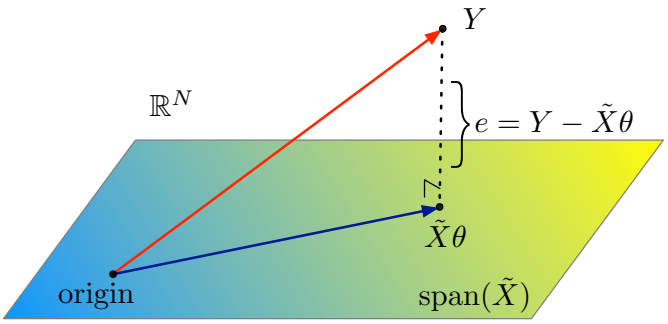


Fig. 2. Interpretación geométrica de la regresión lineal y mínimos cuadrados

esta solución, desde el punto de vista de un sistema lineal, no existe, uno puede proceder a encontrar la solución para θ que reporta *la menor discrepancia* entre ambos lados de la ec. (18). En este sentido, podemos identificar el espacio formado por todos los posibles valores que toma la combinación lineal $\tilde{X}\theta$ para distintos valores de $\theta \in \mathbb{R}^{M+1}$, es decir, el *span* de todas las columnas de $\tilde{X}$ (los datos) definido como el *espacio de las columnas de* $\tilde{X}$ o, simplemente, span($\tilde{X}$). Luego, podemos identificar el elemento de dicho espacio que está más cerca de Y como la proyección del propio Y en span($\tilde{X}$). Esto está ilustrado en la Fig. 2, donde la condición para identificar dicha proyección es precisamente que el vector error $\tilde{X}\theta - Y$ sea ortogonal al espacio span($\tilde{X}$) generado por los datos (de entrada), consecuentemente, ocupando el producto interno tenemos que

$$(\tilde{X}\theta - Y)^\top \tilde{X} = 0, \tag{19}$$

lo cual nos lleva directamente a la solución de mínimos cuadrados.

Finalmente, notemos que el criterio de mínimos cuadrados (MC) también tiene desventajas. Implícitamente, MC está intrínsecamente relacionado con un supuesto de gaussianidad de los datos—esto será evidente cuando estudiemos el criterio de máxima verosimilitud—consecuentemente, el uso de MC produce estimaciones razonables cuando la relación entre x e y es simétrica y sin *grandes desviaciones*. Por el contrario, cuando existen datos que se alejan mucho de dicha la tendencia buscada, las estimaciones encontradas mediante MC pueden desviarse considerablemente de la solución buscada, esto se debe precisamente a la contribución cuadrática del error, donde, coloquialmente, una muestra *muy alejada* pesa tanto o más que varias muestras *ligeramente alejadas*. La Fig. 3 ilustra este fenómeno para el mismo ejemplo de los chirridos en la Fig. 1, donde se ha introducido un *outlier*, es decir, una observación que está inusualmente alejada de los datos y se ha recalculado el resultado de la regresión lineal mediante el criterio de MC. Se puede ver cómo se deteriora la estimación solo con la introducción de un nuevo dato.

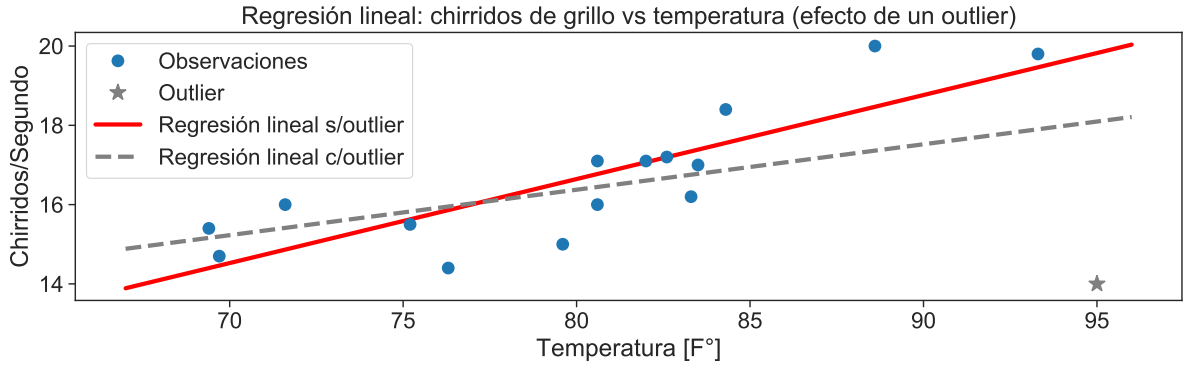


Fig. 3. Efecto de un *outlier* en la regresión lineal usando mínimos cuadrados: Se ha agregado un dato erróneo (*outlier* en gris) y se ha recalculado la regresión lineal, note cómo la inclusión de dicho punto deteriora el resultado de la regresión.

La lección que queda de este ejemplo es que debemos considerar una métrica *ad hoc* al problema que estamos considerando, por ejemplo, si es muy probable que existan outliers, no debemos penalizar cuadráticamente los errores. De igual forma, al elegir una métrica de error debemos verificar cuán relevante es que el error de regresión sea i) nulo vs muy pequeño, o bien ii) grande vs extremadamente grande. La Fig. 4 presenta cuatro métricas de error (como función del propio error), donde podemos interpretar sus propiedades.

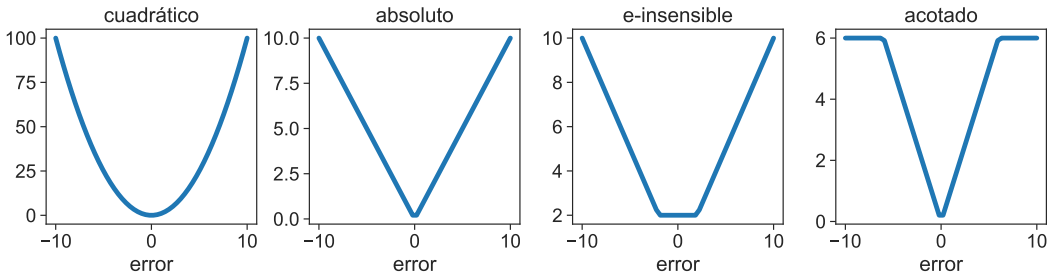


Fig. 4. Distintas funciones de costo en función del error de estimación, de izquierda a derecha: cuadrático, absoluto (crecimiento lineal en función del error), ϵ-insensible (es irrelevante si el error está entre 0 o ϵ) y acotado (es irrelevante si el error es mayor que cierto umbral).

2.1.1. Regularización: ajuste versus generalización

Perseguir ciegamente la solución de mínimos cuadrados puede resultar, como discutimos en la sección anterior, en situaciones donde la inversa de Moore-Penrose sea *cercana* a singular, especialmente en los casos que las observaciones son parecidas o redundantes. En este sentido, debemos considerar un criterio que no simplemente busque un ajuste a los datos, sino que también promueva ciertas propiedades de la solución, por ejemplo, suavidad, bajas magnitudes de los parámetros o incluso pocos parámetros. Nos referiremos a estas soluciones como *regulares*, y el objetivo de este apartado será *regularizar* la solución de mínimos cuadrados.

Las penalizaciones a considerar en el problema de regresión pueden ser codificadas directamente en la función de costo. Por ejemplo, ésta puede incluir un término que promueve el ajuste de los datos y otro término que sanciona soluciones que se alejan de lo deseado. Un criterio estándar de penalización es el basado en la norma de los parámetros, es decir,

$$J_\rho = \frac{1}{2}\sum_{i=1}^{N}(y_i - \theta^\top \tilde{x}_i)^2 + \frac{\rho}{p}||\theta||_p^p,\ p \in \mathbb{R}_+, \tag{20}$$

donde $||\cdot||_p$ denota la norma ℓ_p, es decir, $||\theta||_p = \left(\sum_{j=1}^{N}|\theta_j|^p\right)^{\frac{1}{p}}$ y el parámetro $\rho \geq 0$ tiene el rol de balancear la importancia entre ajuste (primer término) y regularidad de la solución (segundo término). Distintos valores de p inducen distintos propiedades sobre las soluciones, siendo las más usadas las correspondientes a $p = 1$, conocido como **LASSO**[3] (Tibshirani, 1996), y $p = 2$ conocido como **regularización de Tikhonov** (Tikhonov y Arsenin, 1977) o bien ***Ridge Regression*** (Hoerl y Kennard, 1970).

Una ventaja de la regularización de Tikhonov es que su solución, al igual que el caso de mínimos cuadrados no regularizados, puede ser encontrada en forma exacta. En efecto, para $p = 2$ el término de regularización puede ser expresado como $||\theta||_2 = \theta^\top\theta$, con lo que el minimizante del costo cuadrático

[3] *Least Absolute Shrinkage and Selection Operator.*

regularizado está dado por:

$$\begin{aligned}
\nabla_\theta J_\rho = 0 &\Leftrightarrow \sum_{i=1}^{N}(\theta^\top \tilde{x}_i - y_i)\tilde{x}_i^\top + \rho\theta^\top = 0 && \text{def. } J \\
&\Leftrightarrow \sum_{i=1}^{N} y_i\tilde{x}_i^\top = \sum_{i=1}^{N}\theta^\top\tilde{x}_i\tilde{x}_i^\top + \rho\theta^\top && \text{ordenar} \\
&\Leftrightarrow \theta^\top = \sum_{i=1}^{N} y_i\tilde{x}_i^\top \left(\sum_{i=1}^{N}\tilde{x}_i\tilde{x}_i^\top + \rho\mathbb{I}\right)^{-1} && \text{despejar } \theta^\top \\
&\Leftrightarrow \theta = \left(\sum_{i=1}^{N}\tilde{x}_i\tilde{x}_i^\top + \rho\mathbb{I}\right)^{-1}\sum_{i=1}^{N}\tilde{x}_i y_i && \text{transponer} \\
&\Leftrightarrow \theta = \left(\tilde{X}^\top\tilde{X} + \rho\mathbb{I}\right)^{-1}\tilde{X}^\top Y. && \text{def. } \tilde{X} \text{ y } Y
\end{aligned} \tag{21}$$

De la última expresión, es posible ver que el requerimiento de que las observaciones disponibles sean (i) más que la dimensión $M+1$ y que además (ii) éstas sean colineales ya no es necesario para que la solución esté bien definida. De hecho, la matriz $\tilde{X}^\top\tilde{X}$ puede efectivamente estar cercana a ser no invertible, sin embargo, es posible *regularizar* la solución forzando que la matriz $\left(\tilde{X}^\top\tilde{X} + \rho\mathbb{I}\right)$ sea arbitrariamente lejana de las matrices singulares (o tenga un determinante arbitrariamente grande) aumentando el valor de ρ.

Es relevante entender por qué una disminución en la norma de los parámetros, puede ayudar a ajustar *mejores* modelos. A primera impresión, un podría pensar que el criterio de mínimos cuadrados regularizados (MCR) en ningún caso puede reportar mejores modelos que su contraparte MC, pues MCR es una variante restringida del problema original y consecuentemente solo puede *en el mejor de los casos* alcanzar la solución óptima. Esto es cierto si por *mejor modelo* solo consideramos el error cuadrático medio (ECM), sin embargo, solo enfocarse en esta métrica no siempre es el mejor opción. Para ilustrar este concepto, tomemos las siguientes consideraciones: asumamos que efectivamente los datos cumplen la relación

$$y_i = \underbrace{\theta^\top\tilde{x}_i}_{f_i} + \epsilon_i, \tag{22}$$

donde ϵ_i son observaciones iid de una variable aleatoria de varianza σ^2, θ es un parámetro fijo, los $\tilde{x}_i$ son fijos y $f_i = \theta^\top\tilde{x}_i$ se refiere a la *parte determinista* del modelo. Además, consideremos una estimación del parámetro θ construida en base a un conjunto de entrenamiento $D = \{(\tilde{x}_i, y_i)\}_{i=1}^N$, denotada $\hat{\theta} = \hat{\theta}_D$. Con estas consideraciones, para un nuevo par $(\tilde{x}_\star, y_\star)$, podemos escribir el *costo (cuadrático) esperado* asociado a la **predicción** $\hat{f}_\star = \hat{\theta}^\top\tilde{x}_\star$ mediante el *trade-off* entre sesgo y la varianza (James, Witten, Hastie, y Tibshirani, 2014) dado por

$$\mathbb{E}\,(y_\star - \hat{f}_\star)^2 = \text{Sesgo}(\hat{f}_\star)^2 + \text{Varianza}(\hat{f}_\star) + \sigma^2, \tag{23}$$

donde el valor esperado es tomado con respecto a la ley de ϵ, la única fuente de incertidumbre en este escenario, y

- $\text{Sesgo}(\hat{f}_\star) = \mathbb{E}(\hat{f}_\star) - f_\star$, es una medida de exactitud: ¿cuán buena (en valor esperado) es la estimación con respecto al valor real?
- $\text{Varianza}(\hat{f}_\star) = \mathbb{E}(\mathbb{E}(\hat{f}_\star) - \hat{f}_\star)^2$, es una medida de precisión: ¿cuán disperso es el estimador?
- σ^2 es la varianza de *ruido* ϵ y es la parte irreducible del costo, en el sentido que no puede ser controlada por la elección de $\hat{\theta}$.

Podemos evaluar el sesgo y la varianza para el estimador de mínimos cuadrados, denotado $\hat{\theta}_{MC}$, eligiendo $\hat{f}_\star = \hat{\theta}_{MC}^\top \tilde{x}_\star$. En efecto,

$$\text{Sesgo}(\hat{f}_\star) = \mathbb{E}(\theta_{MC}^\top \tilde{x}_\star) - \theta^\top \tilde{x}_\star = 0 \tag{24}$$

$$\text{Varianza}(\hat{f}_\star) = \sigma^2 \tilde{x}_\star^\top (\tilde{X}^\top \tilde{X})^{-1} \tilde{x}_\star \tag{25}$$

Es decir, el modelo de regresión lineal ajustado mediante MC reporta un estimador insesgado (sesgo nulo) pero con una varianza que depende de los datos en el conjunto de entrenamiento D, la varianza del ruido σ^2 y la propia entrada $\tilde{x}_\star$. Si bien no es posible determinar cuánto es esta varianza sin tomar supuestos estadísticos sobre los $\tilde{x}_i$'s, recordemos que la matriz $\tilde{X}^\top \tilde{X}$ puede ser cercana a singular cuando los datos son pocos, redundantes o colineales, lo cual resultará en alta varianza para la predicción $\hat{f}_\star$. De hecho, si asumiéramos que $\tilde{x}_i \sim \mathcal{N}(0, 1)$ iid, tendríamos que $\text{Varianza}(\hat{f}_\star) = \sigma^2 M/N$, es decir, la varianza es inversamente proporcional a la razón entre la dimensión de las entradas (M) y la cantidad de muestras (N).

Evaluaciones *dentro de muestra* y *fuera de muestra*

Notemos que la expresión en la ec. (23) es una medida de error *fuera de muestra*, pues evalúa un estimador $\hat{\theta}$, construido en base a un conjunto D, en un nuevo dato $(\tilde{x}_\star, y_\star)$ que no está originalmente contenido en D. No debemos confundir esta expresión con el error cuadrático medio, en la ec. (5), el cual representa un error *dentro de muestra*. La evaluación de ambos tipos de costos es clave para diseñar modelos y estimadores que puedan *generalizar* a datos no vistos.

Al penalizar la norma cuadrada del parámetro θ, la regresión de Ridge sacrifica la propiedad insesgada del estimador, pero en retorno construye un estimador que tiene menos varianza que el de MC. Esto puede entenderse como el balance entre: i) confiar únicamente en los datos, los cuales pueden ser pocos o muy ruidoso y consecuentemente insuficientes para determinar un modelo apropiado, y ii) introducir un *sesgo* al modelo (por ejemplo, parámetros pequeños) con la finalidad de robustecer el modelo encontrado en función de los datos disponibles. Esta noción de (sobre-)ajustar a los datos de entrenamiento versus generalizar a nuevos puede ser ilustrada con el siguiente ejemplo: Consideremos $N = 1000$ datos relacionados linealmente (pares de entrada y salida) donde la dimensión de entrada es $M = 100$ generados por el siguiente script.

```
##generacion de datos relacionados linealmente
n_samples, n_features = 1000, 100
rng = np.random.RandomState(0)
X = rng.randn(n_samples, n_features)
theta = rng.randn(n_features,1)
y = X@theta + 10*rng.randn(n_samples, 1)
```

En vez de utilizar todas las muestras de entrenamiento, utilicemos solo $N' = 15$ muestras para entrenar usando los criterios de MC, y regresión de Ridge (RR) con $\rho \in \{40, 80\}$. Repitiendo este proceso 400 veces, podemos analizar cómo se comportan los distintos métodos en cuanto a la magnitud de los parámetros encontrados, el error dentro de muestra (con respecto a los datos de entrenamiento) y el error fuera de muestra (con respecto a los datos no usados para entrenar). La Fig. 5 muestra dichos histogramas, desde donde podemos ver que a mayor ρ (recordemos que MC es equivalente a RR con $\rho = 0$), los parámetros encontrados tienen menor magnitud. Adicionalmente, notemos que el modelo no regularizado (MC) se comporta mejor en evaluación dentro de muestra, sin embargo, sus papeles se invierten cuando se trata de evaluación fuera de muestra: el incluir un sesgo en el ajuste de modelos (RR) puede ayudar a generalizar y no sobreajustar cuando se tienen pocos datos.

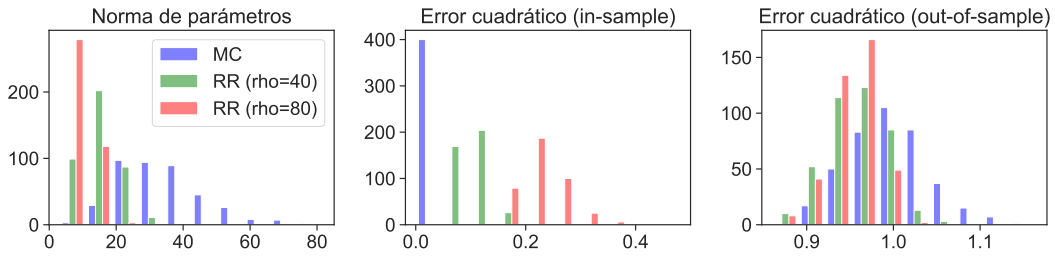


Fig. 5. Mínimos cuadrados versus regresión de Ridge ($\rho \in \{40, 80\}$): determinación de parámetros usando solo 15 muestras para un parámetro de dimensión $M = 100$. De derecha a izquierda podemos ver magnitud de los parámetros encontrados, error dentro de muestra y error fuera de muestra. Experimento repetido 400 veces.

2.1.2. Formulación con restricciones y selección de variables

Para ilustrar cómo el proceso de regularización es equivalente a restringir los parámetros a cumplir con una condición específica, e.g., tener una magnitud dada, consideremos el siguiente problema de regresión lineal con restricciones:

$$\begin{aligned} &\min_{\theta} \frac{1}{2}\sum_{i=1}^{N}(y_i - \theta^\top \tilde{x}_i)^2 \\ &\text{s.a. } ||\theta||_p^p = \tau, \end{aligned} \tag{26}$$

donde asumiremos que $\tau \geq 0$ es una constante conocida. Sabemos que este problema puede ser resuelto en su forma dual mediante la formulación del Lagrangiano dado por

$$L = \frac{1}{2}\sum_{i=1}^{N}(y_i - \theta^\top \tilde{x}_i)^2 - \lambda(||\theta||_p^p - \tau), \tag{27}$$

donde $\lambda \geq 0$ es conocido como el *multiplicador de Lagrange*. Luego, las condiciones de primer orden necesarias para encontrar la solución del problema con restricciones en la ec. (26) están dadas por:

$$\frac{\partial L}{\partial \theta} = 0 \quad \Rightarrow \quad \frac{\partial}{\partial \theta}\left(\frac{1}{2}\sum_{i=1}^{N}(y_i - \theta^\top \tilde{x}_i)^2 - \lambda||\theta||_p^p\right) = 0 \tag{28}$$

$$\frac{\partial L}{\partial \lambda} = 0 \quad \Rightarrow \quad ||\theta||_p^p = \tau, \tag{29}$$

lo cual recupera la forma del problema de minimización de mínimos cuadrados regularizados. Enfatizamos esta relación en el siguiente recuadro.

Mínimos cuadrados regularizados: optimización con restricciones

Observemos que el problema de mínimos cuadrados regularizados, determinado por el costo en la ec. (20), es equivalente al dual de un problema de optimización con restricciones en las ecs. (28)-(29) para un λ dado tal que $\lambda = -\rho/p$. Consecuentemente, como el valor óptimo de λ depende del nivel de la restricción τ, podemos aseverar que **para cualquier $\rho \geq 0$, existe un $\tau \geq 0$ tal que la minimización de** (20) **es equivalente a la de** (26)**.** Consecuentemente, podemos interpretar el problema de MCR como el de MC sujeto a una restricción para (en este caso la norma) del parámetro.

Con la interpretación del problema de optimización con restricciones sobre la norma del parámetro θ, podemos entender distintos regularizadores (distintos $p \geq 0$) mediante sus curvas de nivel. La Fig. 6 ilustra las curvas correspondientes al costo cuadrático (izquierda) y al término de regularización en la eq. (20) para órdenes $p \in \{0.5, 1, 2\}$.

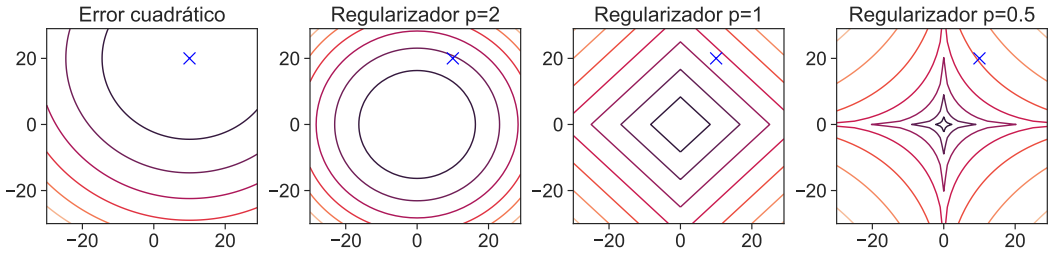


Fig. 6. Curvas de nivel del costo cuadrático para un problema hipotético con solución $\theta = [10, 20]$ (izquierda) y términos de regularización para órdenes $p \in \{0.5, 1, 2\}$. Observe cómo las curvas de nivel atraen el mínimo hacia el origen de distinta forma: $p = 2$ lleva la solución directamente al origen, mientras que $p \in \{0.5, 1\}$ lleva la solución a los bordes, es decir, privilegiando soluciones ralas. La solución $\theta = [10, 20]$ se ha denotado con una cruz azul, recuerde que solo el costo cuadrático depende de este valor, no los términos de regularización.

La formulación en base a restricciones es clave para entender la propiedad de *selección de características* de los MCR. Al estimar el parámetro θ, estamos verificando cuán importante es cada (coordenada de la) entrada o en la jerga de reconocimiento de patrones, cada *característica*. Indirectamente estamos también implícitamente descubriendo cuáles son las características que importan y cuales no, a esto nos referimos como selección de características. Distintas normas para el término de regularización, como las ilustradas en la Fig. 6, inducen distintas propiedades para la solución del problema de MCR. En particular, RR atrae *homogéneamente* el parámetro hacia el origen, lo cual resulta en estimaciones de menor varianza como vimos en el apartado anterior. LASSO ($p = 1$) y el caso $p \leq 1$ en general presenta una propiedad adicional, donde la forma de la curva de nivel permite que usualmente la solución del problema se concentre el las puntas del *diamante* (ver Fig. 6, $p = 1$), llevando algunas coordenadas del parámetro θ directamente a cero. Por esto decimos que LASSO (y $p \leq 1$ en general) tiene la propiedad de selección de variables y entrega modelos ralos con respecto a MC tradicional.

Para ilustrar la propiedad de selección de características, consideremos el *Breast Cancer Wisconsin Data Set*[4], el cual tiene $N = 569$ muestras y un dimensión de entrada de $M = 30$. Los valores para la variable de salida (y) son solo dos, pues este es un problema de clasificación (*cáncer* vs *no-cáncer*), sin embargo, nosotros ajustaremos un modelo de regresión lineal usando MC, RR y LASSO para luego evaluar los pesos encontrados. Usaremos 2/3 de los datos para entrenar y el 1/3 restante para calcular puntajes fuera de muestra. La Fig. 7 presenta los parámetros encontrados para cada uno de los métodos, donde podemos ver la propiedad de selección de variables de LASSO; adicionalmente, la Tabla 1 muestra los puntajes de cada método, tanto dentro como fuera de muestras: en la línea de la discusión anterior, los modelos regularizados presentan mejor generalización y usan menos parámetros (o más parámetros iguales a cero).

[4] `https://archive.ics.uci.edu/ml/datasets/breast+cancer+wisconsin+(original)`

Tabla 1. Puntajes de modelos de regresión implementados en *Breast Cancer Wisconsin Data Set*, más alto es mejor. Observe la superioridad de los modelos regularizados para generalizar.

	in-sample	out-of-sample
MC	**0.7896**	0.6911
RR	0.6905	0.6903
LASSO	0.7452	**0.7242**

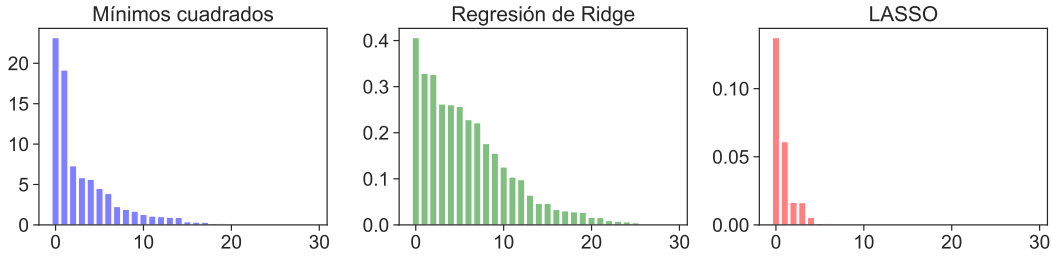


Fig. 7. Parámetros de la regresión lineal del *Breast Cancer Dataset* usando MC, RR y LASSO. Observe cómo RR y LASSO disminuye críticamente la magnitud de los parámetros y, además, LASSO lleva parámetros directamente a cero, resultando en un modelo más simple (i.e, con menos parámetros).

Regularización: Consideraciones generales

Para concluir esta sección, enunciamos las siguientes preguntas para discusión posterior

• **¿es justo comparar MC y MCR en términos del ECM?** Ciertamente no, el criterio de MC siempre reportará un menor ECM, pues ha sido entrenado para minimizar dicho costo. Las ventajas de MCR están en su desempeño fuera de muestra, selección de variables o en general en su habilidad de incorporar sesgo *de diseñador* en la soluciones que no afloren naturalmente de los datos.

• **¿cómo elegir ρ?** De forma general, este *hiperparámetro* determina el balance entre regularización (cuán sesgado) y ajuste (cuán bien replica los datos), consecuentemente lo debemos elegir según nuestra intención. En la práctica, podemos evaluar el desempeño de distintos valores de ρ fuera de muestra con la finalidad de elegir un valor apropiado. Esta técnica se llama *validación cruzada* y busca determinar modelos que no sufran de subajuste, o sobreajuste.

• Vimos que la norma ℓ_p con $0 < p \leq 1$ tiene la propiedad de selección de características, pero, **¿qué pasa con la 'norma' ℓ_0?**. La cantidad $\ell_0(\theta)$ denota la cantidad de elementos no nulos de θ y no es una norma, sin embargo, puede de todas formas ser usada en la definición del costo en la ecu. (20), con la finalidad de directamente penalizar la cantidad de características usadas por el modelo. Desafortunadamente, encontrar la solución usando la "norma" l_0 es muy difícil, sin embargo, bajo ciertas condiciones la consideración de la norma l_1 puede llevar a la misma solución.

2.2. Máxima verosimilitud

En el apartado anterior vimos que el criterio de mínimos cuadrados, y su variante regularizada, ofrecen una alternativa simple, elegante, interpretable y con solución en forma cerrada. Sin embargo, también vimos que dicho criterio sufre de desventajas en cuanto a su capacidad de ajustar modelos en casos generales, pues el criterio de MC es particularmente apropiado para variables contínuas, con

perturbaciones aditivas y simétricamente dispersas con respecto a una tendencia dada. Existen distintos casos donde el criterio de MC no es apropiado, por ejemplo aplicaciones financieras con perturbaciones multiplicativas, mediciones de intensidad como frecuencia de aparición de palabras o sismos en donde las perturbaciones son con alta probabilidad solo positivas, y problemas de clasificación o asignación (*clustering*) en donde las métricas de error toman la forma como "correcto"/"incorrecto", con lo que una medida de error que que reporte ajustes "más incorrectos" no tiene sentido.

Una alternativa natural es considerar una métrica de desempeño distinta y diseñada específicamente en función de cada aplicación con la finalidad de capturar asimetrías, no-estacionariedad, asignación correcta e invarianzas (en el problema de *clustering* por ejemplo) entre otras propiedades. Sin embargo, esta no solo es una tarea tediosa y poco elegante—en el sentido que va en contra los objetivos de inteligencia artificial expuestos en el Capítulo 1—sino que puede ser muchas veces impracticable, pues precisamente no conocemos cuáles son las propiedades de los datos antes de ajustar modelos. Además, usar distintas métricas dificulta la interpretación y comparación de los enfoques considerados. Consecuentemente, nos proponemos considerar un criterio global de ajuste de modelos, el que en cada caso particular *colapse* a una forma explícita que sí es *ad hoc* al problema/modelo en cuestión y permite comparar distintos enfoques de manera unificada.

El enfoque general para ajuste de modelos que consideraremos en esta sección, y continuaremos utilizando durante el resto del curso, será el *criterio de máxima verosimilitud*, dado un conjunto de datos de entrenamiento D. Este es un criterio general para una amplia gama de modelos que, tal como se mencionó en el párrafo anterior, toma una forma específica en cada problema, aunque su solución no siempre es calculable de forma explícita. Este enfoque es radicalmente distinto al de MC y a cualquier otro criterio de "ajuste": con el criterio de MC buscamos un modelo *aproximado* a los datos, donde sabemos que ningún modelo es el modelo correcto, tal que la discrepancia entre el modelo candidato ya los datos sea mínima. Por el contrario, en el criterio de *máxima verosimilitud* nuestro objetivo es encontrar el modelo que—con mayor probabilidad—ha generado *exactamente* los datos observados. Debido a la naturaleza aleatoria de los datos, para implementar este concepto es necesario considerar modelos probabilísticos, de forma de poder calcular la probabilidad de que los datos D hayan sido generados por un modelo m, luego, elegiremos el modelo que maximice dicha probabilidad.

El criterio de máxima verosimilitud (MV) es aplicable a modelos probabilísticos para la generación de datos, a los cuales nos referiremos como *modelo generativos*. Para el caso del problema de regresión, el modelo generativo es cualquiera que modele la variable de salida como una variable aleatoria y a través de una distribución condicional (a la entrada x y el parámetro θ) de la forma

$$y|x,\theta \sim p(y|x,\theta), \tag{30}$$

donde enfatizamos que y es la única variable aleatoria y tanto el parámetro θ como la entrada x son cantidades fijas (la primera desconocida y la segunda conocida).

Notación sobre variables aleatorias

Estrictamente, en base a la notación estándar en probabilidades, la expresión correcta para el lado izquierdo de la ec. (30) debiese ser

$$Y = y|x,\theta, \tag{31}$$

pues Y denota la variable aleatoria, e y el valor que ésta toma. Sin embargo, seguiremos la usanza de la comunidad de Aprendizaje de Máquinas donde denotamos la tanto la variable aleatoria como su valor indistintamente con la letra minúscula, e.g., y. Seguiremos esta notación a menos que sea estrictamente necesario para evitar confusión. Además, en todos los casos asumiremos que las distribuciones de probabilidad consideradas tienen densidad con respecto a

alguna medida—usualmente *Lebesgue* (caso continuo) o *cuenta-puntos* (caso discreto)—ambas denotadas indistintamente por $p(\cdot)$. Finalmente, usualmente escribiremos

$$y|x \sim p(y|x), \tag{32}$$

sin enunciar explícitamente la dependencia del parámetro θ.

En particular, en el caso de la regresión lineal podemos considerar el siguiente modelo generativo:

$$y = a^\top x + b + \epsilon, \quad \epsilon \sim \mathcal{N}(0, \sigma_\epsilon^2), \tag{33}$$

el cual consta de una parte determinística (lineal en x) y una parte aleatoria caracterizada por la variable aleatoria ϵ, la cual hemos elegido gaussiana con media cero y varianza σ_ϵ^2. El modelo probabilístico en la ec. (33) puede expresarse mediante la siguiente densidad condicional

$$y|x \sim p(y|x, \theta) = \mathcal{N}(y; a^\top x + b, \sigma_\epsilon^2), \tag{34}$$

donde hemos denotados el vector de todos los parámetros del modelo mediante $\theta = [a, b, \sigma_\epsilon^2]$.

Si bien, lo siguiente no es necesario en el caso general, usualmente asumiremos que las realizaciones del modelo anterior, i.e., los datos $\{y_i\}_{i=1}^N$ generados a partir de la entrada $\{x_i\}_{i=1}^N$, son **condicionalmente independientes** dado el modelo. Esto significa que *si conociésemos el modelo*, o equivalentemente, si conociésemos θ, y dos entradas x_i, x_j, entonces las salidas correspondientes y_i, y_j son independientes. Es importante clarificar que los valores generados por el modelo $\{y_i\}_{i=1}^N$ **no son independientes**. En efecto, si fuesen independientes no podríamos hacer predicciones: la predicción de una observación nueva $y_\star$ en base a una secuencia de observaciones $\{y_i\}_{i=1}^N$ estaría dada por[5]

[esto es falso] $$p(y_\star|\{y_i\}_{i=1}^N) \overset{\text{(prob. cond.)}}{=} \frac{p(y_\star, \{y_i\}_{i=1}^N)}{p(\{y_i\}_{i=1}^N)} \overset{\text{(indep.)}}{=} \frac{p(y_\star), p(\{y_i\}_{i=1}^N)}{p(\{y_i\}_{i=1}^N)} = p(y_\star), \tag{35}$$

es decir, las observaciones pasadas no aportarían para la predicción. Por el contrario, como nuestro supuesto es de **independencia condicional** la expresión correcta es la siguiente:

[esto es verdadero] $$p(y_\star|\{y_i\}_{i=1}^N, \theta) = p(y_\star|, \theta), \tag{36}$$

lo cual quiere decir que las observaciones pasadas no son útiles para predecir el futuro **solo si conozco el modelo**. Esto es evidente, pues si conozco el modelo, no necesito datos para saber de $y_\star$.

El supuesto de independencia condicional está garantizado al imponer que las realizaciones de $\epsilon \sim \mathcal{N}(0, \sigma_\epsilon^2)$ sean *independientes e idénticamente distribuidas* (iid). Esto es fundamental para poder aprender el modelo desde múltiples observaciones, pues intuitivamente todas las observaciones aportan evidencia no redundante sobre el parámetro en común θ. Por el contrario, si las observaciones fuesen condicionalmente dependientes, entonces la información que reportan para estimar θ sería redundante. Igualmente, si no todas las observaciones siguiesen la misma distribución, entonces cada una tendría "su propio θ" y solo tendríamos "un dato" para estimar cada parámetro. Más adelante en el curso veremos casos donde los datos no son iid pero asumimos cierta regularidad en los modelos que permiten determinar sus parámetros.

2.2.1. Función de verosimilitud

Definición 2.1 (Verosimilitud)**.** Consideremos un modelo generativo definido mediante la densidad de probabilidad $y \sim p(y|\theta)$, donde el parámetro $\theta \in \Theta$ y un conjunto de observaciones $\{y_i\}_{i=1}^N$ generado por dicho modelo. La función $L(\theta) : \Theta \to \mathbb{R}$ definida como la probabilidad de los datos observados condicional al parámetro θ, es decir,

$$\theta \mapsto L(\theta) = p(\{y_i\}_{i=1}^N|\theta), \tag{37}$$

[5]Hemos ignorado la dependencia de las variables independientes $\{x_i\}_{i=1}^N$, $x_\star$.

es conocida como *verosimilitud* del modelo $p(y|\theta)$ o, equivalentemente, del parámetro θ. En algunos casos, consideraremos la notación $L_{\mathbf{y}}(\theta)$ para enfatizar que la verosimilitud es tomada con respecto a las observaciones $\mathbf{y}$.

La definición anterior es elocuente: la función de verosimilitud precisamente cuantifica cuán verosímil es un modelo (o equivalentemente, parámetro) de haber generado las observaciones $\{y_i\}_{i=1}^N$. En este sentido, ante dos valores candidatos para el parámetro, por ejemplo θ_1 y θ_2, éstos pueden se evaluados mediante la comparación de $L(\theta_1)$ y $L(\theta_2)$. En efecto, si la razón $L(\theta_1)/L(\theta_2)$ es, por ejemplo, 3, entonces diremos que *el valor de θ sea θ_1 es 3 veces más verosímil a que sea θ_2*. En este sentido, la función de verosimilitud $L(\theta)$ representa una medida relativa de la *bondad* de cada valor que el parámetro pueda tomar en función de los datos observados.

Es importante enfatizar que la función $L(\theta)$ **no es una densidad de probabilidad**. En efecto, podemos considerar la siguiente función en dos variables: θ y $\mathbf{y} = \{y_i\}_{i=1}^N$

$$L(\theta, \mathbf{y}) = p(\mathbf{y}|\theta), \tag{38}$$

la cual toma distintos significados si fijamos una de las variables: Si fijamos el valor del parámetro θ, entonces, $L(\theta, \cdot) = p(\cdot|\theta)$ es una densidad de probabilidad, en particular,

$$\int_{\mathbb{R}^N} L(\cdot, \mathbf{y}) \mathrm{d}\mathbf{y} = \int_{\mathbb{R}^N} p(\mathbf{y}|\theta) \mathrm{d}\mathbf{y} = 1, \tag{39}$$

lo que quiere decir que para "cualquier θ", entonces, $p(\mathbf{y}|\theta)$ es un modelo válido. Por el contrario, si fijamos $\mathbf{y}$, entonces obtenemos la función de verosimilitud $L(\cdot, \mathbf{y}) = p(\mathbf{y}|\cdot)$, la cual no necesariamente integra uno con respecto a θ.

Ejemplo: Verosimilitud para el modelo gaussiano (muestras independientes)

Consideremos un modelo gaussiano definido por

$$y \sim p(y|\mu, \sigma^2) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(\frac{-(y-\mu)^2}{2\sigma^2}\right), \tag{40}$$

y las observaciones $\mathbf{y} = \{y_i\}_{i=1}^N$ iid. La verosimilitud de $\theta = [\mu, \sigma]$ está dada por

$$\begin{aligned}
L(\theta) = p(\mathbf{y}|\mu, \sigma^2) &\overset{\text{(iid)}}{=} \prod_{i=1}^N p(y_i|\mu, \sigma^2) = \frac{1}{(2\pi\sigma^2)^{N/2}} \exp\left(\frac{-\sum_{i=1}^N (y_i-\mu)^2}{2\sigma^2}\right) \qquad (41)\\
\text{[expandir } (\cdot)^2\text{]} \quad &= \frac{1}{(2\pi\sigma^2)^{N/2}} \exp\left(\frac{-N(\mu^2 - 2\mu\sum_{i=1}^N y_i/N + \sum_{i=1}^N y_i^2/N)}{2\sigma^2}\right)\\
\text{[nikita nipone]} \quad &= \frac{1}{(2\pi\sigma^2)^{N/2}} \exp\left(\frac{-N(\mu^2 - 2\mu\sum_{i=1}^N y_i/N + \sum_{i=1}^N y_i^2/N \pm (\sum_{i=1}^N y_i/N)^2)}{2\sigma^2}\right)\\
\text{[formar } (\cdot)^2\text{]} \quad &= \frac{1}{(2\pi\sigma^2)^{N/2}} \exp\left(\frac{-(\mu - \sum_{i=1}^N y_i/N)^2}{2\sigma^2/N}\right) \exp\left(\frac{-(\sum_{i=1}^N y_i^2/N - (\sum_{i=1}^N y_i/N)^2)}{2\sigma^2/N}\right)\\
\text{[notación]} \quad &= \frac{1}{(2\pi\sigma^2)^{N/2}} \exp\left(\frac{-(\mu - \bar{y})^2}{2\sigma^2/N}\right) \exp\left(\frac{-(\bar{s} - \bar{y}^2)}{2\sigma^2/N}\right),
\end{aligned}$$

donde $\bar{y} = \frac{1}{N}\sum_{i=1}^N y_i$ es el promedio de las observaciones y $\bar{s} = \sum_{i=1}^N y_i^2/N$ es el promedio de los cuadrados de las observaciones. Observemos que como funciones de μ y σ^2, la expresión anterior es respectivamente proporcional a las densidades Normal (para μ) y Gamma-inversa

(para σ^2). Sin embargo, recordemos que esta expresión no necesariamente integra uno y por ende es solo coincidentemente proporcional a una pdf conocida. La Fig. 8 muestra la densidad normal ($\mu = 0, \sigma = 1$) y la verosimilitud para ambos parámetros con 20 y 200 muestras.

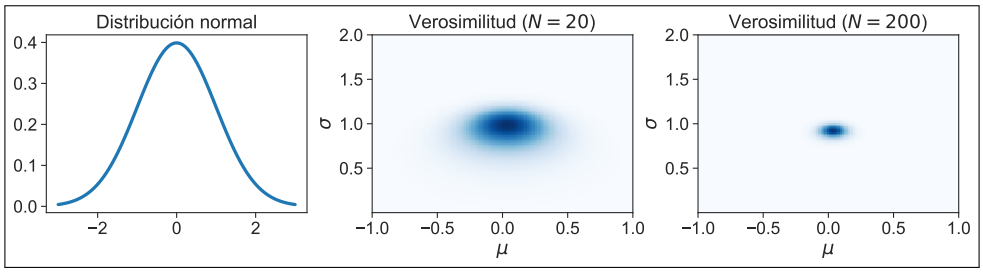


Fig. 8. Densidad normal (izquierda, $\mu = 0$ y $\sigma = 1$) y verosimilitud para la media y la varianza en base a 20 (centro) y 200 (derecha) observaciones.

La verosimilitud es fundamental cuando realizamos *inferencia*, es decir, cuando nuestro objetivo es descubrir o identificar los modelos y parámetros en base a las observaciones que dicho modelo ha generado; esto muchas veces se refiere coloquialmente como *probabilidad inversa*. La importancia de la función de verosimilitud está documentado en el **principio de la verosimilitud**, el cual sentencia que toda la información relevante que la observación $\mathbf{y} = \{y_i\}_{i=1}^N$ puede aportar a la estimación del parámetro θ, está contenido en la función de verosimilitud $L(\theta)$. Una consecuencia directa de este principio es que si diseñamos dos experimentos para realizar inferencia sobre un parámetro desconocido θ y ambos resultan en la misma función de verosimilitud (salvo una constante de proporcionalidad), entonces, ambos experimentos, y los datos adquiridos en ellos, reportan la misma información sobre θ. Lo de igualdad salvo una constante de proporcionalidad es porque recordemos que la verosimilitud es una medida *relativa* de la bondad de (cada valor del) parámetro a inferir.

La pregunta natural entonces es ¿cómo usar la función $L_{\mathbf{y}}(\theta)$ para determinar "el buen θ"? Por supuesto el título de esta sección hace las veces de *spoiler* para esta pregunta: simplemente elegir el máximo de la función, pues éste nos da el (o los) valor más *verosímil* para θ relativo a todo el resto de las opciones disponibles. Sin embargo, veamos que el uso del argumento que maximiza $L_{\mathbf{y}}(\theta)$ tiene un significado mucho más acabado. Consideremos la siguiente forma de encontrar el parámetro θ: recordemos que el modelo real es $p(y|\theta)$ y definamos una discrepancia entre modelos, denotada $D(p_1, p_2)$, luego, encontraremos el $\hat{\theta}$ tal que $p(y|\hat{\theta})$ es lo más *cercano* posible al modelo real con respecto a la discrepancia $D(\cdot, \cdot)$, es decir, el que minimiza la expresión

$$D(p(y|\theta), p(y|\hat{\theta})). \tag{42}$$

Este criterio es interesante, pues notemos que no hemos incorporado ningún supuesto sobre la parametrización de los modelos (cómo el modelo depende de θ) ni de la geometría del espacio Θ; estamos comparando directamente los modelos y no los valores específicos de los parámetros. Desafortunadamente, notemos que formular y resolver este problema no es posible en el caso general, pues la expresión de arriba depende del parámetro real θ, el cual no conocemos, con lo que no podríamos resolver dicho problema de optimización.

Sin embargo, veamos que podemos considerar una métrica que ofrece una alternativa para optimizar la discrepancia entre el modelo real y el aproximado independiente de que no conozcamos el valor de θ. Dicha métrica, la cual es motivada desde la teoría de la información, es un estándar para comparar distribuciones de probabilidad generales y es conocida como la divergencia de Kullback-Leibler $\mathrm{KL}(p, q)$. Evaluada entre el modelo real $p = p(y|\theta)$ y el aproximado $q = p(y|\hat{\theta})$, esta divergencia toma la siguiente forma

$$\mathrm{KL}(p(y|\theta), p(y|\hat{\theta})) = \int_y \log\left(\frac{p(y|\theta)}{p(y|\hat{\theta})}\right) p(y|\theta)\mathrm{d}y. \tag{43}$$

En general, no es claro que podamos calcular dicha integral, sin embargo, observemos que ésta es una esperanza con respecto a la densidad $p(y|\theta)$, por lo que podemos considerar su aproximación de Monte Carlo usando las N observaciones en D, las cuales están precisamente generadas por la medida de la integral en la ec. (43) de acuerdo a

$$\mathrm{KL}(p(y|\theta), p(y|\hat{\theta})) \approx \mathrm{KL}_N(p(y|\theta), p(y|\hat{\theta})) = \frac{1}{N}\sum_{i=1}^{N} \log\left(\frac{p(y_i|\theta)}{p(y_i|\hat{\theta})}\right). \tag{44}$$

Denotemos ahora $\hat{\theta}_N$ el minimizante de la expresión anterior, el cual podemos calcular mediante (ignoramos la constante N^{-1})

$$\begin{aligned}
\hat{\theta}_N &= \arg\min_{\hat{\theta}} \sum_{i=1}^{N} \log\left(\frac{p(y_i|\theta)}{p(y_i|\hat{\theta})}\right) \\
&= \arg\min_{\hat{\theta}} \sum_{i=1}^{N} \log p(y_i|\theta) - \sum_{i=1}^{N} \log p(y_i|\hat{\theta}) \\
&= \arg\max_{\hat{\theta}} \sum_{i=1}^{N} \log p(y_i|\hat{\theta}) \\
&= \arg\max_{\hat{\theta}} \prod_{i=1}^{N} p(y_i|\hat{\theta}),
\end{aligned} \tag{45}$$

donde hemos usado las propiedades del logaritmo y eliminado términos que no dependen de $\hat{\theta}$. Observemos que si nuestra muestras son condicionalmente independientes, entonces la expresión anterior implica que $\hat{\theta}_N$ es también el maximizante de la función de verosimilitud:

$$\hat{\theta}_N = \arg\max_{\hat{\theta}} \prod_{i=1}^{N} p(y_i|\hat{\theta}) = \arg\max_{\hat{\theta}} p(\mathbf{y}|\hat{\theta}) = \arg\max_{\hat{\theta}} L_{\mathbf{y}}(\theta), \tag{46}$$

al que nos referiremos como *estimador de máxima verosimilitud*. Finalmente, queda la pregunta de cómo se relacionan el estimador de máxima verosimilitud (MV) $\hat{\theta}_N$ con el estimador óptimo en el sentido KL, $\hat{\theta}$. Para esto, notemos que la aproximación de Monte Carlo de la KL en la ec. (43) converge puntualmente a la KL, i.e., para cada $\theta \in \Theta$, $\mathrm{KL}_N(p(y|\theta), p(y|\hat{\theta})) \to \mathrm{KL}(p(y|\theta), p(y|\hat{\theta}))$ por la ley de los grande números cuando $N \to \infty$. Consecuentemente, podemos asumir que los minimizantes de la secuencia de aproximaciones de Monte Carlo también convergen al minimizante de la KL. Esta es la razón por la cual consideramos el estimados de máxima verosimilitud: en el límite $N \to \infty$ el estimador de MV es el que reporta la mínima divergencia (KL) entre el modelo real y el aproximado. Esta condición nos da un sentido de *consistencia* del estimador de MV, donde por consistencia entendemos que mientras más datos observamos nuestra aproximación del modelo converge al mejor modelo posible (en la métrica KL).

Retomemos el problema de regresión lineal: la verosimilitud del modelo lineal gaussiano definido en la ec. (34) (con parámetro $\theta = [a, b, \sigma_\epsilon^2]$) está dada por (recordemos que los datos son condicionalmente independientes)

$$L_{\mathbf{y}}(\theta) = \prod_{i=1}^{N} \mathcal{N}(y_i; a^\top x_i + b, \sigma_\epsilon^2). \tag{47}$$

Usualmente, consideraremos el logaritmo de la verosimilitud, referido como *log-verosimilitud*, $l(\theta) = \log L(\theta)$, por su facilidad de interpretación y optimización. Recordemos que esta transformación es válida,

pues la función logaritmo es monótona y consecuentemente el (los) mínimo(s) no cambia(n). La log-verosimilitud del modelo lineal y gaussiano está dada por

$$l(\theta) = \underbrace{-N \log \sqrt{2\pi\sigma_\epsilon^2}}_{\text{dispersión}} + \underbrace{\frac{-1}{2\sigma_\epsilon^2} \sum_{i=1}^{N} (y_i - a^\top x_i - b)^2}_{\text{ajuste}}, \tag{48}$$

donde podemos de inmediato reconocer que la maximización de $l(\theta)$ implica el balance entre dos términos. El de la izquierda es una medida de dispersión o complejidad, pues para aumentar éste termino necesitamos que la varianza sea pequeña o el modelo tenga errores poco dispersos. El término de la derecha, por otro lado, es una medida de ajuste, para aumentar éste término necesitamos que el modelo represente bien, muestra a muestra, nuestros datos.

En particular, el estimador de máxima verosimilitud para los parámetros de la parte lineal (i.e., ignorando σ_ϵ^2) está dado por:

$$[a^{\text{MV}}, b^{\text{MV}}] = \arg\min_{a,b} \sum_{i=1}^{N} (y_i - a^\top x_i - b)^2. \tag{49}$$

Para nuestra sorpresa, observemos que es posible identificar esta última expresión con la del costo cuadrático en la ec.(9), es decir, el estimador de máxima verosimilitud es el minimizante del mismo costo que el estimador de mínimos cuadrados. Consecuentemente, ambos estimadores son iguales y de acuerdo a la ecuación (10) dados por

$$[a^{\text{MV}}, b^{\text{MV}}] = [a^{\text{MC}}, b^{\text{MC}}] = \left(\tilde{X}^\top \tilde{x}\right)^{-1} \tilde{X}^\top Y. \tag{50}$$

Además, recordemos que luego de determinar el estimador con criterio de MC, es posible calcular la varianza de los errores de nuestro modelo mediante

$$\text{Varianza} = \frac{1}{N} \sum_{i=1}^{N} (y_i - a^\top x_i - b)^2, \tag{51}$$

dicha cantidad es precisamente la suma de cuadrados e intuitivamente representa la bondad de ajuste del modelo considerado.

En el contexto de máxima verosimilitud, recordemos que la varianza es un parámetro del modelo y no una cantidad asociada al modelo que calculamos de forma independiente. Este parámetro puede ser calculado maximizando la log-verosimilitud, tal como se hizo para la media en la ecuación (49), de acuerdo a

$$\sigma_{\text{MV}}^2 = \arg\max \frac{N}{2} \log(\sigma_\epsilon^2) + \frac{1}{2\sigma_\epsilon^2} \sum_{i=1}^{N} (y_i - a^\top x_i - b)^2. \tag{52}$$

Usando la condición de primer orden en esta expresión, tenemos que

$$\frac{N}{2\sigma_{\text{MV}}^2} - \frac{1}{2\sigma_{\text{MV}}^4} \sum_{i=1}^{N} (y_i - a^\top x_i - b)^2 = 0 \Rightarrow \sigma_{\text{MV}}^2 = \frac{1}{N} \sum_{i=1}^{N} (y_i - a^\top x_i - b)^2. \tag{53}$$

Con lo cual se obtiene, sin sorpresa alguna, la misma expresión de la varianza que al usar mínimos cuadrados. ¡Observemos que el estimador de MV de la varianza depende de los parámetros a y b, pero no al revés!

En la práctica, consideraremos la minimización de la log-verosimilitud negativa (en vez de la maximización de la log-verosimilitud) en línea con la literatura y software dedicados a la minimización de funciones.

2.3. Regresión via inferencia bayesiana

En el apartado anterior vimos el ajuste de modelos, estimación de parámetros o *inferencia*, mediante la maximización de la función de verosimilitud. Es decir, en base a un conjunto de observaciones $\mathbf{y}$ asignamos el valor más *verosímil* al parámetro desconocido θ, dado por $\hat{\theta}^\star = \arg\max L_{\mathbf{y}}(\hat{\theta})$. Es claro por qué esta estimación puntual tiene sentido, desde el punto de vista de la probabilidad de los datos y de la mínima discrepancia contra el modelo real en base a la divergencia de Kullback-Leibler. Sin embargo, solo tomar el máximo ignora el resto de la información contenida en la función de verosimilitud, por ejemplo, si consideramos funciones de verosimilitud distintas, por ejemplo, bimodales, con colas pesadas/livianas, asimétricas, discretas, todas esa propiedades no se reflejan en la estimación del parámetro θ si estas verosimilitudes coinciden en el máximo. Esta es una limitación fundamental no solo del método de máxima verosimilitud, sino que de la estimación puntual en general.

Otra limitación del paradigma de máxima verosimilitud es que no da la posibilidad de incorporar conocimiento experto, es decir, introducir sesgos necesarios para la inferencia. Por ejemplo, si sabemos que el parámetro desconocido tiene magnitud pequeña, o está lejos del origen, o es ralo, etc. Esta propiedad es necesaria en el análisis moderno de datos, pues en la ciencia de datos los esfuerzos son colaborativos y el conocimiento experto sin duda ayuda a resolver problemas de forma eficiente y con interpretabilidad.

El paradigma bayesiano busca conciliar estas dos desventajas del uso de máxima verosimilitud interpretando el parámetro θ como variable aleatoria, donde la disponibilidad de datos se interpreta como un evento que aporta evidencia sobre el valor de θ. Consecuentemente, el proceso de inferencia ahora se centra en encontrar la distribución condicional $p(\theta|\text{datos})$.

2.3.1. ¿Qué es ser bayesiano?

Thomas Bayes (c. 1701-1761) fue un matemático, filósofo y pastor presbiteriano inglés interesado en el cálculo y uso de las probabilidades. En ese entonces, no existía la diferencia entre probabilidad descriptiva (que caracteriza la generación de datos dado un modelo) e inferencial (que nos permite identificar un modelo dado un conjunto de observaciones). Como vimos en la clase pasada, el uso de probabilidades para estimar (o *inferir*) parámetros usando los propios datos recibía el nombre de *probabilidad inversa*. Tanto Bayes, como el matemático francés Pierre Simon Laplace, estaba al tanto de la siguiente relación para el problema de inferencia, la cual hoy conocemos como el *Teorema de Bayes*:

$$p(\theta|x) = \frac{p(x|\theta)p(\theta)}{p(x)} \propto p(x|\theta)p(\theta), \tag{54}$$

donde $x \in \mathcal{X}$ es el valor de una observación y $\theta \in \Theta$ es un parámetro. En la expresión anterior podemos identificar las siguientes cantidades:

- el prior o distribución a priori: $p(\theta)$,
- la verosimilitud: $p(x|\theta)$,
- la distribución posterior: $p(\theta|x)$,
- la densidad marginal de x: $p(x) = \int_\Theta p(x|\theta)p(\theta)\mathrm{d}\theta$.

Recordemos que en el problema de inferencia, los datos (en este caso x) son conocidos y fijos, mientras que el parámetro es variable (en realidad desconocido). Por esta razón, podemos escribir el lado derecho de la ec. (54), pues nos importa la distribución posterior como función de θ únicamente, tal como fue el caso de la función de verosimilitud en el apartado anterior. En algunas aplicaciones, conocer la versión proporcional de la posterior dada por $p(x|\theta)p(\theta)$ es suficiente para realizar distintos análisis, por ejemplo, cuando usamos Markov chain Monte Carlo.

Bajo cualquier punto de vista de inferencia estadística, e.g., bayesiano u otro, el espacio que contiene los valores de x debe tener suficiente *estructura* para definir probabilidades (o, equivalentemente, modelos);

en particular aquellos de la forma $p(x)$ y $p(x|\theta)$. Adicionalmente, una característica particular del enfoque bayesiano es que asume que el espacio donde donde se encuentra el parámetro θ, denotado Θ, también debe ser un espacio de probabilidad. El enfoque frecuentista, por el contrario, tiene un concepto de probabilidad totalmente distinto que resulta en la consideración de θ como un elemento fijo, como un índice o hipótesis cuyo valor queremos descubrir.

La inferencia bayesiana se sustenta en la noción de probabilidad como medida de incertidumbre, es decir, en una perspectiva subjetiva del conocimiento disponible sobre la generación del valor x. En este sentido, es posible identificar dos tipos de incertidumbre, la primera es la incertidumbre **aleatoria** y dice relación con la variabilidad con la que el sistema/modelo en cuestión genera el dato x. El segundo tipo es la llamada incertidumbre **epistemológica** y representa nuestra inhabilidad de conocer el modelo que genera los datos. Consecuentemente, el enfoque bayesiano hace la distinción entre ambos tipos de incertidumbre mediante los siguiente elementos:

- La verosimilitud $p(x|\theta)$: que modela la aleatoriedad del modelo, el cual produce datos de forma aleatoria incluso cuando el modelo es perfectamente conocido. Este tipo de incertidumbre no puede ser reducida observando datos. **Ejemplo:** En una urna con bolitas rojas y negras donde la probabilidad de elegir una bolita negra es θ, incluso si θ fuese conocido, la incertidumbre aleatoria del modelo resulta en la imposibilidad de predecir el color de la próxima bolita.

- La distribución a priori $p(\theta)$: que encapsula la incertidumbre epistemológica (lo que no sabemos) sobre el sistema, la cual pude ser reducida observando datos y calculando $p(\theta|x)$ mediante el Teorema de Bayes. Distintas distribuciones priori llevarán a distintas distribuciones a posteriori, lo cual establece la subjetividad del enfoque bayesiano. **Ejemplo:** En la misma urna anterior, el no conocer θ es un ejemplo de incertidumbre epistemológica.

Una discusión trascendental desde los inicios del enfoque bayesiano apuntaba a cómo elegir la distribución a priori $p(\theta)$ para ser lo más objetivo posible, es decir, cómo no introducir sesgos en la inferencia (cálculo de la posterior) heredados de una elección subjetiva del prior—esta era el objetivo de los llamados *bayesianos objetivistas*. La postura de Laplace con respecto de esta disyuntiva fue elegir simplemente un prior *no-informativo*, i.e., uno que no asigne más probabilidad a ningún valor de θ en particular, para lo cual se puede elegir un prior uniforme $p(\theta) \propto 1$. Sin embargo, posteriormente se descubrió que esta elección introduce sesgo en la inferencia de todas formas.[6] Hacia fines del sigo XIX comenzó a surgir una demanda por un tratamiento i) objetivo de la inferencia que no dependiese de la elección del prior, algo que los *bayesianos objetivistas* no habían garantizado, y ii) riguroso y formal en el sentido de que fuese consistente con la teoría matemática. En este sentido, durante el siglo XX la inferencia estadística vio avances en la dirección de prescindir del uso del prior para evitar inferencias sesgadas, en particular, el foco estuvo en la maximización directa de la verosimilitud con respecto al *índice* θ, tal como vimos en la clase pasada. De forma más general, y a través de Fisher, Neyman y Pearson, fue posible construir un tratamiento formal de la inferencia estadística sobre la bases de una interpretación frecuentista de la probabilidad, esto es, definir la probabilidad de un evento como el límite de la frecuencia de *casos favorables divido por casos totales* cuando el número de realizaciones de un experimento tiende a infinito. Entonces, al considerar los parámetros como cantidades fijas y desconocidas, y al mismo tiempo, concebir la probabilidad como una medida de frecuencias y no de incertidumbre, el punto de vista frecuentista no busca asociar probabilidades a los parámetros desconocidos.

En la segunda mitad del siglo XX hubo un resurgimiento de la inferencia Bayesiana, en parte por el descontento de algunos estadísticos (y científicos en general) con las métricas frecuentistas estándar como los errores de Tipo-I & Tipo-II, y los los llamados p-valores. Esto permitió utilizar la máquinaria desarrollada por la perspectiva frecuentista para formalizar el enfoque bayesiano, en particular, esta combinación permitió construir priors no informativos invariantes bajo transformaciones una-a-uno (e.g.,

[6] Esto es consecuencia de que el prior uniforme no es invariante bajo transformaciones uno-a-uno de θ, consecuentemente, si el modelo es re-parametizado de forma no lineal y se mantiene el prior uniforme, entonces la posterior cambia.

el prior de Jeffreys), como también definir el concepto de *consistencia*, i.e., si un estimador converge a la respuesta correcta y a qué velocidad ocurre esto. Esta unión entre los conceptos bayesianos y frecuentistas resulta en lo mejor de dos mundos: hace posible construir un modelo que es subjetivo, pues en la práctica queremos incorporar conocimiento experto en los problemas que estemos estudiando, pero al mismo tiempo tenemos una herramienta objetiva (pues el enfoque frecuentista es *automático* en el sentido que no requiere conocimiento experto) para asegurar que nuestros métodos son rigurosos desde una perspectiva matemática.

¿Qué hizo efectivamente Bayes y por qué?

Thomas Bayes fue uno de los primeros *inconformistas anglicanos.*[a] Su trabajo de 1763, titulado

An Essay Towards Solving a Problem in the Doctrine of Chances

esbozó por primera vez el resultado que hoy conocemos como el Teorema de Bayes. Este trabajo fue terminado por el amigo y colega de Bayes, Richard Price (1723 – 1791), el cual envió el artículo a la prestigiosa revista inglesa *Philosophical Transactions of the Royal Society* (PTRS), donde fue publicado póstumamente. Este artículo consideraba el caso de *invertir la distribución binomial*: en un experimento donde el resultado puede ser éxito (con probabilidad p) o fracaso (con probabilidad $1-p$), el objetivo es encontrar la distribución del parámetro p dado que se han observado q éxitos luego de n experimentos. Para resolver este caso particular, Bayes descubrió que efectivamente la distribución posterior es proporcional a la verosimilitud. Sin embargo, a pesar de que Price completara el trabajo de Bayes luego de la repentina muerte de éste último, este trascendental descubrimiento quedó momentáneamente en el olvido. Fue finalmente Laplace quien en 1774 publicó la relación entre las distribuciones prior y posterior como la conocemos hoy, sin dejar de reconocer el trabajo de Bayes. En efecto, en su *Essai Philosophique dur les Probabilites* (1814), Laplace menciona que Bayes ya había llegado al mismo resultado de forma "refinada y muy ingeniosa, aunque un poco confusa".

Existen dudas sobre la autoría de Bayes del artículo en cuestión y cuánto de éste efectivamente venía de las propias notas no publicadas de Bayes, y cuánto de la edición y cálculos de R. Price (Bellhouse, 2004). En efecto, al año siguiente del artículo original, Price publicó en 1764 otro artículo en PTRS llamado *A Demonstration of the Second Rule in the Essay toward the Solution of a Problem in the Doctrine of Chances*, esta vez de su propia autoría. Tanto Bayes como Price estaban muy lejos de ser matemáticos prolíficos de su época, de hecho, las publicaciones de ambos no están particularmente ligadas a las matemáticas sino que son de índole religiosa—por mucho que sus contribuciones hayan sido suficientemente conceptuales y generales para ser consideradas como avances en ambos campos. Una posibilidad para entender las motivaciones de Bayes (y Price) para el desarrollo de los artículos mencionados, pueden encontrarse en el artículo (Stigler, 2013) el cual postula que el verdadero título del artículo de Bayes aparentemente es:

A Method of Calculating the Exact Probability of All Conclusions founded on Induction.

Este título, según explica (Stigler, 2013), pudo haber sido perdido en la edición del artículo o probablemente el propio editor de PTRS lo modificó por encontrarlo un tanto *atrevido*. Si este fuese efectivamente el título, y no el menos informativo título original, se entiende que el artículo tiene por objetivo caracterizar el problema más general de *inducción*. Esto tiene un sentido particular dado el contexto filosófico y religioso en ese entonces, pues pocos años antes la obra de Hume 'Of Miracles' (1748) presentaba un argumento probabilístico para desestimar los milagros (como la resurrección). En pocas palabras, Hume postulaba que la considerable *improbabilidad* de un

milagro sobrepasaba ampliamente la *probabilidad* de que el milagro fuese incorrectamente documentado. Este ensayo de Hume fue ampliamente leído, discutido y—evidentemente—atacado, en particular, tanto Bayes como Price, ambos inconformistas anglicanos, consideraron el ensayo de Hume como una agresión. Consecuentemente, Bayes consideró responder al argumento de Hume mediante la aplicación de probabilidades al problema de inducción, lo cual es confirmado por las notas no publicadas de Bayes con fecha anterior al 31 de diciembre de 1749 (Bellhouse, 2004), las que eventualmente llegaron al famoso artículo de 1763. Sin embargo, el trabajo de Bayes no estaba completo, en particular, no contaba una satisfactoria aproximación de la distribución posterior en el ejemplo de la inversión de la distribución binomial mencionado anteriormente. En este sentido, la motivación de Price para terminar este trabajo no era únicamente concluir el trabajo póstumo de su amigo, sino que también desarrollar un respuesta eficaz contra el argumento de Hume. En efecto, luego de una serie de trabajos relacionados, fue finalmente en 1767 que Price logró publicar su disertación *On the Importance of Christianity, its Evidences, and the Objections which have been made to it*, donde hace una crítica directa a "Of miracles". Básicamente, Price sentencia que Hume habría subestimado el impacto que una (gran) cantidad de observadores **independientes** reportando la existencia del milagro puede tener. En dicho caso, el Teorema de Bayes mostraba que la multiplicación de evidencia, la cual puede ser falible de forma individual, podría sobrepasar la improbabilidad de un evento (el milagro) y establecerlo como verdad (o en realidad muy probable).

Existen varias razones atribuibles a que el trabajo de Bayes haya sido ignorado parcialmente hasta Laplace, e.g., la incompletitud del trabajo del propio Bayes, el rol incierto que tuvo Price en el desarrollo de éste, y el nombre poco elocuente del artículo que probablemente desvió atención al trabajo. Todo esto nos hace cuestionarnos si es correcto referirnos a este trascendental Teorema en honor al Rev. Thomas Bayes, el cual lanzó el primer atisbo de este resultado y no a Laplace, el cual fue el primero en enunciar la forma general para la relación entre prior, posterior y modelo en la forma que hoy conocemos y usamos.

[a] Este término se usa para denotar a los protestantes que no estaban de acuerdo con los métodos y la gobernanza eclesiástica establecida por la "Iglesia de Inglaterra" (The Church of England).

2.3.2. Elección de prior: conjugación

Desde ahora, nuestra misión calcular y analizar la distribución posterior del parámetro θ dado un conjunto de datos D, nos referimos a esto como *inferencia bayesiana* a diferencia de las más general *inferencia estadística* en la cual a veces ses consideran solo estimaciones puntuales de θ.

La distribución posterior está dada, mediante el teorema de Bayes, por

$$p(\theta|D) = \frac{p(D|\theta)p(\theta)}{p(D)} \tag{55}$$

donde $p(D|\theta)$ es la verosimilitud o el modelo que elegimos y $p(\theta)$ es la *distribución a priori* del parámetro y encapsula todos nuestros supuestos, creencias y sesgo sobre el espacio de parámetros (modelos) a considerar. Finalmente, recordemos que en el denominador encontramos la *distribución marginal de los datos* $p(D)$ que actúa como constante de normalización para el numerador, pues el numerador puede ser considerado como una densidad pero no de probabilidad. Esta constante de normalización puede ser calculada mediante el uso de la ley de probabilidades totales, o bien imponiendo la restricción de que la expresión en la ecuación (55) debe integrar uno:

$$p(D) = \int p(D|\theta)p(\theta)d\theta. \tag{56}$$

En base a la forma explícita de $p(D|\theta)p(\theta)$, calcular esta integral puede ser un desafío considerable. Sin embargo, enfatizamos que como esta cantidad no depende del parámetro θ, no es necesario conocerla para explorar o aproximar la (forma de la) distribución posterior $p(\theta|D)$.

Entonces, con el modelo $p(D|\theta)$ acordado, solo nos queda elegir la distribución a priori, los cual es guiado por dos objetivos. En primer lugar, debemos encapsular lo que efectivamente que sabemos del parámetro θ, por ejemplo, en el caso de regresión lineal, podemos tener evidencia que los datos que observamos representan una cantidad que creciente en el tiempo, en cuyo caso, sabemos que $\theta \geq 0$. El segundo objetivo es obtener una forma *amigable* de la distribución posterior, en el sentido que ésta sea un distribución con propiedades que deseemos, en particular, que la podamos calcular, evaluar, y samplear de ella. Una práctica usual en este sentido es elegir un prior $p(\theta)$ tal que la distribución posterior $p(D|\theta)$ están en la misma familia.

Definición 2.2. Diremos que el prior $p(\theta)$ es *conjugado* a la verosimilitud $p(D)$, cuando la posterior $p(\theta|D)$ está en la misma familia, es decir, tienen la misma distribución con parámetros distintos.

El uso de un prior arbitrario resulta en que la posterior tenga una forma arbitraria también, con lo que incluso si tanto el prior como la verosimilitud tienen formas *conocidas*, no tenemos ninguna garantía de que el posterior también la tenga y consecuentemente sea difícil de interpretar. Por el contrario, si usamos priors conjugados la distribución posterior pertenece a *en la misma familia* del prior cuando vamos observando más datos, lo cual tiene un significado relevante: la actualización de prior a posterior mediante la incorporación de datos—conocida como actualización bayesiana—es simplemente un cambio de parámetros, lo cual ofrece una clara interpretación (en el caso que los parámetros tengan significado como media, varianza, o alguna taza), y la nueva distribución ocupan la misma cantidad de memoria que el prior (pues la cantidad de parámetros no cambia). A continuación veremos dos ejemplos de priors conjugados y cómo se interpreta la variación que sufren los parámetros de la posterior al incorporar datos.

Modelo gaussiano. Consideremos un conjunto de observaciones[7] $\mathcal{D} = \{x_i\}_{i=1}^n \subset \mathbb{R}$ generadas independiente e idénticamente distribuidas (iid) por el modelo $\mathcal{N}(\mu, \sigma^2)$. Recordemos que la verosimilitud de la media y varianza respectivamente está dada por

$$L_{\mathcal{D}}(\mu, \sigma^2) = \prod_{i=1}^{n} \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{1}{2\sigma^2}(x_i - \mu)^2\right). \tag{57}$$

A continuación veremos priors conjugados para esta verosimilitud de forma incremental: primero cuando sola la media μ es desconocida, luego cuando solo la varianza σ^2 es desconocida y finalmente cuando ambos parámetros son desconocidos.

Modelo gaussiano (σ^2 conocido). Consideremos el prior sobre la media $p(\mu) = \mathcal{N}(\mu_0, \sigma_0^2)$, con lo que la posterior está dada por

$$p(\mu|\mathcal{D}) \propto \prod_{i=1}^{n} \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{1}{2\sigma^2}(x_i - \mu)^2\right) \frac{1}{\sqrt{2\pi\sigma_0^2}} \exp\left(-\frac{1}{2\sigma_0^2}(\mu - \mu_0)^2\right) \tag{58}$$

$$\propto \exp\left(-\frac{1}{2\sigma^2}\sum_{i=1}^{n}(x_i - \mu)^2 - \frac{1}{2\sigma_0^2}(\mu - \mu_0)^2\right), \tag{59}$$

donde la proporcionalidad viene de ignorar la constante $p(\mathcal{D})$ en la primera línea e ignorar todas las contantes que no dependen de μ en la segunda línea; recordemos que estas constantes para μ incluyen a la varianza de x, σ^2, por lo que ignorar esta cantidad es solo posible debido a que estamos considerando el caso en que σ^2 es conocido. Completando el la forma cuadrática para μ dentro de la exponencial en la ec. (59), obtenemos

$$p(\mu|\mathcal{D}) \propto \exp\left(-\frac{1}{2\sigma_n^2}(\mu - \mu_n)^2\right), \tag{60}$$

[7]Observe que en esta sección no estamos solo enfocados en el problema de regresión, sino que cualquiera que requiera inferencia paramétrica bayesiana.

donde (ya definiremos μ_n y σ_n^2 en breve) como $p(\mu|\mathcal{D})$ debe integrar uno, la única densidad de probabilidad proporcional al lado derecho de la ecuación anterior es la Gaussiana de media μ_n y varianza σ_n^2. Es decir, la constante de proporcionalidad necesaria para la igualdad en la expresión anterior es $\int_{\mathbb{R}} \exp\left(-\frac{1}{2\sigma_n^2}(\mu-\mu_n)^2\right) \mathrm{d}\mu = (2\pi\sigma_n^2)^{n/2}$. Consecuentemente, confirmamos que el prior elegido era efectivamente conjugado con la verosimilitud gaussiana y la posterior está dada por la siguiente gaussiana:

$$p(\mu|\mathcal{D}) = \mathcal{N}(\mu; \mu_n, \sigma_n^2) = \frac{1}{(2\pi\sigma_n^2)^{N/2}} \exp\left(-\frac{1}{2\sigma_n^2}(\mu-\mu_n)^2\right), \tag{61}$$

donde la media y la varianza están dadas respectivamente por

$$\mu_n = \frac{1}{\frac{1}{\sigma_0^2}+\frac{n}{\sigma^2}} \left(\frac{1}{\sigma_0^2}\mu_0 + \frac{n}{\sigma^2}\bar{x}\right), \quad \text{donde } \bar{x} = \frac{1}{n}\sum_{i=1}^{n} x_i \tag{62}$$

$$\sigma_n = \left(\frac{1}{\sigma_0^2}+\frac{n}{\sigma^2}\right)^{-1}. \tag{63}$$

Observación 2.1. La actualización bayesiana transforma los parámetros del prior de μ desde μ_0 y σ_0^2 hacia μ_n y σ_n^2 en las ecs. (62) y (63) respectivamente. Notemos que los parámetros de la posterior son combinaciones (interpretables por lo demás) entre los parámetros del prior y los datos, en efecto, la μ_n es el promedio ponderado entre μ_0 (que es nuestro candidato para μ antes de ver datos) con factor σ_0^{-2} y el promedio de los datos $\bar{x}$ con factor $(\sigma^2/n)^{-1}$, que a su vez es el estimador de máxima verosimilitud. Es importante también notar que estos factores son las varianzas inversas—i.e., precisión—de μ_0 y de $\bar{x}$. Finalmente, observemos que σ_n es la *suma paralela* de las varianzas, pues si expresamos la ec. (63) en términos de *precisiones*, vemos que la precisión inicial σ_0^2 aumenta un término σ^2 con cada dato que vemos; lo cual tiene sentido pues con más información es la precisión la que debe aumentar y no la incertidumbre (en este caso representada por la varianza).

Modelo gaussiano (μ conocido). Ahora procedemos con el siguiente prior para la varianza, llamado Gamma-inverso:

$$p(\sigma^2) = \text{inv-}\Gamma(\sigma^2; \alpha, \beta) = \frac{\beta^\alpha}{\Gamma(\alpha)(\sigma^2)^{\alpha+1}} \exp(-\beta/\sigma^2) \tag{64}$$

esta densidad recibe dicho nombre pues es equivalente a modelar la precisión, definida como el recíproco de la varianza $1/\sigma^2$, mediante la distribución Gamma. Los hiperparámetros α y β son conocidos como parámetros de forma y de tasa (o precisión) respectivamente.

Con este prior, la posterior de la varianza toma la forma:

$$\begin{aligned} p(\sigma^2|\mathcal{D}) &\propto \prod_{i=1}^{N} \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{1}{2\sigma^2}(x_i-\mu)^2\right) \frac{\beta^\alpha}{\Gamma(\alpha)(\sigma^2)^{\alpha+1}} \exp(-\beta/\sigma^2) \\ &\propto \frac{1}{(\sigma^2)^{N/2+\alpha+1}} \exp\left(-\frac{1}{\sigma^2}\left(\frac{1}{2}\sum_{i=1}^{N}(x_i-\mu)^2 + \beta\right)\right) \end{aligned} \tag{65}$$

donde nuevamente la proporcionalidad ha sido mantenida debido a la remoción de las constantes. Esta última expresión es proporcional a una distribución Gamma inversa con hiperparámetros α y β ajustados en base a los datos observados.

Modelo gaussiano (μ y σ^2 desconocidos). Tarea.

Ejemplo: Distribución posterior modelo lineal y gaussiano

Recordemos que, para observaciones $\mathcal{D} = \{(x_i, y_i)\}_{i=1}^n$, el modelo de regresión lineal puede ser

escrito en forma vectorial mediante

$$Y = \tilde{X}\theta + \boldsymbol{\epsilon}, \tag{66}$$

donde $[\tilde{X}]_{i:} = [x_i^\top, 1]$, $[Y]_i = y_i$, $[\boldsymbol{\epsilon}]_i = \epsilon_i \sim \mathcal{N}(0, \sigma^2)$, $\theta = [a; b]$. Con lo que su verosimilitud está dada por

$$\begin{aligned} L(\theta, \sigma^2) &= \text{MVN}(Y; \tilde{X}\theta, \mathbb{I}\sigma^2) \\ &\propto \left(\sigma^2\right)^{-n/2} \exp\left(-\frac{1}{2\sigma^2}(Y - \tilde{X}\theta)^\top (Y - \tilde{X}\theta)\right), \end{aligned} \tag{67}$$

donde la distribución MVN denota la normal multivariada. Observe que esta última expresión es proporcional a una distribución Gamma-Inversa para σ^2 y proporcional a una MVN para θ. Consecuentemente, esta verosmilitud tiene los mismos priors conjugados que el modelo gaussiano en la ec. (40). Consideremos entonces el caso en que σ^2 es conocido y elegimos el prior gaussiano para θ dado por

$$p(\theta) \propto \exp\left(-\frac{1}{2\sigma^2}(\theta - \theta_0)^\top \Lambda_0 (\theta - \theta_0)\right), \tag{68}$$

obtenemos una distribución posterior dada por $\text{MVN}(\theta; \theta_n, \sigma^2\Lambda_n^{-1})$ con parámetros

$$\theta_n = (\tilde{X}^\top \tilde{X} + \Lambda_0)^{-1}(\tilde{X}^\top Y + \Lambda_0\theta_0) = (\tilde{X}^\top \tilde{X} + \Lambda_0)^{-1}(\tilde{X}^\top \tilde{X}(\tilde{X}^\top \tilde{X})^{-1}\tilde{X}^\top Y + \Lambda_0\theta_0) \tag{69}$$

$$\Lambda_n = (\tilde{X}^\top \tilde{X} + \Lambda_0). \tag{70}$$

Es decir, la media posterior θ_n es un promedio ponderado entre la media a priori θ_0 y el estimador de máxima verosimilitud $(\tilde{X}^\top \tilde{X})^{-1}\tilde{X}^\top Y$. Observemos además cómo la varianza Λ_n se mueve desde la varianza a priori Λ_0 hacia $(\tilde{X}^\top \tilde{X})^{-1}$ a medida recibimos más observaciones, resultando en un modelo más preciso.

La Figura 9 muestra una implementación de la regresión lineal bayesiana, para el modelo

$$y = ax + b + \epsilon = 2x - 2 + \epsilon,$$

donde $\epsilon \sim \mathcal{N}(0, 0.5^2)$. Desde arriba hacia abajo, se han considerado $\{0, 1, 2, 50\}$ observaciones, donde cada columna de la figura muestra (de izquierda a derecha): los datos observados, la distribución conjunta de los parámetros a y b, las distribuciones marginales de los parámetros, y el modelo real (azul) junto a muestras del modelo posterior (rojo). Observe cómo rápidamente las distribuciones posteriores se concentran en los valores de los parámetros reales.

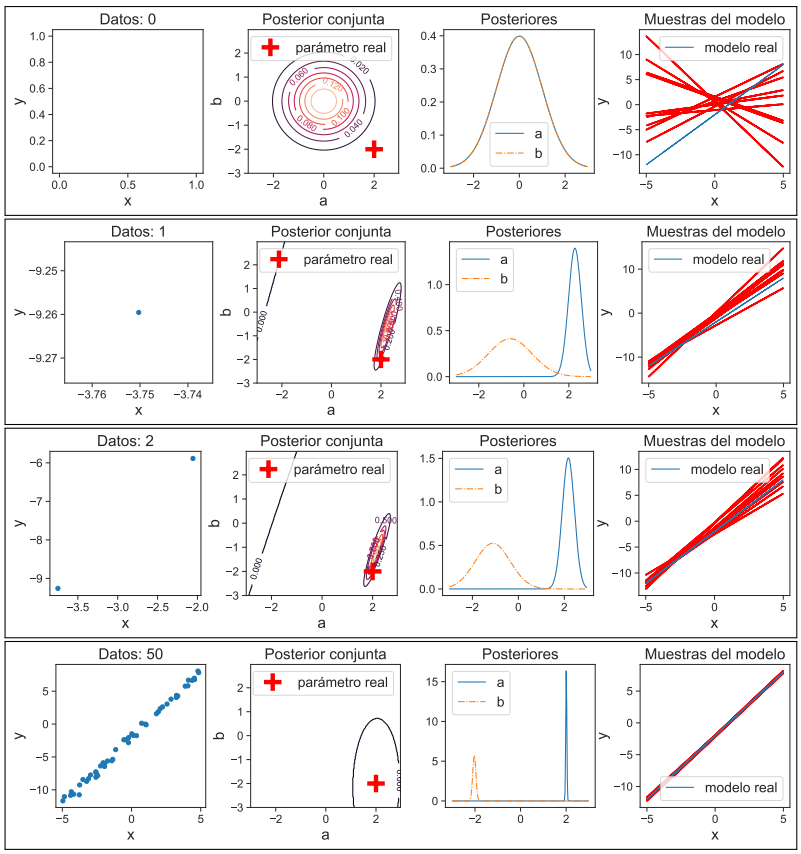


Fig. 9. Regresión lineal bayesiana

2.3.3. Modelo binomial

Ahora ilustraremos la elección de la distribución a priori a través de un segundo ejemplo basado en el modelo binomial, el cual fue considerado en el artículo original de (Bayes, 1763). Al comienzo de su artículo, Bayes enuncia el problema que motiva su trabajo:

Given the number of times in which an unknown event has happened and failed: Required the chance that the probability of its (specific event) happening in a single trial lies somewhere between any two degrees of probability that can be named.

A pesar de lo confuso de la jerga usada por Bayes, al menos ya podemos notar que se hace una diferencia entre la probabilidad de los datos (*probability*) y la del modelo (*chance*). Si tradujéramos esta formulación del problema al español y con una notación moderna, diríamos:

Dados n lanzamientos Bernoulli iid con parámetro θ donde se han registrado k aciertos, ¿cuál es la probabilidad de que el parámetro θ se encuentre entre dos cotas dadas?

Consideremos entonces el evento de obtener "k aciertos en n intentos". Por ejemplo anotar k goles con n intentos de penales, u obtener k veces un número par al lanzar un dado n veces. La probabilidad de obtener entonces los "k aciertos en n intentos" puede ser modelada mediante una distribución binomial, la cual asume que cada acierto es independiente y equiprobable con probabilidad θ (o bien, *Bernoulli*). La verosimilitud de este modelo (con parámetro θ) está dada por la distribución binomial de los datos (n lanzamientos y k aciertos) dada por

$$p(k|n, \theta) = \binom{n}{k} \theta^k (1 - \theta)^{n-k}. \tag{71}$$

Antes de proceder con el prior conjugado para esta verosimilitud, revisemos qué hizo T. Bayes. Él sabía que la verosimilitud era proporcional a la verosimilitud porque consideró—implícitamente—un prior **uniforme** para $\theta \in [0,1]$ (el caso binomial fue el único considerado por Bayes). La elección del prior uniforme tiene sentido, pues θ es una probabilidad de la cual, *a priori*, sabemos nada. Consecuentemente, como solo conocemos una versión proporcional de la posterior a través del Teorema de Bayes, la posterior es necesariamente (recordemos que consideramos un prior uniforme)

$$p(\theta|n,k) = \frac{p(n,k|\theta)}{\int_0^1 p(n,k|\theta)\mathrm{d}\theta} = \frac{\theta^k(1-\theta)^{n-k}}{\int_0^1 \theta^k(1-\theta)^{n-k}\mathrm{d}\theta} = \frac{\theta^k(1-\theta)^{n-k}}{\mathcal{B}(k+1,n-k+1)}, \tag{72}$$

donde $\mathcal{B}(\alpha,\beta) = \int_0^1 x^{\alpha-1}(1-x)^{\beta-1}\mathrm{d}x$ es conocida como la función Beta.

Este resultado era al que Bayes llegó en su artículo, aunque no de forma explícita. Sin embargo, al no entregar una forma cerrada para esta probabilidad, el trabajo de Bayes era difícil de implementar. Actualmente, el cálculo de esta constante de proporcionalidad no es problemático[8] pues sabemos que éstas son densidades de probabilidad que integran 1, un concepto que no existía en la era de Bayes.

A simple vista, el prior uniforme propuesto por Bayes no es conjugado, pues la posterior no es uniforme. Sin embargo, al darle una mirada más detallada, podemos identificar que éste sí es un caso particular de un prior conjugado. En efecto, consideremos el prior dado por una distribución Beta para θ, con parámetros (α,β), con densidad dada por

$$p(\theta) = \mathrm{Beta}(\theta;\alpha,\beta) = \frac{\theta^{\alpha-1}(1-\theta)^{\beta-1}}{\mathcal{B}(\alpha,\beta)},\ \theta \in [0,1], \tag{73}$$

donde la función Beta definida anteriormente actúa como contante de normalización. Notemos que eligiendo $\alpha = \beta = 1$, obtenemos que este prior es efectivamente la densidad uniforme entre 0 y 1. Además, observemos el rol de los parámetros de la distribución Beta, la cual está formada por la multiplicación de dos potencias de θ con ceros en $\theta = 0$ y $\theta = 1$, donde α y β representan, informalmente, los pesos relativos entre de los aciertos y fallos respectivamente. De hecho, $\mathbb{E}(\theta|\alpha,\beta) = \frac{\alpha}{\alpha+\beta}$.

Para calcular la posterior que resulta del uso del prior Beta, veamos un caso un poco más general que el anterior. Consideremos las observaciones $\mathcal{D} = \{(n_i,k_i)\}_{i=1}^N$ correspondientes a N juegos (no solo uno como el caso anterior), donde el i-ésimo juego consistió en n_i intentos y k_i aciertos. Con estos datos y el prior $\mathrm{Beta}(\theta;\alpha,\beta)$ de la ec. (73), la (versión proporcional de la) distribución posterior de θ está dada por

$$\begin{aligned} p(\theta|\mathcal{D}) &\propto \prod_{i=1}^N p(k_i|n_i,\theta)p(\theta) \\ &\propto \prod_{i=1}^N \binom{n_i}{k_i}\theta^{k_i}(1-\theta)^{n_i-k_i}\theta^{\alpha-1}(1-\theta)^{\beta-1} \\ &\propto \theta^{\sum k_i+\alpha-1}(1-\theta)^{\sum(n_i-k_i)+\beta-1}, \end{aligned} \tag{74}$$

donde los símbolos de proporcionalidad se han mantenido debido a la remoción de constantes y hemos usado la notación compacta $\sum k_i = \sum_{i=1}^N k_i$. Notemos que la última expresión es proporcional a la definición de distribución Beta en la ecuación (73), por lo que ajustando la constante de proporcionalidad tenemos que, al igual que el prior, la posterior es Beta también:

$$p(\theta|\mathcal{D}) = \mathrm{Beta}(\theta;\textstyle\sum_{i=1}^N k_i + \alpha, \sum_{i=1}^N (n_i - k_i) + \beta). \tag{75}$$

[8]En particular conocemos la relación entre la función Beta y la función Gamma, $\mathcal{B}(\alpha,\beta) = \frac{\Gamma(\alpha)\Gamma(\beta)}{\Gamma(\alpha+\beta)}$, que facilitan el cálculo de la función Beta.

Vemos entonces explícitamente cómo la distribución posterior es nuevamente una *mezcla* entre el prior y la verosimilitud en función de los parámetros, los cuales de prior a posterior han cambiado:

$$\alpha \to \alpha + \sum_{i=1}^{N} k_i \tag{76}$$

$$\beta \to \beta + \sum_{i=1}^{N} (n_i - k_i), \tag{77}$$

lo cual es consistente con la interpretación inicial de los parámetros de la distribución Beta: el peso de los aciertos (α) se modifica en función de la suma de todos los aciertos y el peso de los fracasos (β) se modifica en función de la suma de todos los fracasos. Adicionalmente, notemos que la razón entre los parámetros (la cual convergerá cuando el número de observaciones tiende a infinito), la magnitud de éstos va aumentando: la consecuencia de esto es que la varianza de la posterior va disminuyendo, pues

$$\mathbb{V}(\theta|\alpha,\beta) = \frac{\alpha\beta}{(\alpha+\beta)^2(\alpha+\beta+1)}, \tag{78}$$

con lo que la incertidumbre de la distribución posterior de θ va disminuyendo a medida que vemos más observaciones.

Ejemplo: Modelo binomial

Apliquemos la actualización bayesiana en el modelo binomial. Consideremos la variable binomial k dada por

$$k \sim \binom{n}{k} \theta^k (1-\theta)^{n-k}, \tag{79}$$

con probabilidad $\theta = 0.66$ (desconocida). Nuestro objetivo es ver cómo depende la distribución posterior de θ de la elección del prior a medida que vamos obteniendo más observaciones. Para esto, elegimos 20 priors Beta distintos para θ dados por

$$p_i(\theta) = \text{Beta}(\alpha_i, \beta_i), \;\; i = 1, \ldots, 20, \tag{80}$$

donde $\alpha_i, \beta_i \sim U(1, 10)$ (iid y discreto).

Ahora, realizaremos tres experimentos, en los cuales elegimos $n \in \{10, 100, 1000\}$ lanzamientos y calculamos los parámetros de las distribuciones a posteriori correspondientes a cada prior de acuerdo a las ecs. (76)-(77). En la Figura 10 podemos ver, desde izquierda a dercha, las distribuciones a priori elegidas y luego las posteriores correspondientes a 10, 100 y 1000 observaciones. Notemos cómo a medida obtenemos más y más datos, la elección del prior es vuelve irrelevante y, como es de esperar, la evidencia de los datos es suficiente para aprender el modelo exitosamente (pues $\theta = 0.66$) sin depender del prior.

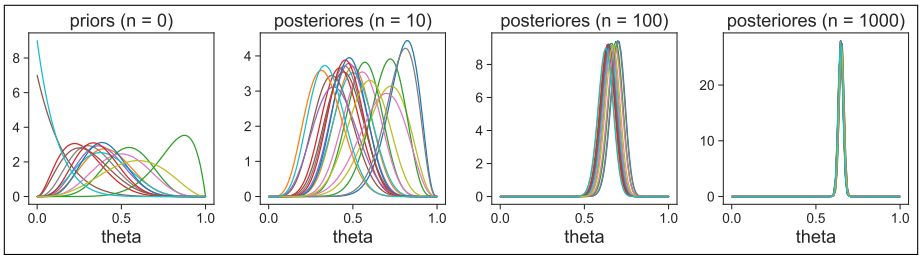


Fig. 10. Concentración de la distribución posterior en el modelo binomial.

2.3.4. Maximo a posteriori

Como ha sido el criterio en esta sección, mediante la distribución posterior podemos representar toda la información que nuestros sesgos y los datos aportan a la caracterización de un modelo (o parámetro),

es decir, simetrías, barras de error, momentos, etc. En particular, es relevante verificar cómo podemos obtener estimaciones puntuales, en una forma similar a las obtenidas mediante mínimos cuadrados o máxima verosimilitud, a través del análisis de la distribución posterior. La motivación para la obtención de estimaciones puntales viene de i) la necesidad de comparar con dichas estimaciones con las obtenidas por MC, MCR y MV, y además ii) los casos donde en efecto necesitamos una estimación puntual y no distribucional, como el problema de toma de decisiones.

Hay distintas alternativas evidentes para extraer una estimación puntual del parámetro θ desde la distribución $p(\theta|\mathcal{D})$, como la media, la mediana y la moda, las cuales son equivalentes cuando la posterior es Gaussiana (o unimodal y simétrica en general). Siguiendo un criterio similar al de máxima verosimilitud consideraremos estimaciones puntuales mediante la maximización de la distribución posterior, consecuentemente, resumiendo la información de la posterior mediante su moda.

Definición 2.3 (Máximo a posteriori)**.** Sea $\theta \in \Theta$ un parámetro con distribución posterior $p(\theta|\mathcal{D})$ definida en todo Θ, entonces nos referimos a estimación puntal dada por

$$\theta_{\text{MAP}} = \arg\max_{\Theta} p(\theta|\mathcal{D}), \tag{81}$$

como *máximo a posteriori (MAP).*

Notemos que es posible encontrar el MAP incluso cuando solo tenemos acceso a una versión *proporcional* a la distribución posterior, un escenario usual en inferencia bayesiana, o también mediante la maximización del logaritmo de ésta última. En efecto,

$$\theta_{\text{MAP}} = \arg\max_{\theta\in\Theta} p(\theta|\mathcal{D}) = \arg\max_{\theta\in\Theta} p(\mathcal{D}|\theta)p(\theta) = \arg\max_{\theta\in\Theta} \left(\underbrace{\log p(\mathcal{D}|\theta)}_{l(\theta)} + \log p(\theta) \right), \tag{82}$$

donde nuevamente vemos la maximización de la función de log-verosimilitud, pero ahora junto al log-prior. Es evidente de esta expresión que si el prior no depende de θ, ess decir, si es uniforme, entonces el criterio MAP es equivalente al MV.

Ejemplo: Máximo a posterior para el modelo lineal y gaussiano

En particular, para el modelo lineal y gaussiano que hemos considerado hasta ahora, podemos calcular θ_{MAP} para un prior Gaussiano de media cero y varianza σ_θ^2. Éste está dado por (asumimos la varianza del ruido σ_ϵ^2 conocida):

$$\begin{aligned}
\theta^\star_{\text{MAP}} &= \arg\max p(Y|\theta, \tilde{X})p(\theta) \\
\text{[independencia, definición]} &= \arg\max \prod_{i=1}^{N} \mathcal{N}(y_i; \theta^\top \tilde{x}_i, \sigma_\epsilon^2)\mathcal{N}(\theta; 0, \sigma_\theta^2) \\
\text{[reordenar]} &= \arg\max \frac{1}{\sqrt{2\pi}\sigma_\epsilon} \frac{1}{(\sqrt{2\pi}\sigma_\theta)^{M+1}} \exp\left(\sum_{i=1}^{N} \frac{-1}{2\sigma_\epsilon^2}(y_i - \theta^\top \tilde{x}_i)^2 - \frac{||\theta||^2}{2\sigma_\theta^2} \right) \\
\text{[logaritmo]} &= \arg\min \sum_{i=1}^{N} (y_i - \theta^\top \tilde{x}_i)^2 + \frac{\sigma_\epsilon^2}{\sigma_\theta^2}||\theta||^2.
\end{aligned} \tag{83}$$

Observemos que esta expresión es equivalente al costo cuadrático regularizado de la ec. (20) con orden $p = 2$, es decir, la solución *máximo a posteriori* del modelo lineal y Gaussiano con prior

Gaussiano es equivalente a la de mínimos cuadrados regularizados (con orden de regularización $p = 2$).

Si bien en la ec. (83) elegimos un prior Gaussiano, pudimos haber elegido un prior exponencial $p(\theta) \propto \exp(\gamma|\theta|)$, con lo que habríamos llegado a MCR con regularización $p = 1$ (o LASSO). Esto conecta claramente el uso de una distribución a priori dentro de la inferencia Bayesiana con el criterio general de regularización: El imponer un prior sobre θ es *promover*, mediante probabilidades relativas, algunas soluciones para θ; mientras que, por el contrario, el uso de un regularizador *penaliza* algunas soluciones. Ajustando apropiadamente la función de regularización y la distribución a priori, podemos llegar a soluciones equivalentes en ambos casos, en particular, la elección de un prior uniforme equivale a un coeficiente de regularización ρ nulo. Sin embargo, debemos recordar que además de la estimación puntual MAP, el enfoque bayesiano entrega la distribución completa sobre el parámetro desconocido.

Un ejemplo de la conexión entre regularizadores y priors puede ser obtenido directamente del desarrollo anterior. La razón entre las varianzas del ruido y del prior en la ec. (83), dada por $\sigma_\epsilon^2/\sigma_\theta^2$, toma el mismo rol que el peso del regularizador ρ/p en la formulación de MCR en la ec. (20). Esto implica que un prior con varianza σ_θ^2 pequeña (cf. grande) es equivalente a un problema de MCR con un ρ grande (cf. pequeño), lo cual tiene sentido: θ puede ser "llevado a cero" mediante la asignación de un prior concentrado en cero o bien penalizando el magnitud de θ (MCR).

Desde ahora, podemos referirnos como MAP a las estimaciones puntuales en general pues, como acabamos de ver, ésta es equivalente a MCR y al mismo tiempo contiene al criterio de máxima verosimilitud como caso particular (cuando el prior es uniforme). Además, para modelos generales (distintos al caso lineal y gaussiano) el MAP no podrá ser calculado de forma explícita imponiendo

$$\nabla_\theta \log p(\theta|\mathcal{D}) = 0, \tag{84}$$

sino que tendremos que considerar algoritmos de optimización. En particular consideraremos algoritmos basados en derivadas con iteraciones de la forma

$$\theta_{i+1} = \theta_i + \eta \nabla_\theta \log p(\theta_i|\mathcal{D}), \tag{85}$$

donde esperamos que la secuencia $\{\theta_i\}_i$, al avanzar en el *ascenso del gradiente de* $\log p(\theta_i|\mathcal{D})$, tienda al punto fijo de la ec. (85), es decir, $\nabla_\theta \log p(\theta|\mathcal{D}) = 0$.

Observación 2.2. En el resto del curso, veremos distintos modelos que requieren técnicas sofisticadas de optimización, las cuales permiten no solo *entrenar* dichos modelos (i.e., encontrar sus parámetros), sino también entender cómo funcionan; estos incluyen redes neuronales, máquinas de soporte vectorial y procesos gaussianos. Con lo visto hasta ahora podemos ver que el aprendizaje de máquinas, se sustenta en dos elementos fundamentales: el modelamiento probabilístico que permite definir *posibles* modelos, y el proceso d optimización que permite encontrar el *buen modelo*. Podemos entonces, a *grosso modo*, decir que "aprender es optimizar".

Ejemplo: MAP para el modelo binomial

Siguiendo el ejemplo anterior de la distribución binomial, recordemos que al asumir un prior $p(\theta) = \text{Beta}(\theta; \alpha, \beta)$ para el parámetro θ, la posterior del mismo (dado un conjuntos de N experimentos con $\{n_i\}$ lanzamientos y $\{k\}_i$ aciertos) está dada por

$$\text{Beta}\left(\theta; \alpha + \sum_{i=1}^{N} k_i, \beta + \sum_{i=1}^{N} n_i - k_i\right). \tag{86}$$

Denotando los parámetros de la posterior mediante $\alpha_N = \alpha + \sum_{i=1}^N k_i$ y $\beta_N = \beta + \sum_{i=1}^N n_i - k_i$,

podemos encontrar el MAP mediante

$$\theta_{\text{MAP}} = \arg\max \theta^{\alpha_N - 1}(1-\theta)^{\beta_N - 1} = \frac{\alpha_N - 1}{\alpha_N + \beta_N - 2,}$$

pues el cálculo no es necesario ya que la moda de la distribución Beta es conocida. Además, la varianza posterior, está dada por

$$\mathbb{V}[\theta|\mathcal{D}] = \frac{\alpha_N \beta_N}{(\alpha_N + \alpha_N)^2(\alpha_N + \beta_N + 1)}. \tag{87}$$

Es directo ver que, como función de las sumas de aciertos/fallos, cuando observamos muchos datos θ_{MAP} tiende a la razón entre aciertos y lanzamientos totales y $\mathbb{V}[\theta|\mathcal{D}]$ tiende a cero. Verificaremos esto numéricamente de la misma forma que el ejemplo anterior: consideremos 20 priors Beta con parámetros α y β aleatorios entre 1 y 10 y grafiquemos tanto θ_{MAP} como $\mathbb{V}[\theta|\mathcal{D}]$ en función de un númeo creciente de lanzamientos binomiales. La Figura 11 confirma nuestra intuición donde (con escala logarítmica para el eje x) podemos ver que con 10 lanzamientos, ya se ve una clara tendencia, mientras que con 100 lanzamientos la convergencia es indudable.

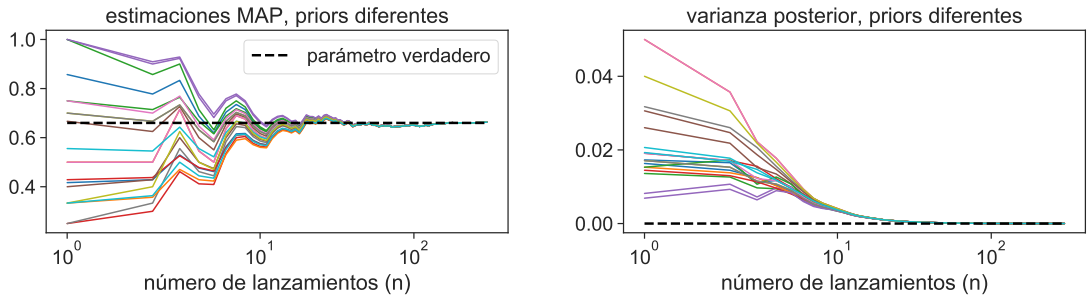


Fig. 11. Concentración de MAP y reducción de varianza posterior: modelo binomial.

2.4. Predicciones

Luego de haber discutido la estimación de parámetros, tanto desde el punto de vista puntual (MC, MCR, MV, MAP) como de estimación bayesiana, ahora veremos cómo estas estimaciones se utilizan para hacer predicciones.

Definamos en primer lugar el contexto para nuestra pedicción. Consideremos un modelo $p(y|x)$, con parámetro θ, del cual obtenido datos denotados mediante $\mathcal{D} = \{(x_i, y_i)\}_{i=1}^N$. Además, introduciremos el concepto de *variable latente*, es decir, una variable dentro del modelo que no es observable, pero que nos interesa predecir. Esta definición es clave para estudiar la predicción: si bien la construcción del modelo para y tiene como objetivo modelar la generación de datos (θ) y sus observaciones (σ), nuestro objetivo final es predecir la cantidad latente, no la observación futura; pues la observación contiene una perturbación asociada a su incertidumbre aleatoria. En el caso lineal, por ejemplo, denotamos la variable latente como $f = \theta^\top x$, donde el modelo está dado por

$$y = \theta^\top x + \epsilon = f + \epsilon, \quad \epsilon \sim \mathcal{N}(0, \sigma^2). \tag{88}$$

Denotaremos entonces mediante $\hat{f}_\star$ e $\hat{y}_\star$ las predicciones de la variable latente f y la observación y para una nueva entrada $x_\star$, condicional a los datos observados $\mathcal{D}$. A continuación veremos que distintos supuestos sobre la estimación de θ resultan en distintas predicciones para estas cantidades, donde utilizaremos el modelo lineal/gaussiano de varianza conocida para ilustrar estas diferencias.

En primer lugar consideremos las estimaciones puntuales, en cuyo caso realizamos un procedimiento conocido como *plug-in prediction*, en donde simplemente utilizamos el valor (puntual) la estimación del parámetro dentro del modelo. En el ejemplo del modelo lineal, si hemos calculado el parámetro mediante máxima verosimilitud (denotado como θ_{MV}) entonces nuestro modelo lineal estimado es $y = \theta_{\text{MV}}^\top \tilde{x} + \epsilon$. Con lo que la predicción usual de la variable lantente correspondiente a una entrada $\tilde{x}_\star$ es determinista (pues tanto θ_{MV} como $x_\star$ lo son) y dada por

$$\hat{f}_\star = \theta_{\text{MV}}^\top \tilde{x}_\star. \tag{89}$$

Por el contrario, si lo que quisiésemos estimar es efectivamente la variable aleatoria $y_\star|\tilde{x}_\star$ (la cual incluye el ruido de observación), esta predicción está dada por

$$\hat{y}_\star \sim \mathcal{N}(\theta_{\text{MV}}^\top \tilde{x}_\star, \sigma^2). \tag{90}$$

Gráficamente, es también usual representar esta predicción en términos de su esperanza y *barras de error*, las cuales para el caso gaussiano son explícitas para el 95 % en función de la desviación estándar del ruido ϵ. Es decir, con un 95 % de probabilidad, $\hat{y}_\star \in [\theta_{\text{MV}}^\top \tilde{x}_\star - 2\sigma, \theta_{\text{MV}}^\top \tilde{x}_\star + 2\sigma]$.

A diferencia de las estimaciones puntuales, cuando el parámetro θ es estimado de forma bayesiana (disponemos de su distribución posterior) el modelo estimado en sí es una **distribución aleatoria**: pues el modelo es una función determinista de un parámetro θ aleatorio. Podemos entonces identificar dos fuentes de incertidumbre en el modelo aprendido como discutimos al comienzo de esta sección: la epistemológica dada por la distribución posterior de θ, y la aleatoria dada por la distribución de ϵ. En nuestro ejemplo del modelo lineal y Gaussiano con varianza conocida el modelo aprendido está dado por

$$y = \theta^\top \tilde{x} + \epsilon, \quad \epsilon \sim \mathcal{N}(0, \sigma^2),\ \theta \sim p(\theta|\mathcal{D}) = \mathcal{N}(\theta_N, \sigma^2 \Lambda_n^{-1}), \tag{91}$$

donde los parámetros de la posterior de θ están descritos en las ecs. (69)-(70). Podemos ver entonces que la relación que tenemos para y está *amarrada* a otras cantidades aleatorias (θ y ϵ), de las cuales nos debemos deshacer para producir la predicción. Esto es porque la introducción de dichas variables fue necesaria para aprender la relación entre x e y a través de nuestro modelo propuesto (o supuesto), y no tienen *necesariamente alguna relación con la realidad*. En la literatura clásica, estos parámetros son llamados *nuisance parameters* y su remoción, la cual es realizda a través del cálculo de una integral, es referido como

to integrate out. Es difícil acordar una traducción al español de este concepto, el cual automáticamente suguiere que estamos "sacando" (*out*) estos parámetros mediante integración; a falta de un mejor término simplemente usaremos la expresión en inglés o su desafortunada traducción "desintegrar".

Entonces, la distribución posterior de la variable latente está dada por la "desintegración" de θ con respecto a su propia distribución posterior, es decir,

$$\hat{f}_\star \sim p(f_\star|x_\star, \mathcal{D}) = \int p(f_\star, \theta|x_\star, \mathcal{D})\mathrm{d}\theta = \int p(f_\star|x_\star, \mathcal{D}, \theta)p(\theta|\mathcal{D}, x_\star)\mathrm{d}\theta = \int p(f_\star|x_\star, \theta)p(\theta|\mathcal{D})\mathrm{d}\theta \tag{92}$$

donde, de izquierda a derecha:

- la primera igualdad pretende *hacer aparecer* $p(f|\theta)$ que es lo que realmente conocemos, donde $p(f|\mathcal{D})$ va a ser calculado mediante la desintegración de θ
- la segunda igualdad es simplemente la definición de distribución condicional
- la tercera igualdad elimina redundacias: dado θ y $x_\star$, $f_\star$ no depende de los datos; de igual forma, la distribución posterior del parámetro depende solo de los datos pasados, no de un nuevo input.

En el caso lineal/gaussiano, el integrando de la expresión más a la derecha de la eq. (92) es la multiplicación de dos gaussianas. Si bien el cálculo de dicha integral es tedioso, ésta tiene la forma de una convolución entre dos gaussianas y sabemos[9] que eso resulta en, nuevamente, una gaussiana. En el caso lineal, sin embargo, ni siquiera es necesario calcular dicha integral, pues podemos calcular la ley posterior de $f_\star$ notando simplemente que $f = \theta^\top \tilde{x}_\star$ y que $\theta \sim \mathcal{N}(\theta_N, \sigma^2\Lambda_n^{-1})$, lo cual da (linealidad)

$$\hat{f}_\star \sim p(f_\star|x_\star, \mathcal{D}) = \mathcal{N}(\theta_N^\top \tilde{x}_\star, \tilde{x}_\star^\top \sigma^2 \Lambda_n^{-1} \tilde{x}_\star). \tag{93}$$

Desde esta expresión es posible graficar el modelo mediante la determinación de las barras de error de los parámetros o bien generando una muestra para el parámetro desde su distribución posterior y luego graficar el modelo con dicha muestra—esto es equivalente a samplear desde la posterior del modelo.

Finalmente, para determinar la predicción de la observación ruidosa $y = f + \epsilon$, simplemente notemos que esta estará dada por la incorporación del estadístico del ruido a la expresión en eq. (93), en efecto,

$$\hat{y}_\star \sim \mathcal{N}(\theta_N^\top \tilde{x}_\star, \tilde{x}_\star^\top \sigma^2 \Lambda_n^{-1} \tilde{x}_\star + \sigma^2). \tag{94}$$

Observación 2.3. De la predicción bayesiana en forma de distribución de la ec. (93) podemos obtener (y es usualmente necesario) una predicción puntual dada por la media, la cual corresponde a

$$\mathbb{E}[f_\star|x_\star, \mathcal{D}] = \mathbb{E}[\theta|\mathcal{D}]^\top \tilde{x}_\star = \bar{\theta}^\top \tilde{x}_\star, \tag{95}$$

donde hemos denotado por $\bar{\theta}$ a la media posterior de θ. Consecuentemente, las estimaciones puntuales reportadas por el método bayesiano y de estimación puntual resultan en la misma predicción. Este fenómeno solo ocurre en situaciones particular, en este caso, debido a que el modelo es lineal y gaussiano.

Finalmente, la Fig. 12 muestra la implementación de las 4 formas de predecir usando el modelo lineal aplicado al dataset de grillos y temperatura usado inicialmente en la Fig. 6. Es posible ver como la predicción bayesiana da sutilmente distintas predicciones que en los extremos del modelo reportan más incertidumbre, mientras que la predicción de y considera tanto ruido que presenta una banda de incertidumbre *uniforme* a lo largo de las observaciones.

[9] Verificable fácilmente a través del teorema de la convolución

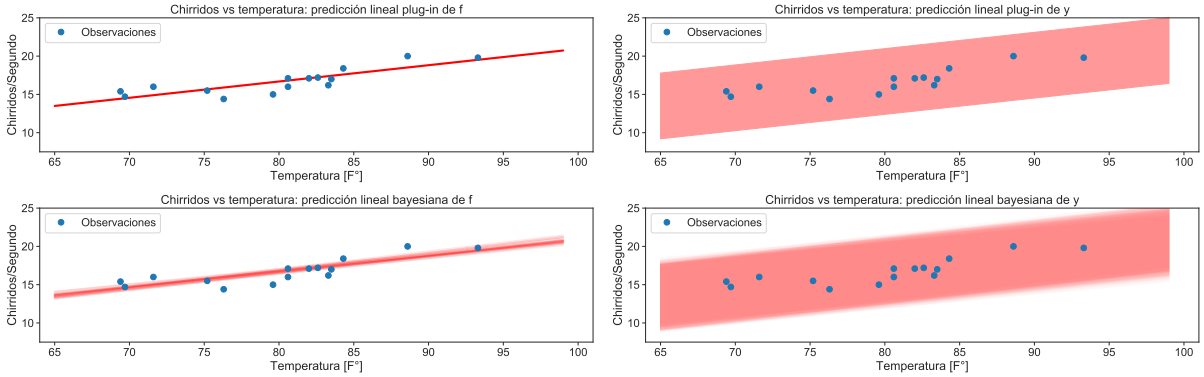


Fig. 12. Ejemplo de predicción usando el modelo lineal para la base de datos de grillos y temperatura. Desde la esquina superior izquierda: predicción plug-in de f, predicción plug-in de y, predicción bayesiana de f (20 muestras), predicción bayesiana de y (20 muestras).

2.5. Ejercicios

Se sabe que el 1 % de las mujeres tienen cancer de mamas, y se tiene un test para detectar si una mujer lo presenta o no. Si la paciente tiene cancer (C), el test dará postitivo (PT) con una probabilidad del 80 % y negativo (NT) con 20 %, en cambio cuando la paciente está sana (NC), hay un 9.6 % de probabilidad que el test salga erroneo y si detecte cancer (PT).

Una paciente se realiza el test y este sale positivo, nos gustaría obtener la probabilidad de que en realidad tenga cancer dado este resultado.

$$p(C|PT) = \frac{p(PT|C)p(C)}{p(PT)} \tag{96}$$

$$= \frac{p(PT|C)p(C)}{p(PT|C)p(C) + p(PT|NC)p(NC)} \tag{97}$$

$$= \frac{0.8 \cdot 0.01}{0.8 \cdot 0.01 + 0.096 \cdot 0.99} \tag{98}$$

$$= 0.0776 \tag{99}$$

De esta misma forma podemos completar todos los casos.

	C (1 %)	NC(99 %)
PT(10.3 %)	7.7 %	92.3 %
NT(89.6 %)	0.2 %	99.8 %

i) Considere el caso en que sus observaciones (entrada x, salida y) solo consisten en

$$D = \{(1, a), (2, b)\}. \tag{100}$$

Usando la expresión para la solución óptima de mínimos cuadrados de la ecuación (10), encuentre los parámetros del modelo lineal dado por

$$y = \theta_1 x + \theta_0 \tag{101}$$

e interprete esta solución para distintos valores de a y b.

ii) ¿Cuál es la estimador muestral de la covarianza entre x e y para las observaciones disponibles?

ii) Interprete la correlación entre x e y

3. Regresión No Lineal

El modelo de regresión lineal visto en el capítulo anterior tiene diversas ventajas, en primer lugar su solución en el caso de mínimos cuadrados (o máxima verosimilitud en el caso de ruido gaussiano) puede ser encontrado de forma explícita y única. Además, sus parámetros permiten interpretar la relación entre cada una de las componentes de la variable de entrada y la variable de salida. Sin embargo, más allá de estas propiedades del modelo lineal, su alcance es muy limitado pues muchas veces necesitamos modelar fenómenos que no siguen una relación lineal.

El concepto de regresión lineal puede ser extendido a una contraparte no lineal en los casos particulares que conocemos a priori el tipo de relación entre las variables de entrada x y salida y (y esta no es lineal). Dicha extensión puede ser obtenida simplemente mediante la aplicación de una transformación (de parámetros fijos) a la variable independiente, e.g. $\phi(x)$, para luego construir un modelo lineal en la variable transformada $\phi = \phi(x)$ en lugar de en la variable original x.

Específicamente, consideraremos transformaciones a valores vectoriales de la variable independiente de la siguiente forma

$$\begin{aligned} \phi\colon \mathbb{R}^M &\to \mathbb{R}^D \\ x &\mapsto \phi(x) = [\phi_1(x), \ldots, \phi_D(x)]^\top, \end{aligned} \tag{102}$$

donde $\phi_i : x \in \mathbb{R}^M \mapsto \phi_i(x) \in \mathbb{R}, \forall i = 1, \ldots, D$ son funciones escalares.

Con la introducción de la transformación $\phi(\cdot)$ es necesario hacer la distinción entre ambas variables de entrada. Nos referiremos entonces a x como datos/entradas crudos (*raw data/input*) y a la variable $\phi = \phi(x)$ como características (*features*). Recordemos que los modelos del capítulo anterior eran aplicados directamente sobre la entrada cruda x (o $\tilde{x}$), con lo que no existía distinción entre entradas crudas o características. Adicionalmente, la introducción del concepto de *característica* es motivado por la búsqueda de una representación de x que permite resolver el problema de regresión usando un modelo de simple (y también interpretable) como el modelo lineal, esto está en contraste con la idea de diseñar un modelo muy complicado (no interpretable) que reciba directamente la entrada cruda x. Un ejemplo de esto es el caso en que x es una imagen: en este caso no es posible aplicar directamente un modelo lineal a la representación matricial de la imagen, sino que a una representación distinta de este, o a sus *características*, es decir, sus bordes, formas, y otros patrones a identificar dentro de la imagen.

En la práctica, la función $\phi : x \mapsto \phi(x)$ es elegida en base al conocimiento *experto* que se tenga del problema de regresión a resolver; como su elección pretende extraer las características de interés y representarlas de una forma compatible con el modelo lineal, entonces, nos referiremos a la construcción *manual* de la función ϕ como *ingeniería de características*. Observemos entonces que la función ϕ puede tener (y en general tiene) parámetros, los cuales en esta primera instancia serán elegidos manualmente. Sin embargo, más adelante veremos modelos donde los parámetros de ϕ también pueden ser encontrados de forma automática (e.g., mediante máxima verosimilitud). En dicho caso, es posible interpretar que la ingeniería de características ya no es realizada de forma manual, sino que automatizada.

Características y modelos

El diseño de características es uno de los problemas fundamentales del AM. Esto porque podemos entender la construcción de modelos de aprendizaje supervisado como dos etapas: i) la identificación de las características relevantes y ii) cómo usar estas características para estimar el output (el modelo). En el modelo no lineal presentado aquí, es claro que la parte no lineal ϕ es la característica y la parte lineal es el modelo, sin embargo, en el caso general la línea entre ambas etapas es muy delgada. Por ejemplo, en una red neuronal de 100 capas, ¿qué corresponde a característica y qué corresponde al modelo? Podríamos argumentar que en modelos más complicados como los de redes neuronales, existen varias interpretaciones de qué corresponde a

cada cosa, y con eso de paso interpretar qué está haciendo nuestro modelo. El hecho de que el diseño de características y el modelo se confundan es interesante también, pues quiere decir que las características se aprenden de igual forma que el resto del modelo; con lo que pasamos desde un diseño manual de características a una búsqueda automáticas de características.

3.1. Modelo lineal en los parámetros

Usando la nueva variable de características $\phi = \phi(x)$ como entrada a un modelo lineal, podemos definir el siguiente modelo de regresión lineal y gaussiano:

$$y = \theta^\top \phi(x) + \epsilon, \quad \epsilon \sim \mathcal{N}(0, \sigma_\epsilon^2), \tag{103}$$

donde asumiremos que contamos con un conjunto de observaciones de la forma

$$\mathcal{D} = \{(x_i, y_i)\}_{i=1}^N, \quad (x_i, y_i) \in \mathbb{R}^M \times \mathbb{R}^1, \forall i = 1, \dots, N. \tag{104}$$

De la misma forma que vimos en el capítulo anterior, este modelo puede ser entrenado mediante el la optimización de costo cuadrático

$$J = \frac{1}{2} \sum_{i=1}^N (y_i - \theta^\top \phi(x_i))^2, \tag{105}$$

lo cual es equivalente a máxima verosimilitud, pues el ruido de observación ϵ es gaussiano.

Para una presentación sencilla, evitaremos el uso de las sumatorias y adoptaremos la siguiente notación matricial:

$$X = \begin{bmatrix} x_1 \\ \vdots \\ x_N \end{bmatrix} \qquad \Phi = \begin{bmatrix} \phi(x_1) \\ \vdots \\ \phi(x_N) \end{bmatrix} = \begin{bmatrix} \phi_1(x_1) & \dots & \phi_D(x_1) \\ \vdots & \ddots & \vdots \\ \phi_1(x_N) & \dots & \phi_D(x_N) \end{bmatrix} \qquad Y = \begin{bmatrix} y_1 \\ \vdots \\ y_N \end{bmatrix}, \tag{106}$$

con lo que el modelo y el funcional de costo puede ser expresados respectivamente mediante

$$Y = \Phi\theta + \boldsymbol{\epsilon}, \quad \boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \sigma_\epsilon^2 \mathbf{I}) \tag{107}$$

$$J = \frac{1}{2}(Y - \Phi\theta)^\top (Y - \Phi\theta). \tag{108}$$

De forma análoga al modelo lineal estándar (el cual toma directamente la variable cruda x como input), la minimización del funcional J puede ser encontrado mediante la condición de primer orden $\nabla_\theta J = 0$ y está dada por

$$\theta\star = (\Phi^\top \Phi)^{-1} \Phi^\top Y. \tag{109}$$

Las siguientes observaciones conectan el modelo no lineal de la ec. (103) con el modelo lineal estudiado en el Capítulo 2.

Observación 3.1. De la ec. (109) vemos que si elegimos $\phi(\cdot) = \mathrm{id}(\cdot)$ recuperamos la expresión para mínimos cuadrados ordinarios. Esto permite interpretar el modelo de regresión no lineal con variables $\{(x_i, y_i)\}_{i=1}^N$ de $\mathbb{R}^N$ a $\mathbb{R}$ como una regresión lineal con variables $\{(\Phi(x_i), y_i)\}_{i=1}^N$ de $\mathbb{R}^D$ a $\mathbb{R}$. Además, la razón por la que la forma de la solución es la misma que el caso lineal es porque si bien el modelo presentado en la ec. (103) es no lineal en la entrada x, este es *lineal en los parámetros* θ.

Observación 3.2. Notemos que la re-parametrización del modelo afín introducida en el capítulo anterior es un caso particular de la transformación Φ. Esto es directo de la siguiente construcción:

$$\Phi = [x, 1]^\top$$
$$y = a^\top x + b = \theta^\top \Phi(x) + \epsilon.$$

Observación 3.3. Finalmente, es posible considerar una extensión regularizada al modelo no lineal presentado en la ec. (103) mediante la consideración de un costo regularizado cuadrático (i.e., *ridge regression*) dado por

$$J_r = \frac{1}{2}\sum_{i=1}^{N}(y_i - \phi(x_i)\theta)^2 + \rho\left\|\theta\right\|^2, \quad \rho \in \mathbb{R}^+. \tag{110}$$

En cuyo caso, la solución está dada por

$$\theta = (\Phi^\top\Phi + \rho\mathbb{I})^{-1}\Phi^\top Y. \tag{111}$$

(S)elección de características

Un buen conjunto de características no solo ayuda a una buena representación (y consecuentemente predicción) de nuestros datos, sino que extrae literalmente las *características* que son relevantes de x para determinar y. Por ejemplo, si necesitamos determinar la condición clínica de un paciente, para el cual tenemos una vasta colección de mediciones como peso, edad, género, dirección, número de teléfono, ocupación, color de ojos, salario, etc, muy probablemente muchas estas características no van a ser útil (por ejemplo, ocupación), con lo que el diseño de las funciones ϕ debe tomar en cuenta esto. Por el contrario, si necesitamos evaluar el mismo conjunto de individuos para un crédito de consumo, las características a considerar variarán y ahora probablemente el salario sí juegue un rol importante. Este concepto permite entender la estrecha relación entre el diseño de características y la *selección de características* y consecuentemente la *selección de modelos*, lo cual veremos en los siguientes capítulos. En particular, recordemos que usando el regulador LASSO, podemos llevar algunas de nuestras coordenadas a cero, realizando una selección automática de características.

3.2. Ejemplos de transformaciones

La selección de las características es fundamental para una representación apropiada de nuestros datos y consecuentemente para el desempeño del modelo de regresión no lineal. En efecto, si la familia de funciones $\phi_1, \ldots, \phi_D$ no captura la representación de las características de forma compatible con el modelo lineal, el esfuerzo por ajustar la parte lineal del modelo será infructuoso. En general, la elección de la base de funciones debe ser diseñada y evaluada caso a caso.

A continuación vemos algunos ejemplos de características que son usualmente usadas en la práctica por su generalidad (habilidad para replicar varios comportamientos) y/o simpleza.

Función Polinomial: $\phi = \{\phi_i\}_{i=0}^{D}$, donde $\phi_i(x) = x^i$, de tal forma que

$$\Phi = \begin{bmatrix} 1 & x_1 & x_1^2 & \ldots & x_1^D \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & x_N & x_N^2 & \ldots & x_N^D \end{bmatrix}. \tag{112}$$

La función polinomial es ampliamente usada debido sla capacidad de la base polinomial para aproximar cualquier función *bien portada* con una precisión determinada (considerando un grado D suficientemente grande). Sin embargo, una desventaja de esta base es que tanto su entrenamiento como predicciones pueden ser inestables: si se usa un grado D muy alto, los valores de $\phi(x)$ crecen, obviamente, de forma *polinomial*.

Función Sinusoidal: $\phi = \{\phi_i\}_{i=0}^{D}$, donde $\phi_i(x) = \cos\left(i\frac{2\pi}{2T}(x - b_i)\right)$. La variable i actúa como una frecuencia normalizada con respecto al período de oscilación T y b_i es un "offset" o fase. Esta transformación está dada por

$$\Phi = \begin{bmatrix} 1 & \cos\left(1\frac{2\pi}{2T}(x_1 - b_1)\right) & \dots & \cos\left(D\frac{2\pi}{2T}(x_1 - b_D)\right) \\ \vdots & \vdots & \ddots & \vdots \\ 1 & \cos\left(1\frac{2\pi}{2T}(x_N - b_1)\right) & \dots & \cos\left(D\frac{2\pi}{2T}(x_N - b_D)\right) \end{bmatrix}. \tag{113}$$

Hemos asumido que el la fase para cada función ϕ_i, la frecuencia $i\frac{2\pi}{2T}$ y fase b_i son fijas. Una forma de evitar definir una fase, es considerar dos transformaciones por cada ϕ_i de la forma

$$\phi_i'(x) = \left[\sin\left(i\frac{2\pi}{2T}x\right),\ \cos\left(i\frac{2\pi}{2T}x\right)\right], \tag{114}$$

donde no es necesario definir la fase sino que ésta es absorbida por el parámetro θ y aprendida.

Al igual que los polinomios, la base de cosenos es también *universal*. Sin embargo, una desventaja de la base senoidal es que solo puede replicar funciones periódicas, con un período máximo en este caso de T. En la práctica, veremos que un modelo sinusoidal se repetirá cada período, pudiendo llevar a conclusiones erróneas. Además, si efectivamente nuestros datos representan un comportamiento periódico pero el ruido de observación resulta en que cada período o *forma de onda* sean ligeramente distinto, puede ser difícil encontrar el período T. La identificación de este período es un desafío en sí mismo, conocido como *detección de periodicidades o frecuencias fundamentales*, usual en astronomía y procesamiento de audio.

Funciones constantes por partes: La siguiente característica está está compuesta por "escalones", los cuales valen "1" dentro de un conjunto especifico y "0" en cualquier otro lugar. Sean $c_1, c_2, \dots, c_D \in \mathbb{R}$ una colección de valores crecientes, definimos la indicatriz I_A y las funciones de pertenencia C_i:

$$C_0(x) = I_{(-\infty, c_1)}(x) \tag{115}$$

$$C_i(x) = I_{[c_i, c_{i+1})}(x), \quad \forall i = 1, \dots, D-1 \tag{116}$$

$$C_D(x) = I_{[c_D, +\infty)}(x) \tag{117}$$

$$I_A(x) = \begin{cases} 1, & x \in A \\ 0, & x \notin A \end{cases} \tag{118}$$

De esta forma la base de funciones $\Phi = \{\phi_i\}_{i=0}^{D}$ queda definida por

$$\phi_i(x) = C_i(x), \quad \forall i = 0, \dots, D \tag{119}$$

$$\Phi(X) = \begin{bmatrix} 1 & C_0(x_1) & \dots & C_D(x_1) \\ \vdots & \vdots & \ddots & \vdots \\ 1 & C_0(x_N) & \dots & C_D(x_N) \end{bmatrix}. \tag{120}$$

Las funciones contantes por partes son útiles cuando trabajamos con experimentos que involucran una discretización, por ejemplo, al modelar la altura/concentración/conductividad/temperatura en una superficie y no es posible parametrizar esta cantidad como función del (en este caso) espacio. Entonces, simplemente dividimos la superficie en una grilla y asignamos un valor distinto a cada celda de esta grilla. Es un método útil pero la cantidad de parámetros (dimensión de θ) crece con el número de celdas de la grilla, lo cual es prohibitivo para casos donde el rango de valores de x y/o su dimensión es medianamente alta.

Ejemplo: Predicción de pasajeros de una aerolínea)

Consideremos el problema de predecir la cantidad de pasajeros en una aerolínea, considerando distintas combinaciones de características. De forma incremental, tomaremos en consideración características polinomiales, senoidales, y senoidales con amplitud creciente. Es decir, denotando x en tiempo y y la cantidad de pasajero, consideraremos el siguiente modelo

$$y = \underbrace{\sum_{i=0}^{3} \theta_i x^i}_{\text{parte polinomial}} + \underbrace{\sum_{i=1}^{2} \alpha_i \exp(-\tau_i x^2) \cos(\omega_i (x - \psi_i))}_{\text{parte oscilatoria}} . \tag{121}$$

La motivación de este modelo es representar la tendencia de los datos mediante la componente polinomial y la oscilación anual mediante las componentes oscilatorias.

Además, hemos considerado 12 años de datos con frecuencia mensual (144 datos), de los cuales solo 9 años (108 datos) han sido usado para encontrar los parámetros del modelo y los 3 años restantes (36 datos) para validar nuestras predicciones. La Figura 13 muestra los datos y las estimaciones para 3 variantes del modelo mencionado: parte polinomial, parte polinomial y oscilatoria de amplitud fija, y finalmente, parte polinomial y oscilatoria de amplitud variable. En cada gráfico se muestran distintas opciones de cada modelo en gris con la mejor trayectoria en negro; se puede ver además en cada gráfico cómo disminuye el error de aproximación a medida consideramos más características.

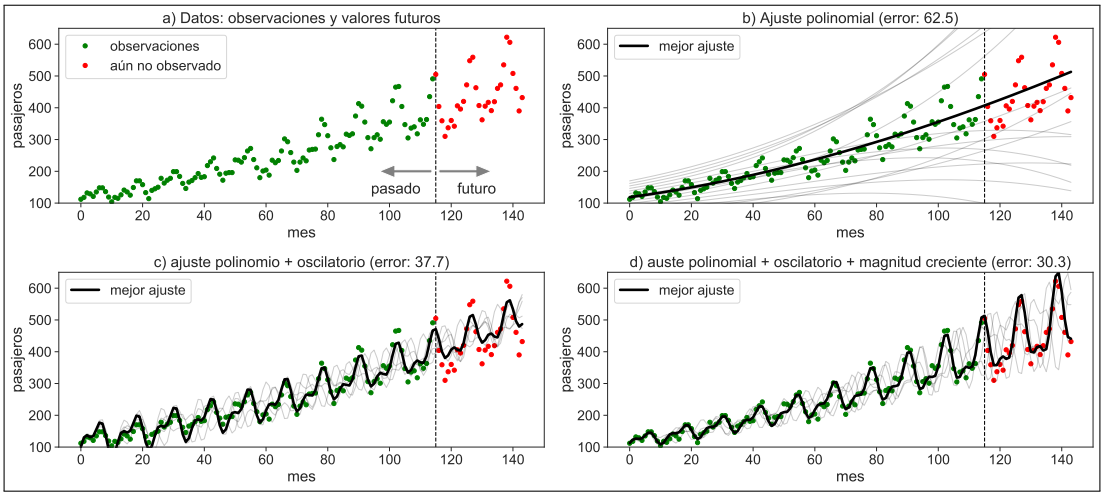


Fig. 13. Regresión no-linea de cantidad de pasajeros en el tiempo. Arriba-izq: datos de entrenamiento, arriba-der: modelo con componente polinomial, abajo-izq: modelo con componentes polinomial y oscilatoria (amplitud fija), abajo-der: modelo con componentes polinomial y oscilatoria (amplitud creciente).

Cosas pendiente

2) Ejemplo de cómo calcular el gradiente y cómo usar gradiente estocástico
3) Discutir el calculo, análisis y muestreo de la posterior
4) Breve mención de variables latentes y casos no-tratables

4. Clasificación

El problema de clasificación dice relación con la identificación del conjunto, categoría o *clase* a la cual pertenece un elemento en base a sus *características*. En el contexto del aprendizaje supervisado, el problema de clasificación puede ser visto como un caso particular del problema de regresión, donde el espacio donde la variable y (salida o variable dependiente) es *categórico* y usualmente denotado por $\{0, 1\}$, para el caso binario, o bien $\{1, 2, \ldots, M\}$ para el caso de clasificación multiclase.

4.1. Formulación: clasificación lineal

Consideremos un conjunto de datos

$$\mathcal{D} = \{(x_i, c_i)\}_{i=1}^{N} \subset \mathbb{R}^M \times \{\mathcal{C}_i\}_{i=1}^{K}, \tag{122}$$

donde, al igual que en el problema de regresión, x es la variable independiente y c es la variable dependiente o *clase*. Consideraremos en primera instancia el caso *binario*, es decir, solo dos clases ($K = 2$), ilustrado en la Fig. 14. Proponemos un modelo lineal para relacionar la variable independiente con su clase, es decir,

$$y(x) = a^\top x + b, \tag{123}$$

donde la asignación de la clase es de la siguiente forma: x será asignado a $\mathcal{C}_1$ si $y(x) \geq 0$ y será asignado $\mathcal{C}_2$ en caso contrario.

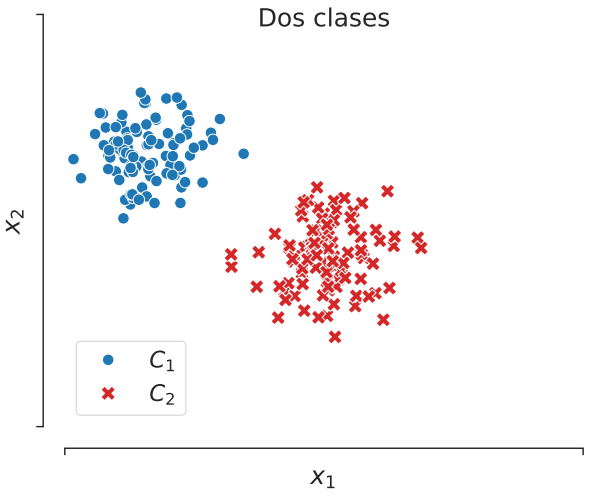


Fig. 14. Ejemplo del problema de clasificación binaria, donde la clase $\mathcal{C}_1$ está presentada en azul y la clase $\mathcal{C}_2$ en rojo.

Nos referiremos al subconjunto que particiona $\mathbb{R}^M$ en clase $\mathcal{C}_1$ y clase $\mathcal{C}_1$ como *superficie/hiperplano/-región de decisión*, la cual está definida por $y(x) = 0$. Entonces, como $x \in \mathbb{R}^M$, para la consideración del modelo lineal esta superficie de decisión corresponde a un hiperplano de dimensión $M - 1$.

Para resolver el problema de clasificación, debemos encontrar los parámetros a y b en la ec. (123). Para este fin, veamos que si x_1, x_2 son dos puntos en la superficie de decisión, entonces tenemos la siguiente relación para el parámetro a:

$$\begin{aligned} 0 &= y(x_1) - y(x_2) \\ &= a^\top x_1 + b - a^\top x_2 - b \\ &= a^\top (x_1 - x_2). \end{aligned} \tag{124}$$

Es decir, a es ortogonal a cualquier vector que esté contenido dentro de la región de decisión. Intuitivamente, podemos decir que el vector $a \in \mathbb{R}^M$ controla la pendiente (u orientación) del hiperplano de decisión. Además, observemos que para cualquier x en la región de decisión, i.e., $y(x) = 0$, su proyección en el vector a está dada por

$$\frac{a^\top x}{\|a\|} = -\frac{b}{\|a\|}, \tag{125}$$

de donde podemos entender que el parámetro b controla el desplazamiento (o ubicación) de la región de decisión, pues el lado derecho de la ecuación representa la distancia entre la región de decisión y el origen.

Es posible también interpretar $y(x)$ como una distancia con signo entre un $x \in \mathbb{R}^M$ cualquiera y la superficie de decisión. Para ver esto, consideremos $x \in \mathbb{R}^M$ y descompongámoslo dos componentes:

la primera denotada por $x_\perp$, la cual es la proyección ortogonal de x en el hiperplano de decisión, y la segunda que es perpendicular al hiperplano (y consecuentemente paralela al vector a) denotada por $r\frac{a}{\|a\|}$, donde r denota la distancia entre x y el hiperplano de decisión. Expresamos entonces

$$x = x_\perp + r\frac{a}{\|a\|}, \tag{126}$$

y observamos que

$$y(x) = a^\top x + b = a^\top \left(x_\perp + r\frac{a}{\|a\|} \right) + b = \underbrace{a^\top x_\perp + b}_{=0} + r\frac{a^\top a}{\|a\|} = r||a||. \tag{127}$$

Consecuentemente, vemos que $\forall x \in \mathbb{R}^M$, $r = \frac{y(x)}{||a||}$, es decir, $y(x)$ representa una medida con signo de la distancia entre el punto x y la región de decisión.

El caso de múltiples clases ($K > 2$) puede ser enfrentado mediante una extensión del caso binario. Una forma puede ser construir $K - 1$ clasificadores binarios, en donde cada uno resuelve el problema de discriminar los elementos de la clase $\mathcal{C}_k$ del resto; esta técnica se llama *one-versus-rest*. Otra forma es construir $K(K-1)/2$ clasificadores binarios, donde cada uno discrimina entre cada posible par de clases; esta técnica se llama *one-versus-one*. El problema con estas dos alternativas es que dejan regiones indefinidas en el espacio, pues solo consideran pares de clases que al ser agregados pueden ser incoherentes.

Una alternativa más robusta para resolver el problema de clasificación multiclase es construir un clasificador para K clases que contiene K funciones lineales de la forma

$$y_k(x) = a_k^\top x + b_k, \quad k = 1, \ldots, K. \tag{128}$$

Donde x es asignado a la clase $\mathcal{C}_k$ si y solo si $y_k(x) > y_j(x), \forall j \neq k$, es decir:

$$\mathcal{C}(x) = \arg\max_k y_j(x). \tag{129}$$

La región de decisión de este clasificador de K clases es efectivamente convexa y no quedan secciones de $\mathbb{R}^M$ sin asignar clase (o con asignación incoherente). Además, si elegimos $K = 2$, obtenemos el clasificador binario. (Esto puede quedar de tarea 3)

4.2. Ajuste mediante mínimos cuadrados

Ya hemos planteado el modelo y analizado el rol y significado de cada uno de sus parámetros; ahora queda por estudiar cómo determinar dichos parámetros a y b, dado un conjunto de datos $\mathcal{D}$. Para esto consideraremos en primer lugar el enfoque de mínimos cuadrados, el cual es la medida de discrepancia por excelencia y a primera vista parece una respuesta natural a este problema. Primero introduciremos un poco de notación para plantear el problema de forma clara.

Consideremos el punto $x \in \mathbb{R}^M$ con clase $c \in \{\mathcal{C}_k\}_{k=1}^K$. Usaremos la *codificación* $t \in \{0,1\}^K$ para representar la pertenencia de x a su respectiva clase. Es decir,

$$c = \mathcal{C}_j \Leftrightarrow [t]_j = 1 \wedge [t]_i = 0, \quad i \neq j. \tag{130}$$

Este tipo de codificación es conocida como *one-hot encoding.*

Observación 4.1. Usamos esta codificación por dos razones. Primero, para poder dar valores numéricos a la clase y que podamos hablar de distancias entre ellos, pues si nuestras clases son "peras y manzanas" necesitamos poder determinar cuán cerca a cada una de ellas está nuestra estimación. En segundo lugar, como los valores asignados al vector t siempre serán puros 0's y un solo 1, entonces todos los posibles valores de t están a la misma distancia unos de otros. De esta forma, si dos elementos tienen distintas clases, la distancia entre éstas, cualquieras sean, será 0 (clases iguales), o bien $\sqrt{2}$ (clases distintas). Esto ayuda a no introduce sesgos en la representación de la clase que puedan alterar el aprendizaje del modelo.

Asumiendo entonces el modelo lineal para cada clase $\mathcal{C}_k$ dado por

$$y_k(x) = a_k^\top x + b_k, \tag{131}$$

podemos reescribir el modelo de forma matricial como:

$$y(x) = \tilde{\Theta}^\top \tilde{x}. \tag{132}$$

Donde la k-esima columna de $\tilde{\Theta}$ es un vector de $M+1$ dimensiones definido por $\tilde{\theta}_k = [b_k, a_k^\top]^\top$, $\tilde{x} = [1, x^\top]^\top$ corresponde al vector x aumentado y la cantidad $y(x)$ es el vector de *asignaciones de clases*. De esta forma, un punto x será asignado a la clase que tenga mayor $y_k = \tilde{\theta}_k^\top \tilde{x}$. Es decir, la clase de x es modelada como el $\arg\max y(x)$.

Con la notación establecida, ahora podemos enfocarnos en el entrenamiento del modelo. Para esto consideremos un conjunto de entrenamiento $\{(x_n, t_n)\}_{n=1}^N$, y definamos las siguientes matrices:

$$T = [t_1, t_2, \ldots, t_N]^\top \tag{133}$$

$$\tilde{X} = [\tilde{x}_1, \tilde{x}_2, \ldots, \tilde{x}_N]^\top. \tag{134}$$

Con lo que estamos en posición de determinar los coeficientes de $\tilde{\Theta}$ mediante la minimización de la suma de los errores cuadráticos de forma similar al problema de regresión. Podemos escribir la función de costovde la siguiente forma:

$$J(\tilde{\Theta}) = \mathrm{Tr}\left[(\tilde{\Theta}\tilde{X} - T)^\top (\tilde{\Theta}\tilde{X} - T)\right], \tag{135}$$

donde $\mathrm{Tr}\,[\cdot]$ corresponde al operador traza. Al aplicar la condición de primer orden a la ec. (135) (i.e., derivar e igualar a 0), obtenemos la solución de mínimos cuadrados para el problema de clasificación dado por:

$$\tilde{\Theta} = (\tilde{X}^\top \tilde{X})^{-1} \tilde{X} T, \tag{136}$$

con lo que la predicción de la clase para un nuevo input x está dada por $y(x) = T^\top \tilde{X}^\top (\tilde{X}^\top \tilde{X})^{-1} \tilde{x}$.

Observación 4.2. Hay dos problemáticas conceptuales con este enfoque. En primer lugar, el uso de mínimos cuadrados es muy sensible a la presencia de puntos aislados (*outliers*). Dado el crecimiento cuadrático de la penalización, los puntos lejanos al promedio de los datos tienen una influencia mucho mayor. Esto ciertamente afecta considerablemente los resultados y no es coherente con el problema de clasificación, donde la estimación es correcta o incorrecta, pero no "más" o "menos" correcta. Este efecto sobre la solución es ilustrado en la Figura 15, en donde la presencia de puntos alejados de clase $\mathcal{C}_2$ afecta el resultado (derecha), incluso donde el resultado original (izquierda) estaba correcto. En segundo lugar, las predicciones de clase $y(x)$ no tienen la forma de las etiquetas originales t, con lo cual es necesario una intervención "manual" en donde, dado un $y(x)$ debemos definir el $t(x)$ apropiado.

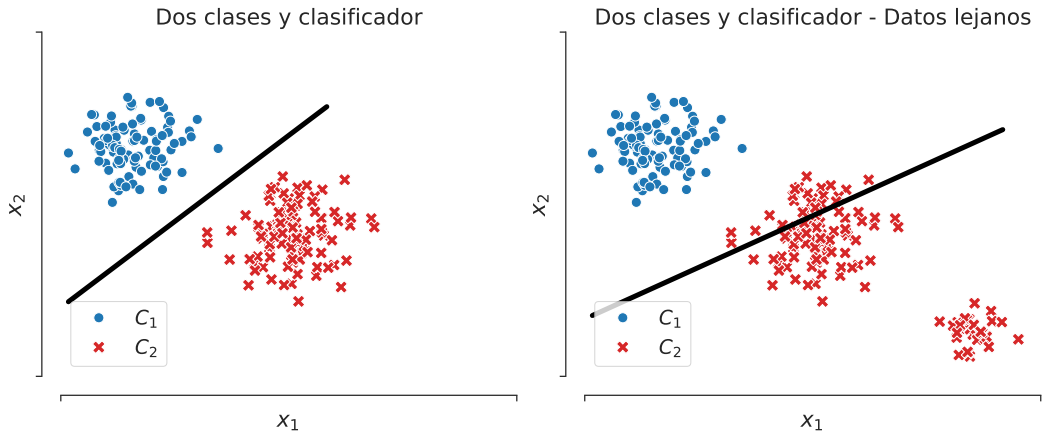


Fig. 15. Ejemplo ilustrativo sobre cómo los puntos lejanos de una clase pueden afectar incorrectamente los resultados.

4.3. Discriminante lineal de Fisher

Para evitar los artefactos (sesgos) introducidos por clases asimétricas en el uso de mínimos cuadrados para clasificación, es posible interpretar el problema de clasificación como uno de *reducción de dimensionalidad*[10], en donde la reducción consiste representar nuestros datos en solo una dimensión, la cual representa su (grado de pertenencia a una) clase. Con este objetivo en mente, consideremos el problema de clasificación binaria de $x \in R^M$, donde proyectamos x en un espacio **unidimensional** con respecto a un vector $a \in \mathbb{R}^M$ de acuerdo a:

$$y = a^\top x, \tag{137}$$

donde podemos definir un umbral b para asignar x a $\mathcal{C}_1$ si $y \geq -b$ y x a $\mathcal{C}_2$ en caso contrario. Notemos que de esta forma recuperamos el modelo lineal para clasificación presentado al comienzo de este capítulo.

En general, al proyectar el objeto M-dimensional en un espacio 1-dimensional, se pierde gran cantidad de la información, lo cual expresa en el hecho que clases claramente separadas en el espacio M-dimensional pueden traslaparse al ser proyectadas a 1 dimensión cuando la elección del vector $a \in \mathbb{R}^M$ no es la mejor. Sin embargo, es posible ajustar el vector a con la finalidad de obtener una proyección de x que maximice el grado de separación entre clases.

Con el objetivo de encontrar el vector a que cumple con este requerimiento en base a un conjunto de datos $\mathcal{D}$, primero definamos las cardinalidades de clases mediante $N_1 = |\{x|x \in \mathcal{C}_1\}|$ y $N_2 = |\{x|x \in \mathcal{C}_2\}|$, lo cual permite calcular los promedios (muestrales) de cada clase mediante:

$$\mu_1 = \frac{1}{N_1} \sum_{n \in \mathcal{C}_1} x_n, \qquad \mu_2 = \frac{1}{N_2} \sum_{n \in \mathcal{C}_2} x_n. \tag{138}$$

La medida más simple de separación entre las proyecciones de las clases sobre a es la distancia entre las medias de sus proyecciones:

$$m_1 - m_2 = a^\top(\mu_1 - \mu_2), \tag{139}$$

donde $m_k = a^\top \mu_k$ corresponde al promedio de los elementos de la clase $\mathcal{C}_k$ proyectado sobre el vector a. Consecuentemente, el vector a que maximiza la distancia entre la proyección de clases es el que maximiza la expresión anterior. Sin embargo, esta expresión puede ser arbitrariamente grande si escalamos a, entonces, para evitar estas redundancias impondremos que $\|a\|_2 = 1$. Usando multiplicadores de Lagrange para optimizar el problema restringido se llega a que $a \propto (\mu_1 - \mu_2)$.

Una desventaja de este enfoque, en el cual se ha ignorado la dispersión de las clases y solo se ha considerado su media, es que pueden existir 2 clases bien separadas en el espacio D-dimensional, pero al proyectar los datos sobre la recta que une sus promedios, las proyecciones de cada clase se traslapen. Para resolver este problema, Fisher propuso maximizar más que la mera distancia entre las (medias de las) clases proyectadas, sino que adicionalmente minimizar la dispersión de los elementos de una misma clase, con el objetivo de disminuir el traslape entre las proyecciones de las clases.

Como medida de dispersión, definimos la varianza (muestral) de los elementos de la clase $\mathcal{C}_k$ mediante

$$s_k^2 = \sum_{n \in \mathcal{C}_k} (a^\top(x_n - \mathrm{m}_k))^2 \tag{140}$$

$$= \sum_{n \in \mathcal{C}_k} (y_n - m_k)^2, \tag{141}$$

lo cual nos permite definir la siguiente función objetivo

$$J(a) = \frac{m_1 - m_2}{s_1^2 + s_2^2}, \tag{142}$$

[10]El problema de reducción de dimensionalidad consiste con construir una representación de dimensión estrictamente menor que los datos originales con la finalidad de interpretar de mejor forma la información contenida en nuestros datos

donde explícitamente vemos la discrepancia "inter" clases en el numerador y la discrepancia "intra " clases en el denominador. Adicionalmente, podemos expresar este costo directamente como función del vector de proyección a:

$$J(a) = \frac{a^\top S_B a}{a^\top S_W a}, \tag{143}$$

donde la matriz de covarianza entre clases S_B y matriz total de covarianza dentro de clases S_W están respectivamente dadas por

$$S_B = (\mu_1 - \mu_2)(\mu_1 - \mu_2)^\top \tag{144}$$

$$S_W = \sum_{n \in \mathcal{C}_1} (x_n - \mu_1)(x_n - \mu_1)^\top + \sum_{n \in \mathcal{C}_2} (x_n - \mu_2)(x_n - \mu_2)^\top. \tag{145}$$

Aplicando la condición de primer orden para $J(a)$, obtenemos que el vector a óptimo debe cumplir

$$(a^\top S_B a) S_W a = (a^\top S_W a) S_B a. \tag{146}$$

Sin embargo, notemos que la norma del vector a es irrelevante, solo interesa su orientación, con lo que ignorando los escalares $(a^\top S_B a)$ y $(a^\top S_W a)$ tenemos que la relación de optimalidad es $S_W a \propto S_B a$. Además, por la definición de S_B, sabemos que $S_B a \propto (\mu_1 - \mu_2)$, con lo que la relación de optimalidad se convierte en es $S_W a \propto (\mu_1 - \mu_2)$. Consecuentemente, el vector optimo a en el criterio de Fisher debe cumplir

$$a \propto S_W^{-1}(\mu_1 - \mu_2). \tag{147}$$

La Figura 16 muestra el discriminador lineal que solo considera los promedios a la izquierda y la corrección de Fisher a la derecha. Observemos cómo el incluir una medida de la dispersión de los datos es clave para lograr un mejor discriminador.

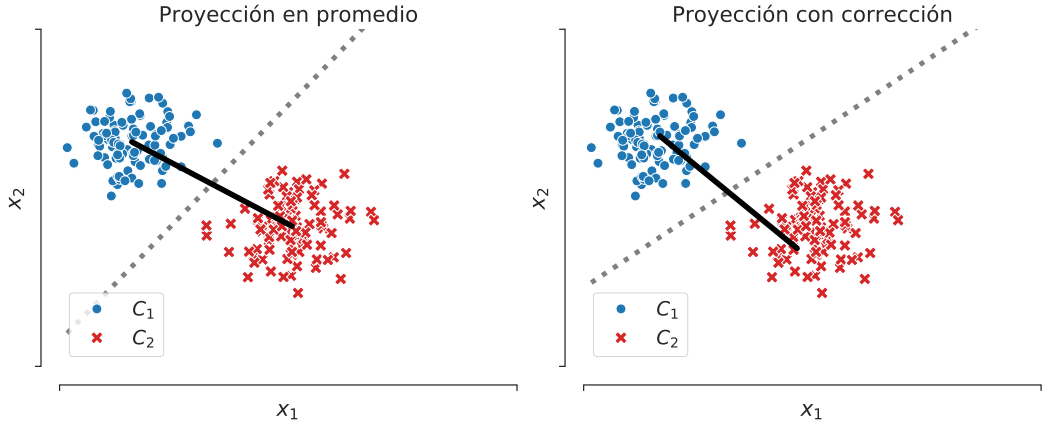


Fig. 16. Discriminador lineal considerando solo la media entre clases (izquierda) y su extensión mediante la corrección de Fisher que incorpora la varianza muestral de los datos.

4.4. Clasificación No Lineal: El Perceptrón

Las nociones básicas que hemos visto hasta ahora para lidiar con el problema de clasificación tienen dos problemas conceptuales. El primero es la falta de una métrica correcta para evaluar la bondad de nuestro modelo, esto es porque el criterios de mínimos cuadrados no es apropiado en la evaluación de una asignación de clases, donde no hay concepto de "más cerca", sino que solo correcto/incorrecto. Además, todos los enfoques considerados en esta sección hasta este punto no tienen una *función de versoimilitud* apropiada que conecte el modelo (variables latentes) con la clase (observación) de forma coherente. Por el contrario, hasta ahora hemos considerado que por un lado tenemos el modelo lineal para clasificación

y por otro lado está *nuestra decisión* de la clase en base a la salida del modelo linear, por ejemplo si $y(x) > b$ decimos que "x es clase $\mathcal{C}_1$".

La incorporación de esta función que conecta el modelo lineal con la clase, resulta en un *modelo lineal generalizado*, es decir, una modelo lineal conectado a una función no-lineal que llamaremos *función de enlace*. Sin embargo, el desafío más importante en esta construcción es que el modelo resultante ya no es lineal, ni en la entrada ni en los parámetros, pues una verosimilitud (función de enlace) lineal nunca nos llevará de un espacio de inputs (hemos asumido $\mathbb{R}^M$) al espacio de categorías $\{\mathcal{C}_1, \mathcal{C}_2, \ldots, \mathcal{C}_k\}$, es decir, necesitamos una no-linealidad "después" de la parte lineal, no antes como en el caso de los modelos lineales en lo parámetros vistos en la Sección 3.

Una forma de resolver estas problemáticas es mediante el uso del *Perceptrón* (Rosenblatt, 1958), un modelo de clasificación binario que tuvo mucha importancia en el área de reconocimiento de patrones. El Perceptrón consiste en una función no-lineal fija usada para transformar x en un vector de características[11] $\phi(x) \in \mathbb{R}^D$, que luego es usado para generar un modelo lineal *generalizado* con función de enlace no lineal $f(\cdot)$ de la siguiente forma:

$$y(x) = f(\theta^\top \phi(x)) \tag{148}$$

$$f(u) = \begin{cases} +1, & u \geq 0 \\ -1, & u < 0 \end{cases} \tag{149}$$

Opciones típicas para el vector $\phi(x)$ es concatenar la entrada con un intercepto para la recta, es decir $\phi(x) = [x^\top, 1]$ como vimos al comienzo del curso, o bien características no lineales como las vistas en la Sección 3. El Perceptrón entonces asigna x a la clase $\mathcal{C}_1$ si $y(x) = +1$ y asignará x a la clase $\mathcal{C}_2$ cuando $y(x) = -1$. Notemos que para el caso que ϕ es lineal, este es el mismo clasificador presentado en la Sección 4.1, pero en este caso el criterio para asignar la clase es **parte del modelo**.

Una condición para determinar el parámetro $\theta \in \mathbb{R}^D$ en base a un conjunto de datos $\mathcal{D} = \{(x_i, t_i)\}_{i=1}^N$, es que para $x \in \mathcal{C}_1$ $(t = 1)$, se cumpla que $\theta^\top \phi(x) > 0$, y para $x \in \mathcal{C}_2$ $(t = -1)$ se cumpla $\theta^\top \phi(x) < 0$. Usando el hecho que las etiquetas están representadas por la codificación $t \in \{1, -1\}$, ambas condiciones pueden ser cubiertas por la expresión:

$$\theta^\top \phi(x_n) t_n > 0, \quad \forall (x_n, t_n) \in \mathcal{D}. \tag{150}$$

Podemos entonces satisfacer esta restricción mediante el "criterio del perceptrón", el cual se basa en examinar los elementos de $\mathcal{D}$ que fueron clasificados incorrectamente. Este criterio asocia a los puntos clasificados correctamente error 0 y a los puntos mal clasificados error $-\theta^\top \phi(x) t > 0$. De esta forma, si denotamos $\mathcal{M}$ el conjunto de puntos mal clasificados, se debe minimizar la siguiente función objetivo:

$$J_{\mathrm{P}}(\theta) = -\sum_{i \in \mathcal{M}} \theta^\top \phi(x_i) t_i. \tag{151}$$

Observemos que al no usar directamente la función de activación perceptrón $f(\cdot)$, este costo es diferenciable (asumimos que $\phi(\cdot)$ lo es) y podemos minimizarlo mediante *descenso de gradiente estocástico*. Para cada elemento $x_i \in \mathcal{M}$, estas actualizaciones toman la siguiente forma:

$$\begin{aligned} \theta^{\tau+1} &= \theta^\tau - \eta \nabla J_{\mathrm{P}}(\theta) \\ &= \theta^\tau + \eta \phi(x_i) t_i. \end{aligned} \tag{152}$$

La variable $\tau \in \mathbb{N}$ simplemente indexa la secuencia $\{\theta_0, \theta_1, \ldots\}$, la cual parte de un θ_0 arbitrario y esperamos que converja al parámetro apropiado. El hyperparámetro η es una constante que se conoce como *tasa de aprendizaje*. Sin embargo, como la función perceptrón $y(x) = y(x, \theta)$ no cambia si θ se amplifica

[11] En este caso consideramos no-linealidad antes y después de la parte lineal, sin embargo, considerar la entrada como x o como $\phi(x)$ es equivalente en base a lo visto en los modelos lineales en los parámetros.

por una constante arbitraria, sin perdida de generalidad, podemos asumir que $\eta = 1$. Es importante notar que al actualizar el vector θ, el conjunto de puntos mal clasificados $\mathcal{M}$ va a cambiar, pues (esperamos que) en cada iteración los elementos del conjunto de puntos mal clasificados vaya disminuyendo.

La interpretación del algoritmo usado para ajustar el perceptrón es simple debido al uso del gradiente estocástico:

i) se recorre el conjunto de puntos de entrenamiento $\{x_n\}_{n=1}^N$,

ii) si el punto x_n fue clasificado correctamente el vector de pesos de mantiene igual

iii) six_n fue clasificado incorrectamente, el vector θ^τ es actualizado según la ec. (152) con $\eta = 1$ mediante

$$\theta^{\tau+1} = \theta^\tau + \phi(x_i)t_i. \tag{153}$$

Es decir, el parámetro θ está paso a paso modificado en la dirección de las características $\phi(x_i)$ con multiplicador ± 1 en base a la clase verdadera de x_i hasta que todos los puntos de $\mathcal{D}$ están bien clasificados.

4.5. Clasificación Probabilística: modelo generativo

Los modelos que hemos revisado hasta este punto son del tipo *discriminativo*, es decir, modelan directamente la función $f : x \mapsto c$. Con una interpretación probabilística, esto es equivalente a modelar la probabilidad condicional $\mathbb{P}(\mathcal{C}_k|x)$, es decir, dado que conozco el input (o características de) x, cuál es la distribución de probabilidad sobre las clases. Sin embargo, hemos considerado métodos determinísticos, que solo asignan probabilidad 1 a una sola clase.

Un paradigma alternativo es considerar es un enfoque *generativo*, en el cual modelamos dos objetos: en primer lugar la "probabilidad condicional de clase" la cual representa cómo distribuyen los valores de los inputs x cuando la clase es, por ejemplo, $\mathcal{C}_k$, denotada por $\mathbb{P}(x|\mathcal{C}_k)$. En segundo lugar las "probabilidades de clase", o el prior sobre clases, denotada $\mathbb{P}(\mathcal{C}_k)$. Luego, podemos calcular la densidad posterior sobre las clases dado un input x usando el Teorema de Bayes de acuerdo a

$$\mathbb{P}(\mathcal{C}_k|x) = \frac{\mathbb{P}(x|\mathcal{C}_k)\mathbb{P}(\mathcal{C}_k)}{\mathbb{P}(x)}. \tag{154}$$

Para el caso de 2 clases, es posible calcular la probabilidad de la clase $\mathcal{C}_1$ dado x de la forma:

$$\begin{aligned}
\mathbb{P}(\mathcal{C}_1|x) &= \frac{\mathbb{P}(x|\mathcal{C}_1)\mathbb{P}(\mathcal{C}_1)}{\mathbb{P}(x)} \\
&= \frac{\mathbb{P}(x|\mathcal{C}_1)\mathbb{P}(\mathcal{C}_1)}{\mathbb{P}(x|\mathcal{C}_2)\mathbb{P}(\mathcal{C}_2) + \mathbb{P}(x|\mathcal{C}_1)\mathbb{P}(\mathcal{C}_1)} \\
&= \frac{1}{1 + \frac{\mathbb{P}(x|\mathcal{C}_1)\mathbb{P}(\mathcal{C}_1)}{\mathbb{P}(x|\mathcal{C}_2)\mathbb{P}(\mathcal{C}_2)}} \\
&= \frac{1}{1 + \exp(-r)} = \sigma(r).
\end{aligned} \tag{155}$$

Donde hemos introducido la notación $r = r(x) = \ln\left(\frac{\mathbb{P}(x|\mathcal{C}_1)\mathbb{P}(\mathcal{C}_1)}{\mathbb{P}(x|\mathcal{C}_2)\mathbb{P}(\mathcal{C}_2)}\right)$ y la función logística definida mediante $\sigma(r) = \frac{1}{1+e^{-r}}$, la cual tiene propiedades que serán útiles en el entrenamiento, en particular,

$$\text{reflejo: } \sigma(-r) = 1 - \sigma(r) \tag{156}$$

$$\text{derivada: } \frac{d}{dr}\sigma(r) = \sigma(r)(1 - \sigma(r)) \tag{157}$$

$$\text{inversa: } r(\sigma) = \ln\left(\frac{\sigma}{1-\sigma}\right). \tag{158}$$

Observación 4.3. Si bien la expresión de la distribución condicional en la ec. (155) parece una presentación antojadiza para hacer aparecer la función logística (sigmoide), pues $r = r(x)$ puede ser cualquier cosa. Sin embargo, veremos que existe una elección particular de las distribuciones condicionales de clase que lleva a un r que es efectivamente lineal en x. En general, nos referiremos a este clasificador como **regresión logística** en dicho caso, es decir, cuando $r(x) = a^\top x + b$.

Podemos ahora considerar el caso de múltiples clases $\{\mathcal{C}_1, \ldots, \mathcal{C}_K\}$, donde un desarrollo similar al anterior resulta en:

$$\mathbb{P}(\mathcal{C}_i|x) = \frac{\mathbb{P}(x|\mathcal{C}_i)\mathbb{P}(\mathcal{C}_i)}{\sum_j \mathbb{P}(x|\mathcal{C}_j)\mathbb{P}(\mathcal{C}_j)} = \frac{\exp(s_i)}{\sum_{j\neq i}\exp(s_j)}, \tag{159}$$

donde hemos denotado $s_i = \mathbb{P}(x|\mathcal{C}_i)\mathbb{P}(\mathcal{C}_i)$. La función que aparece al lado derecho de la ec. (159) se conoce como *exponencial normalizada* o *softmax*, y corresponde a una generalización de la función logística a múltiples clases. Además, esta función tiene la propiedad de ser una aproximación suave de la función máximo y convertir cualquier vector $s = [s_1, \ldots, s_k]$ en una distribución de probabilidad, donde podemos hablar de "la probabilidad de ser clase $\mathcal{C}_k$".

4.5.1. Regresión logística

Analizaremos ahora los supuestos sobre el modelo generativo (i.e., las probabilidades de clase y condicionales) para encontrar un r –en la ec. (155)– que resulta en la bien conocida regresión logística. Consideraremos el caso binario donde las densidades condicionales de clase son Gaussianas multivariadas, dadas por

$$p(x|\mathcal{C}_k) \sim \mathcal{N}(\mu_k, \Sigma) = \frac{1}{(2\pi)^{\frac{D}{2}}|\Sigma|^{\frac{1}{2}}}\exp(-\frac{1}{2}(x-\mu_k)^\top\Sigma^{-1}(x-\mu_k)) \quad k \in \{1,2\}. \tag{160}$$

En cuyo caso obtenemos r en ec. (155) dado por

$$r = \ln\left(\exp\left(-\frac{1}{2}(x-\mu_1)^\top\Sigma^{-1}(x-\mu_1) + \frac{1}{2}(x-\mu_2)^\top\Sigma^{-1}(x-\mu_2)\right)\right) = a^\top x + b, \tag{161}$$

donde hemos usado la notación

$$a = \Sigma^{-1}(\mu_1 - \mu_2) \tag{162}$$

$$b = \frac{1}{2}(\mu_1^\top\Sigma^{-1}\mu_1 + \mu_2^\top\Sigma^{-1}\mu_2) + \ln\left(\frac{p(\mathcal{C}_1)}{p(\mathcal{C}_2)}\right). \tag{163}$$

Lo cual nos entrega la regresión logística (lineal) para el caso binario, donde al incorporar la expresión anterior en la ec. (155) obtenemos

$$p(\mathcal{C}_k|x) = \sigma(a^\top x + b) = \frac{1}{1+\exp(a^\top x + b)}. \tag{164}$$

Observación 4.4. El modelo de clasificación binaria en la ec. 164 es conocido como *regresión logística* y es la consecuencia de asumir un modelo generativo Gaussiano con distintas medias pero la misma varianza para una de las clases. Como consecuencia, la región de decisión se encuentra imponiendo que la probabilidad de ser clase $\mathcal{C}_1$ sea $1/2$ (y por ende, ser $\mathcal{C}_2$ también tiene probabilidad $1/2$), lo cual se tiene para

$$a^\top x + b = 0. \tag{165}$$

Ahora que hemos definido el modelo para nuestro problema de clasificación, aflora naturalmente la siguiente pregunta: ¿Cómo ajustar los parámetros de las condicionales a la clase y priors respectivamente? Para esto, reiteremos que los parámetros del modelos serán los de la probabilidad de clase $p(\mathcal{C}_k)$ y de la probabilidades condicionales de clase $p(x|\mathcal{C}_k)$. Respectivamente:

- Probabilidad de clase
$$p(\mathcal{C}_1) = \pi, \quad p(\mathcal{C}_2) = 1 - \pi, \tag{166}$$
es decir, un parámetro π

- Probabilidad condicional de clase
$$p(x|\mathcal{C}_k) = \mathcal{N}(\mu_k, \Sigma); k = 1, 2, \tag{167}$$
es decir, parámetros $\mu_1 \in \mathbb{R}^M, \mu_2 \in \mathbb{R}^M, \Sigma \in \mathbb{R}^M \times \mathbb{R}^M$ o, equivalentemente, $M+M+M(M+1)/2 = M(M+5)/2$ parámetros escalares.

Denotaremos todos los parámetros mediante el parámetro agregado $\theta = \{\mu_1, \mu_2, \Sigma\}$.

Realizaremos el entrenamiento del modelo, i.e., encontrar θ en base a un conjunto de datos $\mathcal{D}$, mediante el método de máxima verosimilitud. Para esto, notemos que solo hemos definido "la probabilidad de ser clase $\mathcal{C}_1$", noción que extendemos usando la codificación de clases $t \in \{0, 1\}$. Con esta codificación, donde la observación (x_i, t_i) corresponde a clase $\mathcal{C}_1$ con $t_i = 1$ y a clase $\mathcal{C}_2$ con $t = 0$, podemos expresar la verosimilitud con una observación mediante:
$$L_i(\theta) = p(x_i, t_i|\theta) = p(x_i, \mathcal{C}_1|\theta)^{t_i} p(x_i, \mathcal{C}_0|\theta)^{1-t_i}. \tag{168}$$
Esta expresión equivale a la probabilidad de ser clase $\mathcal{C}_1$ cuando $t_i = 1$ y a la probabilidad de ser clase $\mathcal{C}_0$ cuando $t_i = 0$. Consecuentemente, para un conjunto de datos $\mathcal{D}$ de la forma
$$T = (t_1, t_2, \ldots, t_N) \in \{0, 1\}^N \tag{169}$$
$$X = (x_1, x_2, \ldots, x_N), \tag{170}$$
podemos escribir la verosimilitud mediante $L(\theta) = p(X, T|\pi, \mu_1, \mu_2, \Sigma)$
$$\begin{aligned} L(\theta) &= \prod_{i=1}^{N} p(x_i, t_i|\pi, \mu_1, \mu_2, \Sigma) \\ &= \prod_{i=1}^{N} p(x_i, \mathcal{C}_1)^{t_i} p(x_i, \mathcal{C}_0)^{1-t_i} \\ &= \prod_{i=1}^{N} (\pi \mathcal{N}(x_i|\mu_1, \Sigma))^{t_i} ((1-\pi)\mathcal{N}(x_i|\mu_2, \Sigma))^{1-t_i}. \end{aligned} \tag{171}$$

Al igual que en el escenario de regresión lineal, para proceder con la optimización nos enfocamos en la log-verosimilitud, dada por:
$$l(\theta) := \log L(\theta) = \sum_{i=1}^{N} t_i(\log(\pi) + \log(\mathcal{N}(x_i|\mu_1, \Sigma))) + (1 - t_i)(\log(1 - \pi) + \log(\mathcal{N}(x_i|\mu_2, \Sigma))). \tag{172}$$

Aplicando entonces las condiciones de primer orden, tenemos que

- **1)** Con respecto a π:
$$\begin{aligned} &\frac{\partial \log(L)}{\partial \pi} = \sum_{i=1}^{N} \frac{t_i}{\pi} + \frac{1 - t_i}{1 - \pi} = 0 \\ &\Rightarrow \quad (i - \pi) \sum_{i=1}^{N} t_i = \pi \sum_{i=1}^{N} (1 - t_i) \\ &\Rightarrow \quad \sum_{i=1}^{N} t_i = \pi N \quad \Rightarrow \quad \pi = \frac{\sum_{i=1}^{N} t_i}{N} = \frac{N_1}{N_1 + N_2} \end{aligned} \tag{173}$$

- **2)** Con respecto a μ_1:

$$\begin{aligned}\frac{\partial \log(L)}{\partial \mu_1} &= \sum_{i=1}^{N} t_i \frac{\partial}{\partial \mu_1}(-\frac{1}{2}(x_i-\mu_1)^\top \Sigma^{-1}(x_i-\mu_1)) \\ &= \sum_{i=1}^{N} t_i(\Sigma^{-1}(x_i-\mu_1)) = \Sigma^{-1}\sum_{i=1}^{N} t_i(x_i-\mu_1) = 0 \\ \Rightarrow \quad \sum_{i=1}^{N} t_i x_i = \mu_1 \sum_{i=1}^{N} t_i \quad &\Rightarrow \quad \mu_1 = \frac{1}{N_1}\sum_{i=1}^{N} t_i x_i \end{aligned} \tag{174}$$

De forma análoga:

$$\mu_2 = \frac{1}{N_2}\sum_{i=1}^{N}(1-t_i)x_i \tag{175}$$

Observación 4.5. La ec. (173) revela que el parámetro óptimo π es precisamente la razón entre la cantidad de elementos de la clase $\mathcal{C}_1$ y el total de datos, esto es porque π es la probabilidad de ser clase $\mathcal{C}_1$; lo mismo aplica para $(1-\pi)$ y clase $\mathcal{C}_0$. Además, de las ecs. (174)-(175) vemos también que el estimador de MV para la media de las clases es la media muestral de los datos disponibles por cada clase. Queda la siguiente pregunta entonces: ¿cuál es el estimador de MV de Σ? $\rightarrow$ tarea.

4.5.2. Regresión Logística v/s Modelo Generativo

Recordemos que los supuestos tonados sobre el modelo generativo para el problema de clasificación resultaron en:

$$p(\mathcal{C}_1|x) = \sigma(w^\top x) = \frac{1}{1+e^{-w^\top x}}, \tag{176}$$

donde por claridad de notación hemos elegido la representación lineal $(w^\top x)$ y no afín $(a^\top x + b)$.

En el caso anterior se ha entrenado el modelo generativo completo, es decir, $\pi, \mu_1, \mu_2, \Sigma$, lo cual tiene la ventaja de tener solución en forma cerrada, sin embargo, puede ser innecesario cuando solo necesitamos conocer el peso w en la ecuación anterior. Otra forma de entrenar la regresión logística es considerar el modelo discriminativo en la ec. (176) y optimizar directamente la verosimilitud sobre este modelo para encontrar w; en vez de todos los parámetros $\pi, \mu_1, \mu_2, \Sigma$ del modelo generativo.

Calculemos la verosimilitud de la regresión logística con datos $\mathcal{D} = \{(x_i, t_i)\}_{i=1}^N$, para hacer la notación más compacta denotamos $\sigma_i = \sigma(w^\top x_i)$. Entonces:

$$p(t_{1:N}|x_{1:N}, w) = \prod_{i=1}^{N} p(t_i|x_i, w) = \prod_{i=1}^{N} \sigma_i^{t_i}(1-\sigma_i)^{1-t_i}. \tag{177}$$

Con lo que la log-verosimilitud está dada por

$$l(w) = \sum_{i=1}^{N} t_i \log(\sigma_i) - (1-t_i)\log(1-\sigma_i). \tag{178}$$

Notemos que este problema de optimización no exhibe una solución en forma cerrada, con lo que podemos resolverlo mediante gradiente estocástico, para lo cual es necesario calcular el gradiente de $l(w)$ respecto a w:

$$\nabla_w l(w) = \sum_{i=1}^{N}(t_i - \sigma_i)x_i, \tag{179}$$

lo cual nos da una regla de ajuste $\theta \mapsto \theta - \eta \sum_{i=1}^{N} (\sigma_i - t_i) x_i$, o bien

$$\theta \mapsto \theta + \eta (t_i - \sigma_i) x_i, \tag{180}$$

si tomamos los datos de "a uno".

Observación 4.6. Notemos que la regla de ajuste del parámetro θ en la ec. (180) recuerda el ajuste del Perceptrón en la ec. (152), donde los puntos mal clasificados se agregan como características al vector θ. Sin embargo, a diferencia del perceptrón, incluso los elementos bien clasificados (σ cercano a 1 ó 0) tomarán parte en la actualización, consecuentemente forzando una fuerte componente de sobreajuste haciendo tender θ a infinito.

5. Selección de Modelo

Supongamos que tenemos un montón de datos, de los cuales podemos ajustar un millón de modelos, tanto como la imaginación nos limite. Entonces ¿Con cuál te quedarías al final? Cuando nos enfrentamos a la decisión de elegir un modelo usualmente lo que más interesa es la preciso del modelo a la hora de generalizar. El problema de solo buscar precisión es claro: *overfitting* (ver figura 17).

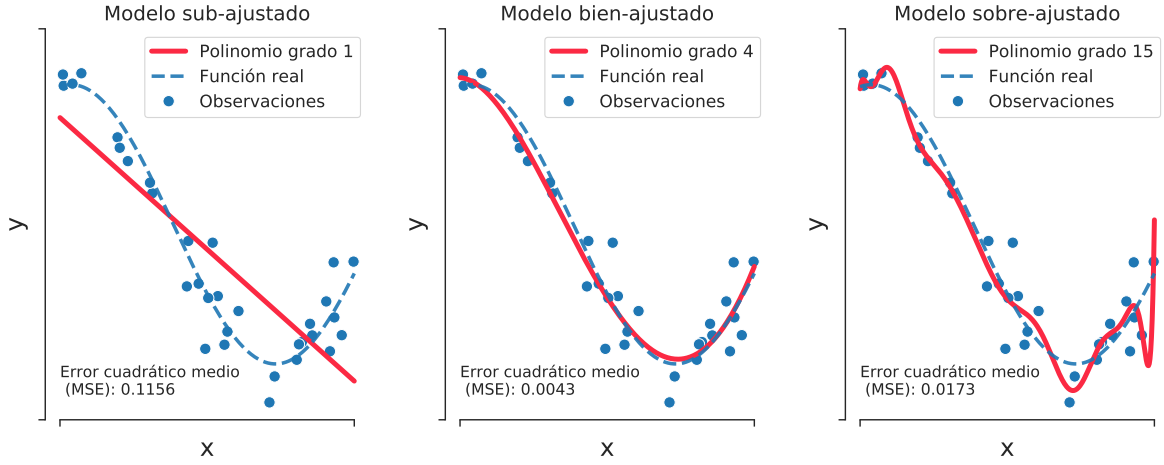


Fig. 17. Ejemplos de sub, sobre y correcto ajuste.

La elección de modelo es uno de los problemas más recurrentes en análisis de datos, un ejemplo es la elección de las variables para la regresión ¿Por qué elegir un modelo cuadrático y no uno cúbico? ¿Por qué no incorporar todas las variables cruzadas $(X_1X_2, X_1X_3, ..., X_nX_m)$? Estas preguntas son fácilmente abordables si volvemos a la realidad y queremos correr los modelos en nuestros computadores: Incorporar más variables usualmente aumenta la complejidad computacional y con esto el tiempo de cómputo.

Bajo la filosofía anterior, encontrar un *trade-off* entre *bias* (la flexibilidad del modelo de ajustarse a los datos) y performance sería lo ideal. Es por esta razón que crear un estadístico que pudiera expresar correctamente el compromiso precisión - flexibilidad ayudaría en la elección de modelo. Esto último es lo que busca relizar el *Akaike Information Criterion* (AIC) y el *Bayesian Information Criterion* (BIC) los cuales son dos enfoques distintos para abordar esta problemática.

5.1. Criterio de Información Akaike

La siguiente aproximación es válida cuando $N \to \infty$:

$$-2\mathbb{E}[\log P_{\hat{\theta}}(Y)] \approx \frac{-2}{N}\mathbb{E}[\log L] + 2\frac{d}{N},$$

en donde d es la cantidad de parámetros del modelo y L es la función de verosimilitud definida por:

$$L = \prod_{i=1}^{N} P(y_i|\hat{\theta}). \tag{181}$$

De esta manera se define el estadístico AIC de la forma:

$$AIC(M) = 2d - 2\log(L) \tag{182}$$

Otros autores definen el estadístico como:

$$AIC(M) = 2\frac{d}{N} - 2\frac{log(L)}{N}. \tag{183}$$

Donde N es la cantidad de observaciones.

Ejemplo: Si tenemos un conjunto de modelos $f_\alpha(x)$ indexados por un parámetro α de modo que el modelo α sigue un modelo gaussiano, es decir:

$$\log(L) = -\sum_i (y_i - \hat{f}(x_i))^2/(2\sigma_e^2),$$

entonces el estimador AIC se puede escribir como

$$\text{AIC}(M) = 2d(\alpha) - \overline{err}(\alpha), \tag{184}$$

en donde la función $d(\alpha)$ denota una medida de la complejidad del modelo y $\overline{err}(\alpha)$ denota el error promedio del modelo α.

5.2. Criterio de Información Bayesiano

Nuevamente, si nuestro setting nos permite obtener el máximo de la función de verosimilitud. Podemos utilizar el *Bayesian Information Criterion* (BIC) el cual está definido por:

$$\text{BIC}(M) = log(N)d - 2\log(L) \tag{185}$$

En donde N es la cantidad de observaciones, d es la cantidad de parámetros del modelo y L es el máximo de la función de verosimilitud. Este criterio también es conocido como *Schwarz criterion*. Este estadístico se dice bayesiano por que se obtiene de realizar una aproximación de laplace sobre sobre $\log \mathbb{P}(X|M)$ obteniéndose:

$$\log \mathbb{P}(X|M) \approx \log \mathbb{P}(X|\hat{\theta}_m, M) - \frac{d_m}{2}(\log(N) - \log(2\pi)) \tag{186}$$

Donde $\hat{\theta}_m$ es el máximo del estimador de verosimilitud. Luego para N muy grande se observa que la expresión $-2\log \mathbb{P}(X|M) = BIC(M)$.

Ejemplo: Asumiendo que el modelo sigue un modelo gaussiano, es decir:

$$\log(L) = -\sum_i (y_i - \hat{f}(x_i))^2/(2\sigma_e^2),$$

se tiene que el estadístico BIC está dado por:

$$BIC(M) = \frac{N\overline{err}(M)}{\sigma_e^2} + log(N)d \tag{187}$$

5.3. Evaluación y comparación de modelos

Existen varias formas de evaluar un modelo, una de ella podría ser simplemente evaluar la precisión de sus predicciones. A veces fijarse es natural fijarse en la precisión, como en los problemas de pronóstico. Otras veces la precisión es importante para evaluar diferentes modelos y elegir uno de ellos. En esta sección presentaremos dos maneras distintas de evaluar modelos, cada forma sirve en distintos escenarios, los cuales se discutirán a través de la predicción puntal, que resume la predicción de un conjunto de datos en una solo valor.

5.3.1. Error cuadrático medio

El ajuste del modelo a nuevos datos se puede resumir en una predicción puntual llamada error cuadrático medio, el cual está definido por:

$$MSE(\theta) = \frac{1}{n}\sum_{i=1}^{N}(y_i - \mathbb{E}(y_i|\theta))^2 \tag{188}$$

o su versión ponderada:

$$MSE(\theta) = \frac{1}{n}\sum_{i=1}^{N}\frac{(y_i - \mathbb{E}(y_i|\theta))^2}{\mathbb{V}\text{ar}(y_i|\theta)} \tag{189}$$

5.3.2. log-densidad predictiva o log-verosimilitud

Otra forma de realizar esta evaluación es utilizando el estadístico *log-densidad preditiva* $\log p(y|\theta)$ el cual es proporcional a error cuadrático medio si el modelo es normal con varianza constante. Estudiaremos el caso de un solo punto, para luego extrapolar a más de un punto.

Predictive accuracy para un punto: Sea f el modelo real, y las observaciones (es decir, una realización del dataset y de la distribución $f(y)$), y llamaremos $\tilde{y}$ a la data futura o un dataset alternativos que podemos ver. El ajuste predictivo out-of-sample para un nuevo punto $\tilde{y}_i$ está dado por:

$$\log p_{\text{post}}(\tilde{y}_i) = \log \mathbb{E}[p(\tilde{y}_i|\theta)] = \log \int p(\tilde{y}_i|\theta)p_{\text{post}}(\theta)d\theta \tag{190}$$

Promedio de las distribuciones para un punto: Al tener un dato nuevo $\tilde{y}_i$ entonces se puede calcular el la log-densidad predictiva (elpd, por su sigla en inglés) para el nuevo punto:

$$\begin{aligned}\text{elpd} &= \mathbb{E}_f[\log p_{\text{post}}(\tilde{y}_i)] \\ &= \int \log p_{\text{post}}(\tilde{y}_i) f(\tilde{y}_i) d\tilde{y}\end{aligned} \tag{191}$$

Promedio de las distribuciones para datasets futuros: Como usualmente, no se tiene solo un punto, se debe realizar la suma sobre el conjunto de puntos, calculando así la log-densidad predicitiva puntual (elppd, por su sigla en inglés).

$$\text{elppd} = \sum_{i=1}^{N}\mathbb{E}_f[\log p_{\text{post}}(\tilde{y}_i)] \tag{192}$$

En la práctica, como siempre se tiene la distribución de todos los modelos y la expresión anterior requiere de esto, se suele calcular el estadístico sobre una estimación de un modelo $\hat{\theta}$ (como por ejemplo, el máximo de la función de verosimilitud):

$$\text{elppd}|\hat{\theta} = \sum_{i=1}^{N}\mathbb{E}_f[\log p_{\text{post}}(\tilde{y}_i|\hat{\theta})] \tag{193}$$

Finalmente, una última extensión de este estadístico es cuando se puede tener *draws* de la posterior, es decir, tenemos $\{\theta^s\}_{s=1}^{S}$, entonces el lppd computado es:

$$\text{computed lppd} = \sum_{i=1}^{N}\left(\frac{1}{S}\sum_{s=1}^{S}p(y_i|\theta^s)\right) \tag{194}$$

5.3.3. Otros métodos

Existen otros estadísticos o métodos que se pueden utilizar para comparar modelos.

- **Deviance Information Criterion:** Se podría decir que es una versión bayesiana de AIC, donde se realizan dos cambios principales 1) se cambia la estimación del estimador de máxima verosimilitud, por el promedio de la posterior y 2) se reemplaza k con una perrción de sesgo de los datos. Formalizando:

$$\text{DIC} = -2\log(y|\hat{\theta}_{Bayes}) + 2p_{\text{DIC}}, \tag{195}$$

donde p_{DIC} está dado por:

$$p_{\text{DIC}} = 2(\log(p(y|\hat{\theta}_{Bayes}) - \mathbb{E}_{\text{post}}(\log(p(y|\theta)). \tag{196}$$

Nuevamente, puede haber un versión *computada* que está dada por:

$$\text{computed } p_{\text{DIC}} = 2[\log(p(y|\hat{\theta}_{Bayes}) - \frac{1}{S}\sum_{s=1}^{S}\log(p(y|\theta^s))]. \tag{197}$$

- **Watanabe-Akaike o widely available information criterion:** WAIC es un forma aún más bayesiana para abordar el problema de asginación de evaluación.

$$\text{WAIC} = -2\text{lppd} + 2p_{\text{WAIC}} \tag{198}$$

donde hay dos formas de calcular p_{WAIC}.

$$p_{\text{WAIC1}} = 2\sum_{i=1}^{n}(\log(\mathbb{E}_{\text{post}}[p(y_i|\theta)]) - \mathbb{E}_{\text{post}}[\log(p(y_i|\theta))) \tag{199}$$

$$p_{\text{WAIC2}} = \sum_{i=1}^{n}(\mathbb{V}\text{ar}_{\text{post}}[\log(p(y_i|\theta))])\,. \tag{200}$$

- **Leave-one-out cross-validation:** También se puede usar validación cruzada bayesiana para estimar $-2\text{lppd} + 2\text{lppd}_{\text{loo-cv}}$ la cual sirve como estadístico de comparación:

$$\text{lppd}_{\text{loo-cv}} = \sum_{i=1}^{n}\log p_{\text{post}(-i)}(y_i) \tag{201}$$

que se calcula como:

$$\text{computed lppd}_{\text{loo-cv}} = \sum_{i=1}^{n}\log\left(\sum_{s=1}^{S}\frac{1}{S}p(y_i|\theta^{is})\right) \tag{202}$$

también se puede calcular a versión insesgada, la cual no se utiliza mucho pero se presenta por completitud:

$$\text{lppd}_{\text{cloo-cv}} = \text{lppd}_{\text{loo-cv}} + b. \tag{203}$$

Donde b está dado por:

$$b = \text{lppd}_{-i} - \overline{\text{lppd}_{-i}}, \tag{204}$$

con

$$\overline{\text{lppd}_{-i}} = \frac{1}{n}\sum_{i=1}^{n}\sum_{j=1}^{n}\log p_{\text{post}(-i)}(y_j) \tag{205}$$

$$\text{computed } \overline{\text{lppd}_{-i}} = \frac{1}{n}\sum_{i=1}^{n}\sum_{j=1}^{n}\log\left(\frac{1}{S}\sum_{s=1}^{S}p(y_j|\theta^{is})\right). \tag{206}$$

5.4. Promedio de Modelos

Hay veces que no queremos elegir solo un modelo, puesto que quizás nos interesan las estimaciones de dos o más modelos. Para estos casos, se puede utilizar la técnica de *Model Averaging*, la cual consiste en simplemente ponderar los modelos de la siguiente forma:

$$\hat{\mu} = \sum_{s\in A} c(s)\hat{\mu}_s \tag{207}$$

donde A es el conjunto de todos los modelos. Además, se debe cumplir que:

$$\sum_{s\in A} c(s) = 1 \tag{208}$$

Notemos que la elección:

$$c(s) = \begin{cases} 1 & \text{modelo con mejor criterio.} \\ 0 & \sim \end{cases} \tag{209}$$

es una elección de pesos posible.

5.4.1. Elección de pesos mediante softmax

La principal dificultad para elegir los pesos, está en que se debe asegurar 1) la suma de los pesos sea unitaria, 2) que los pesos sean todos positivos. Una forma de asegurar esto es aplicar una función f positiva sobre una función g de *score*, de modo que:

$$\sum_{s\in A}(f\circ g)(s) > 0 \tag{210}$$

Así los pesos quedan se pueden definir cómo:

$$c_f(s) = \frac{(f\circ g)(s)}{\sum_{z\in A}(f\circ g)(s)} \tag{211}$$

La elección más común de para $f(x) = \text{softmax}(x)$, mientras que para la función de *score* se puede utilizar las recién estudiadas AIC o BIC.

$$c_{AIC}(s) = \frac{\exp\{\frac{1}{2}\text{AIC}_s\}}{\sum_{z\in A}\exp\{\frac{1}{2}\text{AIC}_z\}} \qquad c_{BIC}(s) = \frac{\exp\{\frac{1}{2}\text{BIC}_s\}}{\sum_{z\in A}\exp\{\frac{1}{2}\text{BIC}_z\}} \tag{212}$$

5.4.2. Bayesian Model Averaring

Supongamos que tenemos $\{M_j\}_{j=1}^k$ modelos, de los cuales todos realizan estimaciones razonables de una cantidad μ, dado los datos y. Para realizar el modelo promedio bayesiano de estos modelos es necesario encontrar la distribución posterior de μ dado los datos y sin condicionar a un modelo en específico. De este modo, es necesari:

1. Cada modelo tenga los mismos puntos de interpolación.
2. Prior para cada modelo $p(M_j)$.
3. Prior sobre los parámetros de los modelos $\pi(\theta_i|M_j)$[12].

Luego se puede calcular la función de verosimilitud para los parámetros:

$$\mathcal{L}_{\theta_j}(y) = p(y|\theta_j) \tag{213}$$

Luego se marginaliza sobre los parámetros para obtener la función de verosimilitud del modelo:

$$\lambda_{M_j}(y) = p(y|M_j) \tag{214}$$

$$= \int \mathcal{L}_{\theta_j}(y)\pi(\theta_i|M_j)d\theta_j \tag{215}$$

La expresión (215) es también densidad marginal de las observaciones. Con esto la densidad posterior del modelo es:

$$p(M_j|y) = \frac{p(M_j)\lambda_{M_j}(y)}{\sum_{l=1}^k p(M_l)\lambda_{M_l}(y)} \tag{216}$$

Finalmente, se puede obtener la distribución posterior para μ asumiendo que el modelo M_j es verdadero ($\pi(\mu|M_j, y)$), y así, la distribución posterior buscada:

$$\pi(\mu|y) = \sum_{j=1}^{k} p(M_j|y)\pi(\mu|M_j, y) \tag{217}$$

[12]Se puede obtener mediante MCMC.

6. Redes Neuronales - Editado

6.1. Introducción y Arquitectura

6.1.1. Conceptos Básicos

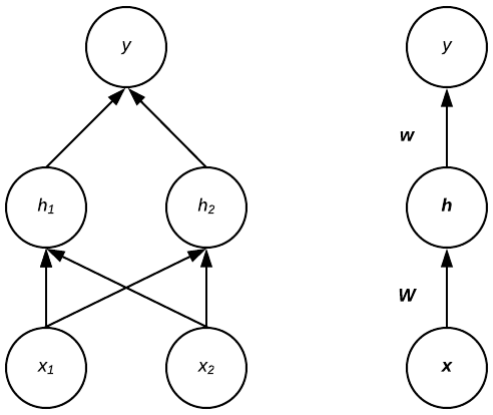
Los modelos escenciales de redes neuronales se conocen como **feedforward neural networks**, o **multilayer perceptrons** (MLPs). Una red neuronal busca aproximar una función f^*. Una red feedforward define un mapping $\boldsymbol{y} = f(\boldsymbol{x}; \boldsymbol{\theta})$ y aprende los parámetros $\boldsymbol{\theta}$ que resultan en la mejor aproximación posible.

Estos modelos se conocen como redes ya que que típicamente son el resultado de composiciones sucesivas de varios tipos de funciones. Por ejemplo, sean $f^{(1)}$, $f^{(2)}$ y $f^{(3)}$ funciones, estas se 'conectan' para formar $f(\boldsymbol{x}) = f^{(3)}(f^{(2)}(f^{(1)}(\boldsymbol{x})))$, se dice que $f^{(1)}$ es la primera **capa** de la red, $f^{(2)}$ la segunda y $f^{(3)}$ la tercera. El largo de esta cadena define la **profundidad** de la red. El uso de redes neuronales con múltiples capas se conoce como **deep learning**, aunque este término se usa para problemas más específicos de aprendizaje de máquinas usando redes neuronales profundas (ver sección 3.5).

Durante el entrenamiento de una red neuronal se busca que $f(\boldsymbol{x}) \approx f^*(\boldsymbol{x})$, donde la data de entrenamiento provee aproximaciones ruidosas de $f^*(\boldsymbol{x})$. Cada dato $\boldsymbol{x}$ viene acompañado de una etiqueta, $y \approx f^*(\boldsymbol{x})$. Los datos de entrenamiento indica qué debe reproducir la red en su última capa, lo cual debe ser un valor aproximado de y. El comportamiento del resto de las capas no tiene una interpretación tan directa como la última capa, por lo que el algoritmo de aprendizaje debe decidir cómo adaptar las capas para que el output producido por la red sea lo más cercano posible a y, la etiqueta, para cada punto $\boldsymbol{x}$. Es por esto que estas capas intermedias se denominan **capas ocultas**.

Estas redes se llaman *neuronales* por su inspiración neurocientífica. Cada capa oculta de la red corresponde a un vector. Se podría pensar que cada elemento de estos vectores tiene un rol similar al de una neurona, en vez de pensar cada capa como una función que produce un mapping de vector a vector, podría considerarse que una capa consiste en muchas **unidades** que actúan en paralelo, cada una representando un mapping de vector a escalar. La elección de las funciones $f^{(i)}(\boldsymbol{x})$, conocidas como **funciones de activación**, en varios casos ha sido guiada por observaciones sobre el comportamiento de neuronas biológicas.

Fig. 18. Ejemplo de una red neuronal de 1 capa: (*izquierda*) Se muestra la red con inputs x_1 y x_2, que luego pasan a la capa oculta para producir el output y. (*derecha*) Misma red con una representación vectorial de las capas. La matriz $\boldsymbol{W}$ describe el mapping de $\boldsymbol{x}$ a $\boldsymbol{h}$, y la matriz w el mapping de $\boldsymbol{h}$ a y.



Como se puede apreciar en la figura, los coeficientes $\boldsymbol{W}$ y w se usan para producir el output para la siguiente capa (denominados **pesos**), por lo que la primera operación de esta red será entregar h_1 y h_2 mediante la transformación $\boldsymbol{W}^T\boldsymbol{x} + \boldsymbol{b}^{(1)}$. Luego, esta red aplica $w^T\boldsymbol{h} + b^{(2)}$ para producir el output

y. Los coeficientes $\boldsymbol{b}^{(1)}$ y $b^{(2)}$ se conocen como términos de **bias** (sesgo). Todos los parámetros de la red neuronal, los pesos y *bias*, serán agrupados en el término $\boldsymbol{\theta}$.

6.1.2. Función de Costos, Unidades de Output y la Formulación como un Problema Probabilístico

Una de las principales diferencias entre los modelos lineales antes vistos y una red neuronal, es que el uso de ciertas funciones de activación hacen que la función de costos no sea convexa, esto hace que el entrenamiento realizado en base a descenso de gradiente no entregue garantías de que se alcanzará el óptimo global, o una buena solución en términos generales, ya que el algoritmo podría estancarse en un optimo local que entregue resultados pobres.

En la mayoría de los casos, el modelo paramétrico define una distribución $p(y|\boldsymbol{x};\boldsymbol{\theta})$, por lo que los parámetros del modelo se estimarán usando máxima verosimilitud, así, se optimizará la log-verosimilitud negativa, es decir, la **función de costos** a usar será la **cross-entropy**:

$$J(\boldsymbol{\theta}) = -\mathbb{E}_{\boldsymbol{x},\boldsymbol{y}\sim\hat{p}_{\text{data}}}(\log p_{\text{modelo}}(\boldsymbol{y}|\boldsymbol{x})) \tag{218}$$

De esta forma la elección de la **unidad de output** definirá la forma que toma la función de costos, pero no será necesario definir una función especifica para cada problema; en general siempre se resolverá el problema por máxima verosimilitud. La elección en la unidad de output dependerá del tipo de problema que se quiera resolver. Cuando se quiera retornar la media de una distribución Gaussiana condicional, $p(\boldsymbol{y}|\boldsymbol{x}) = \mathcal{N}(\boldsymbol{y}|\hat{\boldsymbol{y}};\boldsymbol{I})$, la unidad de output deberá ser un modelo lineal, $\hat{\boldsymbol{y}} = \boldsymbol{W}^T\boldsymbol{h} + \boldsymbol{b}$, en donde $\boldsymbol{h}$ es el output de la red que proviene de todas las capas ocultas anteriores. En este caso, maximizar la log-verosimilitud es equivalente a minimizar el error cuadrático medio, por esta razón este tipo de output es adecuada para un problema de regresión.

Cuando el problema objetivo es clasificación binaria, una unidad de output apropiada es la función sigmoidal. Para resolver el problema de máxima verosimilitud, se modela utilizando una distribución Bernoulli en y condicional en $\boldsymbol{x}$. Una unidad de output sigmoidal se define como $\hat{y} = \sigma(\boldsymbol{w}^T\boldsymbol{h} + b)$. Esto entrega como output $P(y = 1|\boldsymbol{x})$, por lo que se podrá decidir a qué clase pertenece y basado en el output de la red, esto se interpreta como la probabilidad de que la observación sea de la clase 1.

Para un problema de clasificación multiclase, la unidad de output apropiada es la generalización de la función sigmoidal, conocida como softmax. El problema es predecir a cuál de n clases pertenece y, por lo que se requiere producir un vector $\hat{\boldsymbol{y}}$, con $\hat{y}_i = P(y = i|\boldsymbol{x})$, por lo que se usará una distribución multinoulli. Primero, una capa lineal predice las log-probabilidades no normalizadas, $\boldsymbol{z} = \boldsymbol{W}^T\boldsymbol{h} + \boldsymbol{b}$, y luego se aplica la función softmax para obtener los valores de $\hat{\boldsymbol{y}}$ antes descritos:

$$\log \text{softmax}(\boldsymbol{z})_i = \log\frac{\exp(z_i)}{\sum_{j=1}^{n}\exp(z_j)} = z_i - \log\sum_{j=1}^{n}\exp(z_j) \tag{219}$$

6.1.3. Estructura Interna de la Red

Las capas escondidas proveen a la red neuronal la flexibilidad necesaria para aprender funciones extremadamente complejas, esto mediante el uso de activaciones no lineales. En varios casos estas funciones son incluso no diferenciables, lo que llevaría a pensar de que no son válidas para ser usadas en conjunto con algoritmos que usan descenso de gradiente, pero en la práctica tienen un desempeño suficientemente bueno para ser usadas en tareas de aprendizaje de máquinas. En general, las funciones de activación para las unidades escondidas toman como input un vector $\boldsymbol{x}$ para el cual obtienen una transformación afín $\boldsymbol{z} = \boldsymbol{W}^T\boldsymbol{x} + \boldsymbol{b}$, para luego aplicar una transformación no lineal $g(\boldsymbol{z})$. Las unidades escondidas más comunes son las **rectified linear units** o **ReLUs**, $g(\boldsymbol{z}) = \max(0, \boldsymbol{z})$, y sus generalizaciones como **leaky ReLU**, $g(\boldsymbol{z})_i = \max(0, \boldsymbol{z}_i) + \alpha_i\min(0, \boldsymbol{z}_i)$, con α_i una constante pequeña como 0.01; la función sigmoidal $g(\boldsymbol{z}) = \sigma(\boldsymbol{z})$; y la tangente hiperbólica $g(\boldsymbol{z}) = \tanh(\boldsymbol{z}) = 2\sigma(2\boldsymbol{z}) - 1$. A diferencia de las funciones

lineales por partes, las unidades sigmoidales se saturan en la mayor parte de su dominio (su gradiente se aproxima a 0) haciendo imposible el aprendizaje por el gradiente cuando esto ocurre, por lo que su uso como unidades escondidas ha sido desalentado.

6.1.4. Diseño de Arquitectura y Teorema de Aproximación Universal

La **arquitectura** de una red neuronal se refiere a la totalidad de su estructura: la cantidad de capas, la cantidad de unidades escondidas, la conexión entre las unidades, etc. Bajo la estructura mostrada para una red feedforward, la primera capa y la segunda capa son de la forma:

$$\begin{aligned} \boldsymbol{h}^{(1)} &= g^{(1)}(\boldsymbol{W}^{(1)T}\boldsymbol{x} + \boldsymbol{b}^{(1)}) \\ \boldsymbol{h}^{(2)} &= g^{(1)}(\boldsymbol{W}^{(2)T}\boldsymbol{h}^{(1)} + \boldsymbol{b}^{(2)}) \end{aligned} \tag{220}$$

En una arquitectura de este tipo, la principal consideración respecto al diseño es la profundidad y ancho (cantidad de unidades por capa) de cada capa. Una red con solo 1 capa escondida puede ser suficiente para ajustar el set de entrenamiento, cabe destacar que la cantidad de neuronas necesarias en esta única capa podría ser muy grande y la capacidad de generalización puede ser baja, dependiendo de la complejidad de los datos con los que se pretende trabajar. Redes más profundas generalmente permiten disminuir el número de unidades por capa (de esta forma tener una menor cantidad parámetros en total), como así también una mejor capacidad de generalización en el set de testeo, con esto se hace referencia a la capacidad de la red para clasificar correctamente ejemplos que no fueron usados para el proceso de entrenamiento, naturalmente hacer más intrincado el diseño de la red hará que el proceso de optimización para ajustar los parámetros sea más costoso. La arquitectura ideal se debe encontrar por experimentación con distintas estructuras acompañado de conocimiento sobre el tipo de datos, esto se apoya mediante el monitoreo del error en el set de validación.

Una de las principales justificaciones para usar redes neuronales se debe al **Teorema de Aproximación Universal** (Horniket al., 1989; Cybenko, 1989), el cual muestra que una red feedforward con una capa de output lineal y al menos una capa escondida con función de activación sigmoidal (y otras similares) puede aproximar cualquier función Borel medible [13] de un espacio dimensión finita a otro, con un nivel de error arbitrariamente pequeño, provisto de que hayan suficientes unidades escondidas. Una red neuronal también puede aproximar un mapping de cualquier espacio dimensional finito y discreto a otro, y este teorema también se ha probado para una amplia gama de funciones de activación, como para la más comúnmente usada ReLU (Leshno et al., 1993). La desventaja del teorema es que a pesar que asegura que la red podrá aproximar cualquiera de estas funciones, esto no implica que necesariamente podrá *aprender* la función en sí. Una razón posible es que el algoritmo de optimización podría no encontrar los parámetros de la red que alcancen el nivel de error deseado. Tampoco especifica la cantidad de unidades que son necesarias para alcanzar un nivel de error dado, lo cual podría ser excesivamente grande. Sin embargo en muchos casos modelos más profundos necesitarán menos unidades y potencialmente permiten reducir el error de generalización.

6.2. Entrenamiento de una Red Neuronal

6.2.1. Forward Propagation y Back-Propagation

Al usar una red neuronal feedforward, la información fluye a través de la red desde el ingreso de un input $\boldsymbol{x}$ hasta producir un output $\hat{\boldsymbol{y}}$. Esto se conoce como **forward propagation**. Durante el entrenamiento, *forward propagation* continúa hasta producir el costo escalar $J(\boldsymbol{\theta})$.

[13] En particular cualquier función continua en un subconjunto cerrado y acotado de $\mathbb{R}^n$ es Borel medible, la clase de funciones Borel medibles es mucho más rica y variada que las funciones continuas.

El algoritmo de **back-propagation** permite que la información del costo fluya en sentido inverso a través de la red para calcular el gradiente de manera computacionalmente eficiente. El gradiente se calculará de esta forma ya que, aunque es posible obtener una expresión analítica para este, evaluar la expresión puede ser muy caro computacionalmente. Luego de obtener el gradiente, otro algoritmo como descenso de gradiente estocástico (ver sección 4) realiza el aprendizaje usando la expresión que fue calculada. En algoritmos de aprendizaje el gradiente que más comúnmente se requiere obtener es $\nabla_{\boldsymbol{\theta}} J(\boldsymbol{\theta})$, aunque el algoritmo de *back-propagation* no se limita a esto y puede ser usando para otras tareas que involucren obtener derivadas.

Sea $\boldsymbol{x} \in \mathbb{R}^m, \boldsymbol{y} \in \mathbb{R}^n, g : \mathbb{R}^m \to \mathbb{R}^n, f : \mathbb{R}^n \to \mathbb{R}, \boldsymbol{y} = g(\boldsymbol{x})$ y $z = f(\boldsymbol{y})$, la regla de la cadena indica que:

$$\frac{\partial z}{\partial x_i} = \sum_j \frac{\partial z}{\partial y_j} \frac{\partial y_j}{\partial x_i} \tag{221}$$

O de manera equivalente visto en su forma vectorial:

$$\nabla_{\boldsymbol{x}} z = (\frac{\partial \boldsymbol{y}}{\partial \boldsymbol{x}})^T \nabla_{\boldsymbol{y}} z \tag{222}$$

en donde $\frac{\partial \boldsymbol{y}}{\partial \boldsymbol{x}}$ es la matriz Jacobiana de g, con esto se puede ver que para obtener el gradiente con respecto a la variable $\boldsymbol{x}$, basta multiplicar la matriz Jacobiana por el gradiente con respecto a $\boldsymbol{y}$.

Usualmente se aplicará *back-propagation* a **tensores** de dimensionalidad arbitraria, no solo vectores. Se denota entonces el gradiente de z con respecto a un tensor $\mathbf{X}$ como $\nabla_{\mathbf{X}} z$. Sean $\mathbf{Y} = g(\mathbf{X})$ y $z = f(\mathbf{Y})$, entonces la regla de la cadena se escribe como:

$$\nabla_{\mathbf{X}} z = \sum_j (\nabla_{\mathbf{X}} \mathrm{Y}_j) \frac{\partial z}{\partial \mathrm{Y}_j} \tag{223}$$

Antes de proseguir a revisar los algoritmos de *forward-propagation* y *back-propagation* se realizará una breve deducción de *back-propagation* que permitirá comprender de mejor manera el proceso de entrenamiento en una red neuronal, en primer lugar de analizará una red neuronal con solo una capa escondida, luego veremos que esto puede ser extendido de forma fácil a arquitecturas más complejas (osea, más capas).

Sea $\{(x_i, y_i)\}_{i=1}^N$ un conjunto de puntos sobre los que se quiere ajustar los pesos de una red neuronal, el conjunto de entrenamiento es tal que $x_i \in \mathbb{R}^p, y_i \in \mathbb{R}^K \quad \forall i = 1, ..., N$, el problema planteado corresponde a uno de clasificación sobre K clases distintas. Adicionalmente se tiene que el error (función de perdida) puede ser escrito como:

$$J(\theta) = \sum_{i=1}^N J_i = \sum_{i=1}^N L(y_i, f(x_i; \theta)) \tag{224}$$

Donde J_i corresponde a la discrepancia entre y_i y el valor entregado por la red $f(x_i; \theta)$ evaluado por la función de perdida L. Para fijar ideas, la arquitectura de la red está compuesta por p unidades de entrada, la capa escondida tiene M neuronas con funciones de activación sigmoides ($\sigma(x) = \frac{1}{1-e^{-t}}$) y la capa de salida (output) tendrá K unidades y una función de salida g. De esta forma el problema puede ser planteado como:

$$Z_m = \sigma(\alpha_{0m} + \alpha_m^T X), \quad \forall m = 1, ..., M \tag{225}$$

$$N_k = \beta_{0k} + \beta_k^T Z, \qquad \forall k = 1, ..., K \tag{226}$$

$$O_k = g_k(N), \qquad \forall k = 1, ..., K \tag{227}$$

$$f(X) = O \tag{228}$$

$$\tag{229}$$

Donde $Z = (Z_1, ..., Z_M)$, $N = (N_1, ..., N_K)$, $O = (O_1, ..., O_K)$ y se puede ver que $g : \mathbb{R}^K \to \mathbb{R}^K$, de esta forma los parámetros θ del modelo son:

$$\begin{aligned} \{\alpha_{0m}, \alpha_m; \quad m = 1, ..., M\} \quad M(p+1) \quad \text{pesos} \\ \{\beta_{0k}, \beta_k; \quad k = 1, ..., K\} \quad K(M+1) \quad \text{pesos} \end{aligned}$$

De ahora en adelante $J \equiv J(\theta)$, también es importante distinguir que es distinto calcular la derivada de J respecto a un peso que está en la capa de salida y un peso que pertenece a una capa escondida, siendo el primer caso mucho más fácil de calcular, esto se justifica por el hecho que los valores de la capa de salida participan directamente en el calculo del error J, mientras que los valores de las neuronas escondidas no, esto se hará evidente en los cálculos. Como último preámbulo se definen las siguientes variables que nos ayudaran a hacer los cálculos más claros:

$$\begin{aligned} z_{mi} &= \sigma(\alpha_{0m} + \alpha_m^T x_i) \\ z_i &= (z_{1i}, ..., z_{Mi}) \\ n_{ki} &= \beta_{0k} + \beta_k^T z_i \\ n_i &= (n_{1i}, ..., n_{Ki}) \\ o_i &= g(n_i) \\ o_{ki} &= (o_i)_k \end{aligned}$$

Considerando que derivar es una operación lineal calcularemos la derivada de J_i, a partir de este termino podemos obtener la derivada de J sumando sobre todo $i = 1, ..., N$.

$$\frac{\partial J_i}{\partial \beta_{km}} = \frac{\partial J_i}{\partial o_{ki}} \frac{\partial o_{ki}}{\partial n_{ki}} \frac{\partial n_{ki}}{\partial \beta_{km}} \tag{230}$$

$$= \frac{\partial J_i}{\partial o_{ki}} g_k'(n_{ki}) z_{mi} \tag{231}$$

Dado que o_{ki} pertenece a la capa de salida, esto implica que participa directamente en la expresión de J_i y por lo tanto la derivada $\frac{\partial J_i}{\partial o_{ki}}$ puede ser calculada directamente. El calculo recién hecho se conoce como regla delta ('delta rule') y es el paso principal de *back-propagation*, este consiste en usar la regla de la cadena 2 veces sucesivamente.

Como se mencionó con anterioridad calcular la derivada de J_i respecto a un peso perteneciente a una capa escondida es más engorroso, ya que las neuronas en las capas intermedias no participan directamente en el error, aún así es posible y el calculo entrega una formula recursiva que permite llegar a una expresión cerrada y explicita.

$$\frac{\partial J_i}{\partial \alpha_{ml}} = \frac{\partial J_i}{\partial z_{mi}} \frac{\partial z_{mi}}{\partial \alpha_{ml}} \tag{232}$$

$$= \left[\sum_{k=1}^{K} \frac{\partial J_i}{\partial o_{ki}} \frac{\partial o_{ki}}{\partial n_{ki}} \frac{\partial n_{ki}}{\partial z_{mi}} \right] \frac{\partial z_{mi}}{\partial \alpha_{ml}} \tag{233}$$

$$= \left[\sum_{k=1}^{K} \frac{\partial J_i}{\partial o_{ki}} \frac{\partial o_{ki}}{\partial n_{ki}} \frac{\partial n_{ki}}{\partial z_{mi}} \right] \sigma'(\alpha_0 + \alpha_m x_i) x_{il} \tag{234}$$

$$= \left[\sum_{k=1}^{K} \frac{\partial J_i}{\partial o_{ki}} g'(n_{ki}) \beta_{km} \right] \sigma'(\alpha_0 + \alpha_m x_i) x_{il} \tag{235}$$

$$= \sum_{k=1}^{K} \frac{\partial J_i}{\partial o_{ki}} g'(n_{ki}) \beta_{km} \sigma'(\alpha_0 + \alpha_m x_i) x_{il} \tag{236}$$

Al llegar a la expresión final se puede ver que todos los términos pueden ser calculados de forma explicita, en este punto se puede apreciar porqué el nombre *back-propagation*, a partir de los cálculos obtenidos en las capas exteriores se pueden obtener derivadas de los pesos en capas anteriores, en este sentido la información se propaga 'hacia atrás' en la red. Finalmente reescribimos (231) y (236) como:

$$\frac{\partial J_i}{\partial \beta_{km}} = \delta_{ki} z_{mi} \qquad \frac{\partial J_i}{\partial \alpha_{ml}} = s_{mi} x_{il} \tag{237}$$

Donde:

$$\delta_{ki} = \frac{\partial J_i}{\partial o_{ki}} g'_k(n_{ki}) \tag{238}$$

$$s_{mi} = \sigma'(\alpha_0 + \alpha_m x_i) \sum_{k=1}^{K} \beta_{km} \delta_{ki} \tag{239}$$

Las cantidades δ_{ki} y s_{mi} son errores del modelo en la capa de salida y capas escondidas respectivamente, las 4 ultimas ecuaciones presentadas son conocidas como como ecuaciones de *back-propagation*. Para extender el desarrollo a una red neuronal con más capas es importante notar que a partir de la capa final y la anterior se logró calcular todas las derivadas del gradiente del error, si se considerará la capa de entrada (input) como otra capa escondida más, se pueden repetir los mismos razonamientos y extenderlos a un nivel de profundidad arbitrario.

A continuación se presentan los algoritmos de *forward propagation* y *back-propagation* para una red MLP en donde se conectan todas las unidades de una capa con la siguiente (fully connected MLP).

Algoritmo 1 Forward Propagation

Requerir: Profundidad de la red, l
Requerir: $\boldsymbol{W}^{(i)}, i \in \{1, ..., l\}$, pesos de la red
Requerir: $\boldsymbol{b}^{(i)}, i \in \{1, ..., l\}$, parámetros bias de la red
Requerir: $\boldsymbol{x}$, el input
Requerir: $\boldsymbol{y}$, el output target
$\boldsymbol{h}^{(0)} = \boldsymbol{x}$
for $k = 1, ..., l$:
$\boldsymbol{a}^{(k)} = \boldsymbol{b}^{(k)} + \boldsymbol{W}^{(k)} \boldsymbol{h}^{(k-1)}$
$\boldsymbol{h}^{(k)} = f(\boldsymbol{a}^{(k)})$
$\hat{\boldsymbol{y}} = \boldsymbol{h}^{(l)}$
$J = L(\hat{\boldsymbol{y}}, \boldsymbol{y})$

Algoritmo 2 Back-Propagation

Luego de completar forward propagation:
$\boldsymbol{g} \leftarrow \nabla_{\hat{\boldsymbol{y}}} J = \nabla_{\hat{\boldsymbol{y}}} L(\hat{\boldsymbol{y}}, \boldsymbol{y})$
for $k = l, l-1, ..., 1$:
$\nabla_{\boldsymbol{a}^{(k)}} J = \boldsymbol{g} \odot f'(\boldsymbol{a}^{(k)})$
$\nabla_{\boldsymbol{b}^{(k)}} J = \boldsymbol{g}$
$\nabla_{\boldsymbol{W}^{(k)}} J = \boldsymbol{g}\boldsymbol{h}^{(k-1)T}$
$\boldsymbol{g} \leftarrow \nabla_{\boldsymbol{h}^{(k-1)}} J = \boldsymbol{W}^{(k)T}\boldsymbol{g}$

Con este algoritmo se obtienen los gradientes con respecto a todos los parámetros hasta la primera capa, propagando el gradiente desde una capa a la anterior mediante la última actualización del algoritmo para cada iteración.

6.3. Regularización para una Red Neuronal

Las redes neuronales y algoritmos de deep learning son aplicados a tareas extremadamente complejas como lo son el procesamiento de imágenes, audio, y texto. Controlar la complejidad de un modelo no solo se reduce a encontrar el tamaño y cantidad de parámetros adecuados, como se ha visto para otros modelos de aprendizaje de máquinas, sino que en la práctica el modelo con el mejor ajuste por lo general será un modelo grande (profundo) que ha sido regularizado apropiadamente.

6.3.1. Regularización L^2

Una regularización que se basa en limitar la norma de los parámetros del modelo es la ya conocida **regularización $\boldsymbol{L}^2$** (o **ridge regression**), mediante la cual se obtiene la función objetivo regularizada $\tilde{J}$:

$$\tilde{J}(\boldsymbol{\theta}; \boldsymbol{X}, \boldsymbol{y}) = J(\boldsymbol{\theta}; \boldsymbol{X}, \boldsymbol{y}) + \frac{\alpha}{2}||\boldsymbol{\theta}||_2^2 \tag{240}$$

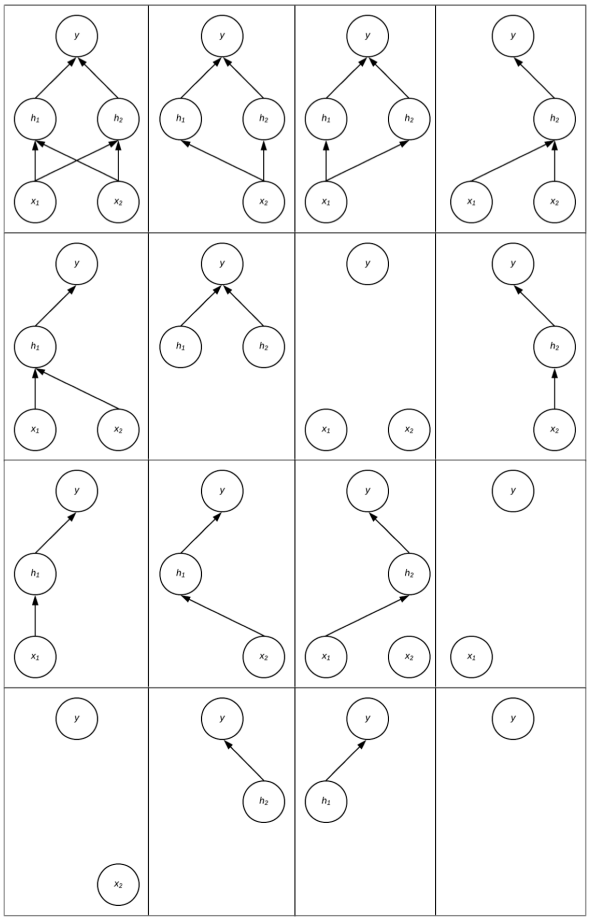
en donde el hiperparámetro $\alpha \in [0, \infty[$ indica que tanta importancia se le da al termino de regularización sobre el objetivo, no habrá regularización cuando $\alpha = 0$ y se observará un mayor efecto regularizador a medida que α crece. Cabe destacar que típicamente en una regularización por la norma solo se regularizan los *pesos*, dejando los términos de *bias* sin regularizar. Esto ya que cada término de *bias* controla el comportamiento de solo 1 variable implicando que no se introduce mucha varianza **[(*overfitting*), no se si es pertinente esta acotación]*** al dejarlos sin regularizar, por otro lado regularizar los *bias* puede inducir un alto nivel de *underfitting*.

6.3.2. Dropout

Bagging consiste en entrenar múltiples modelos y evaluarlos en cada dato del set de testeo. Esto no es práctico para redes neuronales**[Duda, según lo que vi baggin se refiera a cuando se entrenan muchos clasificadores y luego votan sobre el resultado]***, ya que un solo modelo puede ser muy caro de entrenar y evaluar. **Dropout** provee una aproximación barata (computacionalmente) para entrenar y evaluar *bagged* ensambles compuestos por una cantidad exponencial de redes neuronales. Dropout entrena los ensambles de posiblemente todas las subredes que se puedan formar al remover unidades (que no sean las de output) de un modelo de red neuronal (ver figura). Esto se puede realizar al multiplicar por 0 el output de alguna unidad para la mayoría de los casos. Específicamente, para entrenar con dropout se usa un algoritmo por mini-batches (ver sección 4) para que en cada iteración se entrene una subred aplicando una máscara binaria a todas las capas de input y escondidas. Los hiperparámetros de este método de regularización corresponden al diseño de la máscara binaria, especificando la probabilidad de que se incluya una unidad de input y la probabilidad de que se incluya una unidad escondida. Típicamente

se usan los valores 0.8 y 0.5, respectivamente, sin embargo estos deben ser ajustados para controlar el nivel de regularización (mayor valor para las probabilidades implica menor efecto regularizador) para obtener un desempeño óptimo en el set de validación.

Fig. 19. Dropout entrena potencialemente todas las subredes que se puedan formar a partir de la red neuronal original (primer recuadro) al apagar el output que producen las distintas unidades



6.3.3. Otros Métodos de Regularización

Otras formas de regularización también buscan introducir alguna fuente de ruido (como en dropout) para que la red neuronal aprenda principalmente los parámetros más importantes, así logrando un bajo error de generalización. Una de estas técnicas es **dataset augmentation**, que consiste en generar nuevos datos de entrenamiento (inyectando ruido en el set de entrenamiento), creando datos $\boldsymbol{x}$ falsos para los cuales se pueda tener una etiqueta y (por ejemplo, una imagen invertida de un gato sigue siendo un gato), o **entrenamiento adversarial**, en donde se perturban ejemplos para fortalecer a la red (por ejemplo, cambiar pixeles de una imagen que generen cambios imperceptibles para un humano pero que pueden afectar fuertemente la capacidad de predicción de un modelo). Otra técnica es **noise injection** en los pesos (Jim et al., 1996; Graves, 2011), lo cual se puede interpretar como una implementación estocástica de inferencia Bayesiana sobre los pesos, debido a que el aprendizaje consideraría que los pesos son inciertos y, por lo tanto, representables mediante una distribución de probabilidad.

También, por supuesto, **early stopping** es una técnica válida para regularizar redes neuronales.

6.4. Algoritmos de Optimización

Como ya se ha comentado, la optimización en redes neuronales busca resolver un problema particular: encontrar los parámetros $\boldsymbol{\theta}$ que disminuyan significativamente $J(\boldsymbol{\theta})$, que depende de alguna medida de desempeño evaluada en la totalidad del set de entrenamiento, luego se evaluá el error en el set de validación para tener una idea del desempeño, finalmente se ven los resultados en el set de testeo.

Esto se reduce a minimizar la esperanza del error sobre la distribución generadora de los datos, p_{data}:

$$J^*(\boldsymbol{\theta}) = \mathbb{E}_{(\boldsymbol{x},y)\sim p_{\text{data}}} L(f(\boldsymbol{x};\boldsymbol{\theta}), y) \tag{241}$$

Reemplazamos la expresión anterior por un problema sustituto, que consiste en escribir la función de costos como un promedio sobre el set de entrenamiento, como se puede observar la diferencia entre las dos expresiones radica en el hecho que se considera que los datos fueron generados por distribuciones de probabilidad distintas (p_{data} en la primera expresión, $\hat{p}_{\text{data}}$ en la segunda):

$$J(\boldsymbol{\theta}) = \mathbb{E}_{(\boldsymbol{x},y)\sim \hat{p}_{\text{data}}} L(f(\boldsymbol{x};\boldsymbol{\theta}), y) = \frac{1}{m}\sum_{i=1}^{m} L(f(x^{(i)};\theta), y^{(i)}) \tag{242}$$

6.4.1. Descenso del Gradiente Estocástico y por Batches

Los algoritmos de optimización para aprendizaje de máquinas típicamente actualizan los parámetros usando un valor esperado del costo, obtenido a través de un subset de los términos de la función de costos. La propiedad más usada respecto de la función objetivo (1) es sobre el gradiente, este cumple la siguiente expresión:

$$\nabla_{\boldsymbol{\theta}} J(\boldsymbol{\theta}) = -\,\mathbb{E}_{\boldsymbol{x},y\sim\hat{p}_{\text{data}}} \nabla_{\boldsymbol{\theta}} \log p_{\text{modelo}}(y|\boldsymbol{x}) \tag{243}$$

Calcular esta expresión es computacionalmente caro, ya que requiere evaluar cada ejemplo del set de entrenamiento, por lo que se puede optar por samplear un pequeño número de ejemplos para obtener este valor esperado, calculando el promedio usando solo estos ejemplos. Los algoritmos de optimización que usan el set de entrenamiento completo para actualizar los parámetros en cada iteración se conocen como **métodos de batch** o **determinísticos**. Los algoritmos que usan un solo ejemplo a la vez se conocen como **métodos estocásticos** u **online** (aunque el término *online* se suele usar para describir un entrenamiento con un flujo continuo de nuevos ejemplos). La mayoría de los algoritmos usados pertenecen a una categoría intermedia, estos son los **métodos de minibatch** o **minibatch estocástico**, los cuales usan un subconjunto de tamaño reducido de la totalidad de los ejemplos. Un criterio guía para decidir el número de batches a usar, es que batches más grandes proveen estimadores más precisos del gradiente, en este caso se obtienen retornos menores a uno lineal.

Una motivación importante para usar descenso de gradiente por mini-batches es que sigue el gradiente del costo que considera el error de generalización (241) mientras no se repitan ejemplos. En la práctica las implementaciones de descenso del gradiente por mini-batches desordenan el set de datos una vez y luego pasan por él múltiples veces.

6.4.2. Algoritmos con Momentum

Aunque los métodos de descenso de gradiente estocástico sigue siendo un algoritmo popular, el aprendizaje a veces puede ser lento. Los algoritmos que incorporan momentum fueron diseñados para acelerar el aprendizaje, especialmente en presencia de altas curvaturas, cuando se tienen gradientes pequeños pero consistentes o gradientes ruidosos. El algoritmo **descenso de gradiente estocástico con momentum** acumula un decaimiento exponencial de media móvil de los gradientes pasados y continúa su movimiento en esta dirección. Un hiperparámetro α determina qué tan rápido las contribuciones de gradientes pasados decaen exponencialmente. Se actualiza mediante:

$$\boldsymbol{v} \longleftarrow \alpha \boldsymbol{v} - \epsilon \nabla_{\boldsymbol{\theta}} \Big(\frac{1}{m} \sum_{i=1}^{m} L(f(\boldsymbol{x}^{(i)}; \boldsymbol{\theta}), y^{(i)}) \Big)$$
$$\theta \longleftarrow \theta + \boldsymbol{v}$$

Con ϵ el learning rate y $\boldsymbol{v}$ la velocidad o momentum.

6.4.3. Algoritmos con Learning Rates Adaptativos

En la práctica el learning rate resulta ser uno de los hiperparámetros más difíciles de ajustar debido a su importante efecto en el desempeño del modelo. La función de costos suele ser altamente sensible (a crecer o decrecer) en algunas direcciones en el espacio de los parámetros e insensible en otras, por lo que hace sentido usar un learning rate distinto para cada parámetro y automáticamente adaptar este parámetro durante el aprendizaje. El algoritmo **AdaGrad** adapta el learning rate de todos los parámetros al escalarlos de manera inversamente proporcional a la raíz cuadrada de la suma de todos las raíces cuadradas históricas del gradiente. Los parámetros con derivadas parciales más grandes tienen un rápido decrecimiento en su learning rate, mientras que los parámetros con derivadas parciales pequeñas decrecen en menor cantidad su learning rate. El efecto neto es mayor progreso en zonas más planas del espacio de los parámetros. Se actualiza mediante:

$$\delta = 10^{-7}; \boldsymbol{r} = 0$$
$$\boldsymbol{g} \longleftarrow \frac{1}{m} \nabla_{\boldsymbol{\theta}} \sum_{i=1}^{m} L(f(\boldsymbol{x}^{(i)}; \boldsymbol{\theta}), y^{(i)})$$
$$\boldsymbol{r} \longleftarrow \boldsymbol{r} + \boldsymbol{g} \odot \boldsymbol{g}$$
$$\triangle \boldsymbol{\theta} \longleftarrow -\frac{\epsilon}{\delta + \sqrt{\boldsymbol{r}}} \odot \boldsymbol{g}$$
$$\boldsymbol{\theta} \longleftarrow \boldsymbol{\theta} + \triangle \boldsymbol{\theta}$$

Con δ una constante pequeña para estabilidad numérica (puede ser otra), $\boldsymbol{r}$ la variable de acumulación del gradiente, $\boldsymbol{g}$ el gradiente para el batch, y $\triangle \boldsymbol{\theta}$ la actualización de los parámetros al final de una iteración.

Otras generalizaciones populares son el algoritmo **RMSProp**, que modifica el algoritmo AdaGrad para tener un mejor desempeño en funciones no convexas al cambiar la acumulación del gradiente por una media móvil que decae exponencialmente, y el algoritmo **Adam**, que combina RMSProp con momentum (con algunas distinciones importantes).

Hasta el momento no hay un algoritmo que tenga un desempeño superior al de los demás en distintos escenarios (Schaul et al., 2014), por lo que se recomienda usar el algoritmo de optimización con el que el usuario se sienta más cómodo al momento de ajustar los hiperparámetros.

6.5. Deep Learning y Otros Tipos de Redes Neuronales

El término **deep learning** se asocia a resolver problemas más intuitivos (y fáciles) para los humanos que hasta hace solo algunos años eran extremadamente difíciles para una máquina, como lo son el reconocimiento de objetos en imágenes (visión de computadores), la traducción de texto desde un lenguaje a otro (machine translation), reconocimiento de voz, entre otros. Los problemas más difíciles para los humanos ya se han estado resolviendo hace mucho tiempo antes del deep learning, como divisar una estrategia ganadora en ajedrez (`https://en.wikipedia.org/wiki/Deep_Blue_versus_Garry_Kasparov`), aunque las redes neuronales profundas han seguido progresando en resolver este tipo de problemas (`https://deepmind.com/research/alphago/`). Las arquitecturas principales que han permitido resolver

estos problemas en los últimos años (sumado a los avances en poder de computación y cantidad de datos que existen hoy) se presentan en esta sección: las **redes convolucionales** para procesamiento de imágenes, y las **redes recurrentes** para modelar series de tiempo (e.g., texto, audio). También se presentan otras arquitecturas que son tema activo de investigación en deep learning.: los **autoencoders** y las **redes generativas adversariales**.

6.5.1. Redes Neuronales Convolucionales

Las **redes neuronales convolucionales** (o **CNNs**) son un tipo de redes neuronales que fueron diseñadas para procesar datos con una tipología tipo-*grid* (grilla). Una serie de tiempo que tiene observaciones en intervalos regulares de tiempo se puede pensar como un *grid* de 1 dimensión. Las imágenes se pueden pensar como *grids* de 2-D de pixeles. El nombre de esta arquitectura hace referencia a que usan una operación matemática conocida como convolución.

La operación de **convolución** se define como:

$$s(t) = (x * w)(t) = \int x(a)w(t-a)da \tag{244}$$

En el contexto de CNNs, el primer argumento a convolucionar, x, es el **input**, y el segundo argumento, w, se conoce como el **kernel**. El output, $s(t)$ se conoce como **feature map**. En aplicaciones de aprendizaje de máquinas, el input serán un arreglo multidimensional de datos, y el kernel un arreglo multidimensional de parámetros que se buscarán aprender. Usualmente, al trabajar con datos en un computador el tiempo se considerará discreto, por lo que resulta conveniente definir la operación de convolución discreta:

$$s(t) = (x * w)(t) = \sum_{a=-\infty}^{\infty} x(a)w(t-a) \tag{245}$$

Se asumirá que las funciones son 0 en todo su dominio excepto en el set finito de puntos para el cual se guardan valores, permitiendo realizar estas sumatorias infinitas. Las librerías de redes neuronales implementan la función **cross-correlation** y la llaman convolución. Para una imagen I de 2 dimensiones y un kernel K de 2 dimensiones, esto es:

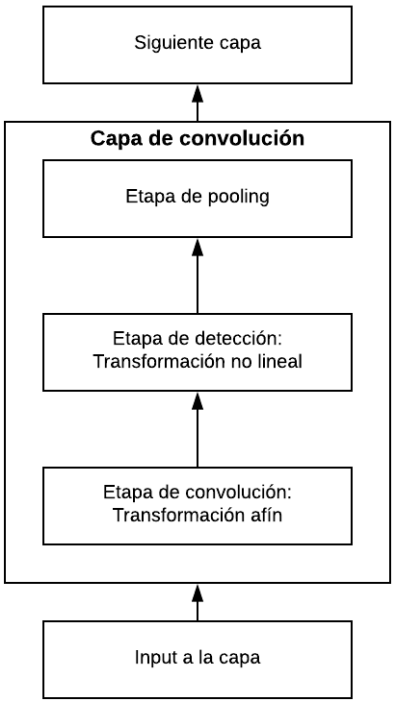
$$S(i,j) = (I * K)(i,j) = \sum_m \sum_n I(i+m, j+n)K(m,n) \tag{246}$$

Esto es equivalente a la operación de convolución discreta con la diferencia que el operador no es conmutativo, $S(i,j) = (I * K)(i,j) \neq (K * I)(i,j)$, lo cual no es importante para implementaciones de redes neuronales.

La motivación por usar convoluciones surge de 3 importantes propiedades: interacciones *sparse*, *parameter sharing*, y representaciones equivariantes. **Interacciones sparse** se refiere a que hay menos conexiones que en una red *fully connected*, esto mediante el uso de kernels de menor dimensionalidad que el input, lo cual implica una eficiencia en términos de memoria por tener que almacenar menos parámetros, y eficiencia computacional por realizar menos operaciones. **Parameter sharing** se refiere a usar los mismo parámetros para distintas funciones dentro del modelo. Esto implica usar los pesos aprendidos en múltiples partes del input (como para reconocer bordes, por ejemplo). Que una función sea **equivariante** significa que si el input cambia, el output cambia de la misma forma. Esto permite que en imágenes la convolución cree un map 2-D dónde ciertos atributos aparecen en el input.

Una capa típica de una red convolucional consta de 3 etapas: primero, aplicar varias convoluciones para producir un set de activaciones lineales. Luego, aplicar una activación no lineal (**detector stage**). Finalmente, una función de *pooling* para modificar aún más el output de la capa (ver figura). Una función de **pooling** reemplaza el output de la red en alguna locación por estadísticos de los outputs cercanos.

Fig. 20. Capa de una red convolucional



Max pooling retorna el valor máximo de un output en una vecindad rectangular. Las operaciones de pooling permiten que la red sea invariante a pequeñas transformaciones en el input. Pooling también es escencial para procesar inputs de tamaño variable (por ejemplo imágenes de distinto tamaño).

Otras diferencias con respecto a la operación de convolución en el contexto de redes neuronales son, por ejemplo, el aplicar múltiples convoluciones en paralelo, esto permite extraer distintos tipos de atributos en vez de 1 solo. Por otro lado, el **stride** hace referencia a cada cuántos pixeles se quieren convolucionar en cada dirección en el output. En la figura se muestra el ejemplo de una convolución con stride. Esta operación permite reducir nuevamente el costo computacional. Esto también implica que el output disminuye su tamaño en cada capa. El uso de *padding* puede revertir esto. **Padding** se refiere a agrandar el input con ceros para hacerlo más amplio. Una convolución en la que no se usa **zero-padding** se conoce como **valid**. Una convolución que mantiene el tamaño desde el input al output se conoce como **same** (ver figura). En la práctica, las capas de una red convolucional usan operaciones entre una convolución valid y same.

Fig. 21. Convolución con un stride igual a 2

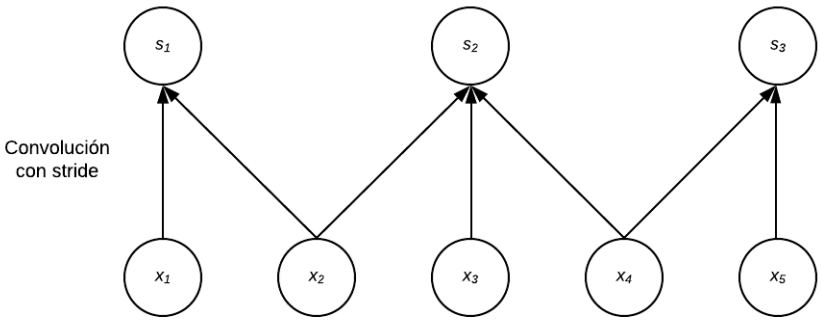
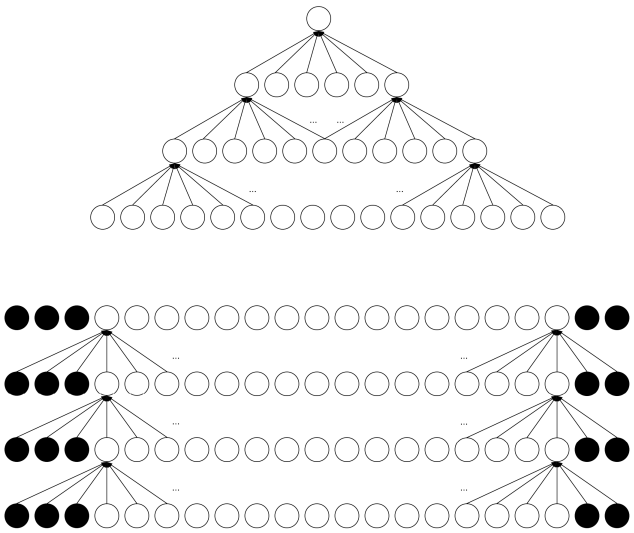


Fig. 22. Efecto de no usar zero-padding en una red convolucional (Arriba) y efecto de usar zero padding en una red convolucional (Abajo) en cuanto al tamaño de la red



6.5.2. Redes Neuronales Recurrentes

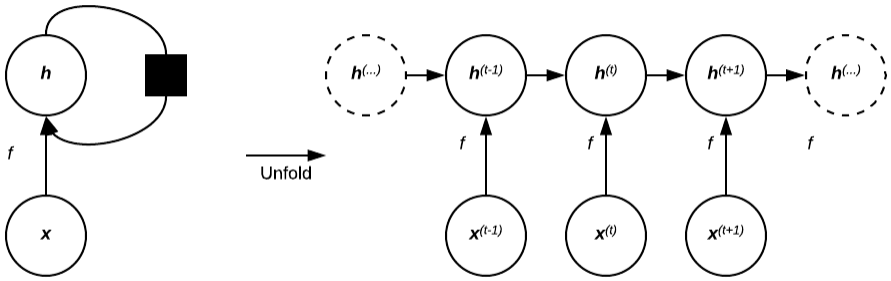
Las **redes neuronales recurrentes** o **RNNs** son una familia modelos de redes neuronales especializados para procesar datos secuenciales, $\boldsymbol{x}^{(1)}, ..., \boldsymbol{x}^{(\tau)}$. Las RNNs también comparten parámetros, pero en una forma muy distinta que las CNNs. En una RNN, cada miembro del output en una etapa es una función de cada miembro del output de la etapa anterior.

Se denota por $\boldsymbol{h}^{(t)}$ al estado de un sistema dinámico que involucra una recurrencia conducido por un input externo $\boldsymbol{x}^{(t)}$:

$$\boldsymbol{h}^{(t)} = f(\boldsymbol{h}^{(t-1)}; \boldsymbol{x}^{(t)}, \boldsymbol{\theta}) \tag{247}$$

La figura siguiente muestra una red recurrente que procesa un input $\boldsymbol{x}$ incorporándolo al estado $\boldsymbol{h}$ que es traspasado a través del tiempo.

Fig. 23. Ejemplo de una red recurrente sin output



Las redes recurrentes se pueden construir de muchas formas distintas. Al igual que una red neuronal puede representar casi cualquier función, una red recurrente modela cualquier función que involucre una

recurrencia. Se puede representar el estado de una red recurrente luego de t pasos mediante una función $g^{(t)}$:

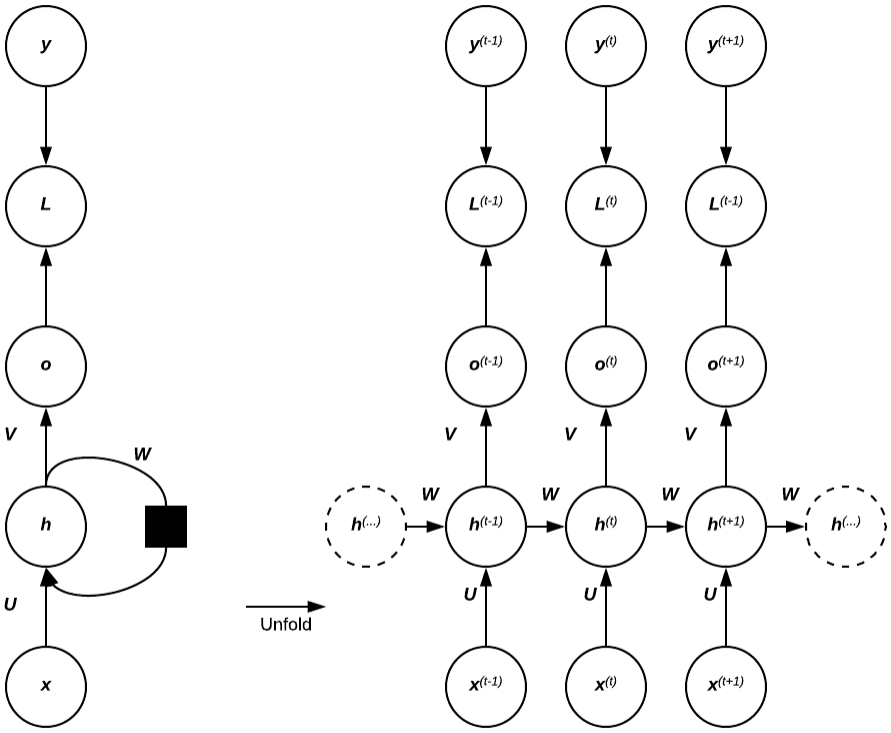
$$\boldsymbol{h}^{(t)} = g^{(t)}(\boldsymbol{x}^{(t)}, \boldsymbol{x}^{(t-1)}, ..., \boldsymbol{x}^{(1)}) = f(\boldsymbol{h}^{(t-1)}; \boldsymbol{x}^{(t)}, \boldsymbol{\theta}) \tag{248}$$

Existen varios tipos de RNNs que se han diseñado para distintos fines. Algunos ejemplos de estas son:

- Redes recurrentes que producen un output en cada instante de tiempo y tienen conexiones entre todas las unidades escondidas
- Redes recurrentes que producen un output en cada instante de tiempo y tienen conexiones entre el output a la unidad escondida del siguiente instante
- Redes recurrentes con conexiones entre las unidad escondidas, que procesan una secuencia entera antes de producir el output

La figura muestra un ejemplo de la arquitectura para el primer caso.

Fig. 24. Ejemplo de una red recurrente que produce un output en cada instante de tiempo, y que comparte el estado a través del tiempo



Esta red presentada puede ser usada para producir palabras en cada instante, y así producir oraciones que hagan sentido en una conversación. Como ejemplo de entrenamiento de una red con esta arquitectura, en donde en la última capa se decide mediante una función softmax la palabra más probable que deba seguir a la palabra anterior, se tienen las ecuaciones de *forward propagation* para cada instante de tiempo:

$$
\begin{aligned}
\boldsymbol{a}^{(t)} &= \boldsymbol{b} + \boldsymbol{W}\boldsymbol{h}^{(t-1)} + \boldsymbol{U}\boldsymbol{x}^{(t-1)} \\
\boldsymbol{h}^{(t)} &= \tanh(\boldsymbol{a}^{(t)}) \\
\boldsymbol{o}^{(t)} &= \boldsymbol{c} + \boldsymbol{V}\boldsymbol{h}^{(t)} \\
\hat{\boldsymbol{y}}^{(t)} &= \text{softmax}(\boldsymbol{o}^{(t)})
\end{aligned}
\tag{249}
$$

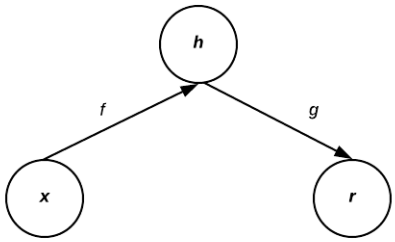
El algoritmo aplicado para obtener el gradiente en este tipo de arquitectura se conoce como **back-propagation through time**, y consiste en aplicar el algoritmo de *back-propagation* generalizado para el grafo computacional *unfolded* de la red, como los mostrados en las figuras de redes recurrentes.

Las redes recurrentes sufren de no poder recordar largas dependencias a través del tiempo, debido a que las recurrencias implican multiplicar una matriz de pesos múltiples veces a través de la red, provocando superficies planas o muy empinadas que resultan en que los algoritmos de aprendizaje por gradiente tengan problemas de **vanishing gradients** o **exploding gradients**, respectivamente. Arquitecturas que han logrado superar esto son las que incluyen compuertas (funciones sigmoidales) que deciden automáticamente qué olvidar y qué seguir propagando a través de la red. Estos son los modelos de **gated recurrent units** (**GRUs**) y **long-short term memory network** (**LSTM**).

6.5.3. Autoencoders

Un **autoencoder** es una red neuronal que busca replicar el input hacia el output, osea, se busca que la información que entra a la red sea lo más parecida posible a la de salida, para lo cual cuenta con una capa interna $\boldsymbol{h} = f(\boldsymbol{x})$ que codifica el input (genera una representación de este) llamada **encoder** y una función que produce la reconstrucción $\boldsymbol{r} = g(\boldsymbol{h})$, el **decoder**. Un *autoencoder* buscará aprender los *encoder* y *decoder* tales que $g(f(\boldsymbol{x})) = \boldsymbol{x}$ para todo $\boldsymbol{x}$. Como el modelo está forzado a aprender los atributos más importantes para que pueda efectivamente reproducir el input en su output, este aprenderá en general propiedades útiles de los datos de entrenamiento. Los *autoencoders* modernos modelan mappings estocásticos $p_{\text{encoder}}(\boldsymbol{h}|\boldsymbol{x})$ y $p_{\text{decoder}}(\boldsymbol{x}|\boldsymbol{h})$, en vez de funciones determinísticas. En la figura se muestra la arquitectura de un *autoencoder*.

Fig. 25. Estructura de un *autoencoder* típico



Una manera de obtener atributos útiles de $\boldsymbol{x}$ es forzando a que el *encoder* tenga una dimensionalidad menor que el input. Este tipo de *autoencoders* se denominan **undercomplete**. El aprendizaje se describe mediante la optimización de la función de pérdida, $L(\boldsymbol{x}, g(f(\boldsymbol{x})))$. Cuando el *decoder* es lineal y la función de pérdida es el error cuadrático medio, un *undercomplete autoencoder* aprende a generar el mismo subespacio que el algoritmo **principal component analysis** (**PCA**), es decir, el *autoencoder* que fue entrenado para reproducir los datos de entrenamiento mediante una reducción de dimensionalidad y una reconstrucción aprendió como efecto colateral el subespacio principal. Es así como entonces, *autoencoders* con *encoders* y *decoders* no lineales pueden aprender representaciones no lineales más poderosas que PCA.

Un *autoencoder* con dimensión de su *encoder* igual a la del input se conoce como **overcomplete**. Estos *autoencoders*, al igual que los *undercomplete*, pueden fallar en aprender una representación útil del

input si tienen mucha capacidad, por lo que será importante también regularizar estas redes neuronales.

Otras aplicaciones de los autoencoders, aparte de aprender una reducción de dimensionalidad, es aprender representaciones útiles que sirvan para un posterior modelo de redes neuronales (o, más general, de aprendizaje de máquinas). Por ejemplo, en vez de usar **one-hot-vectors** para representar palabras (en donde se tiene un vector del largo de cierto vocabulario compuesto por ceros excepto para la palabra que se quiere representar, indicando un valor de 1 en esa posición), se pueden usar **embeddings**, que son representaciones del input a un espacio de valores reales. A diferencia de los *one-hot-vectors*, en un *embedding* la distancia entre las representaciones del texto sí tiene un significado, y este tipo de representaciones podría entregar mejores resultados en la tarea en que se esté usando.

6.5.4. Redes Generativas Adversariales

Una **red generativa adversarial** (o **GAN**) se basa en un escenario de teoría de juegos, en donde una **red generadora** debe competir con un adversario. La red generadora produce muestras $\boldsymbol{x} = g(\boldsymbol{z}; \boldsymbol{\theta}^{(g)})$, mientras que una **red discriminadora** trata de distinguir entre muestras obtenidas de los datos de entrenamiento y muestras generadas por la red generadora. El discriminador retorna una probabilidad, $d(\boldsymbol{x}; \boldsymbol{\theta}^{(d)})$, indicando la probabilidad de que $\boldsymbol{x}$ sea un dato real y no uno simulado.

Para formuuar el aprendizaje, se describe un juego de suma cero en donde una función $v(\boldsymbol{\theta}^{(g)}, \boldsymbol{\theta}^{(d)})$ determina el pago del discriminador, y el generador recibe $-v(\boldsymbol{\theta}^{(g)}, \boldsymbol{\theta}^{(d)})$ como pago. Así, durante el entrenamiento cada jugador intenta maximizar su propio pago, para que en convergencia se tenga

$$g^* = \text{argmin}_g \text{max}_d v(g, d) \tag{250}$$

Esto motiva a que el discriminador aprenda a clasificar correctamente entre muestras reales y falsas y, simulatáneamente, el generador intenta engañar al clasificador para que crea que las muestras generadas son reales. En convergencia, las muestras del generador son indistinguibles de los datos reales. Una motivación del uso de GANs es que cuando $\text{max}_d v(g, d)$ es convexa en $\boldsymbol{\theta}^{(g)}$, el procedimiento asegura la convergencia.

Fig. 26. Imágenes generadas por una GAN entrenada con el set de datos LSUN. (Izquierda) Imágenes de dormitorios generadas por el modelo DCGAN (imagen de Radford et al., 2015). (Derecha) Imágenes de iglesias generadas por el modelo LAPGAN (imagen de Denton et al., 2015)

7. Support Vector Machines

7.1. Introducción

En este capítulo del apunte, discutiremos las llamadas **máquinas de soporte vectorial** (SVM) que constituyen un set de herramientas para resolver el problema de clasificación de datos. Fue desarrollada en los 90 dentro de la comunidad de computer science y ha crecido enormemente desde entonces. Esta técnica ha demostrado ser útil en diversos escenarios, y es considerado uno de los mejores clasificadores para "llegar y usar".

7.2. Idea general

Los SVM son una generalización de un clasificador más simple e intuitivo llamado el clasificador de márgen máximo (*maximum margin classifier*)

Para estudiar dicho clasficador, comenzaremos estudiando el caso en que tenemos que clasificar los datos en dos grupos, y además supondremos que los datos son linealmente separables (es decir, existe un hiperplano que los separa, ver fig. 27). Esto último es claramente un supuesto muy fuerte, pero lo generalizaremos mas adelante.

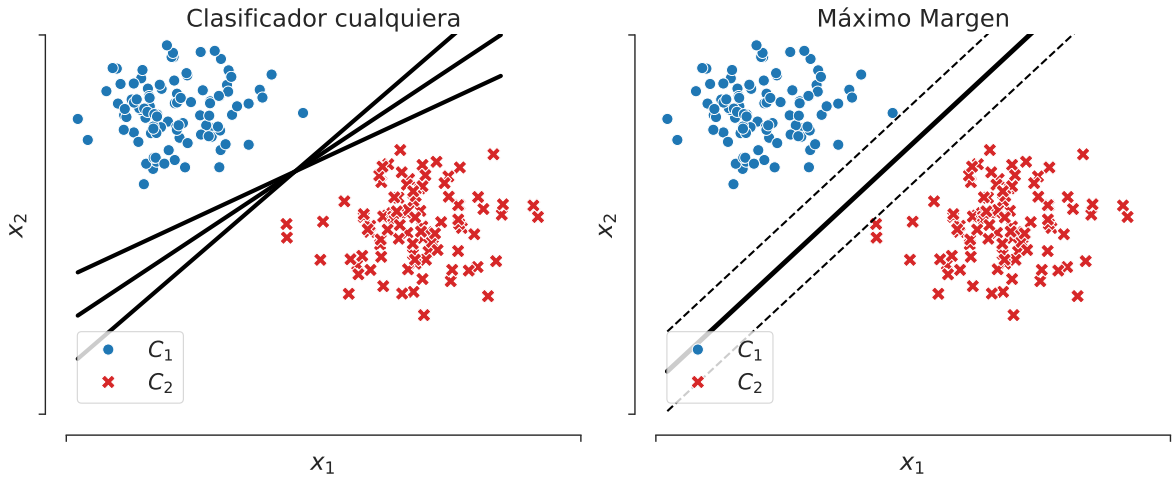


Fig. 27. Modelo de máximo margen entre los datos

Como se puede apreciar en la fig. 27 (centro), dados puntos linealmente separables, existen diferentes hiperplanos que separan los datos. Cada una de estas líneas define un modelo, ya que dado un nuevo dato x^*, si este está por encima de la línea, entonces es un dato verde, en cambio si esta por debajo, entonces es un dato rojo.

¿Cuál elegir?

Estudiaremos una respuesta natural al problema: ¿cómo elegir el hiperplano que nos entregue el **mayor margen** posible entre ambos grupos? Esto se traduce en reemplazar el problema de separar mediante una línea, en separar mediante una *cinta* de ancho máximo. A este modelo se le llama el separador de máximo margen que mencionamos antes.

El argumento de ocupar dicho modelo es que uno esperaría que si un modelo tiene un buen margen en los datos de entrenamiento, entonces el **error de generalización** (Cuanto se equivoca el modelo con datos nuevos) debiese ser bajo. Existe un argumento matemático, y a grandes rasgos es que a mayor margen y a medida aumentan los datos, podemos acotar la probabilidad de error (Al estilo desigualdad

de Markov) sin embargo los detalles se escapan de los contenidos del curso.

Continuando, en la fig. 27 (derecha) podemos visualizar el modelo. Notar que el margen esta definido solamente por algunos puntos. En la figura son dos, podrían ser más, pero nunca menos por la naturaleza " simétrica " del modelo.

Dichos puntos que definen el márgen son llamados los vectores de soporte (support vectors) que le entregan el nombre al método.

Un ejemplo sencillo para recordar que es lo que hace SVM es imaginar que el ejemplo de los gráficos trata sobre clasificar manzanas y naranjas:
A diferencia de otros métodos que aprenden analizando todos los datos y llegando a una respuesta (Cómo Naive Bayes, Regresión Lineal, etc...) de lo que en promedio podría ser una manzana, SVM mira la manzana mas naranjezca y la naranja mas manzanezca y con esos datos define un hiperplano separador.

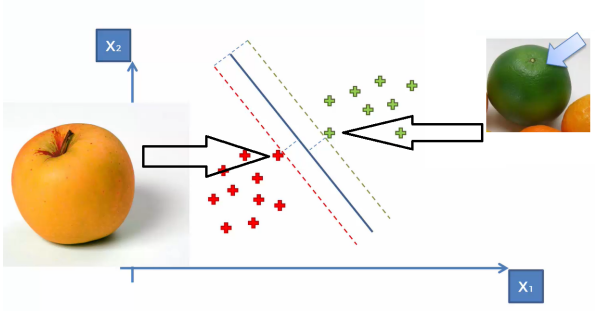


Fig. 28. Los vectores de soporte son la manzana más naranjezca y la naranja más manzanezca

7.3. Problema

Como ya mencionamos tenemos datos de entrenamiento $\{x_i\}_{1,...,N}$ que suponemos linealmente separables y deseamos encontrar un hiperplano de máximo margen que los separe.
Un hiperplano en general son los $x \in \mathbb{R}^n$ que cumplan al ecuación

$$w \cdot x + b = 0 \tag{251}$$

Donde $w \in \mathbb{R}^n$ es el vector perpendicular al hiperplano y $b \in \mathbb{R}$ es un parámetro. La idea es encontrar w y b tales que entreguen el hiperplano separador de mayor margen. Este problema no tiene solución única (Pues si w es solución entonces λw también). Para evitar ese problema, escalamos el plano de manera que los vectores de soporte (x_+ y x_-) sean tales que $w \cdot x_+ + b = 1$ y $w \cdot x_- + b = -1$ (Pueden haber mas vectores de soporte, pero para efectos del cálculo usaremos dos). Si bien aún no tenemos los vectores de soporte para hacer el escalamiento, este será parte de las restricciones del problema de optimización que resolveremos.
El márgen del hiperplano es la mitad de la diferencia entre ambos vectores de soporte, proyectada en la dirección $\frac{w}{||w||}$ (Ver figura 5).
Calculamos:

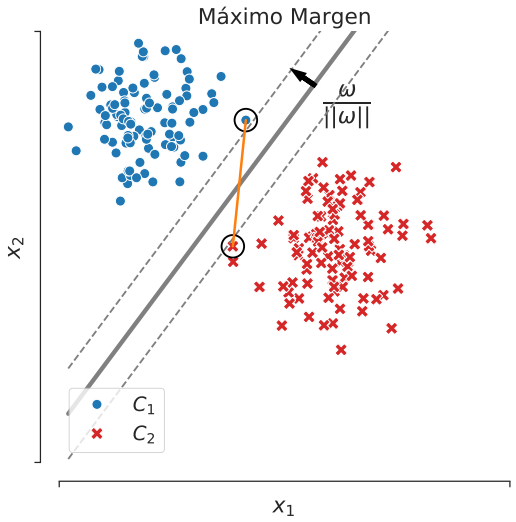


Fig. 29. En azul los vectores x_+ y x_-. En rojo el vector $(x_+ - x_-)$. En naranjo la componente del vector rojo, proyectada en la dirección definda por $\frac{w}{||w||}$. Notar que corresponde justamente al doble del margen

$$\begin{aligned}\frac{1}{2||w||}[w \cdot (x_+ - x_-)] &= \frac{1}{2||w||}[(w \cdot x_+) - (w \cdot x_-)] \\ &= \frac{1}{2||w||}[1 - (-1)] \\ &= \frac{1}{||w||}\end{aligned}$$

Definiendo a y_i como 1 si $w \cdot x_i + b \geq 1$ y $y_i = -1$ cuando $w \cdot x_i + b \leq -1$, podemos formular el problema del hiperplano de margén máximo como

$$\begin{aligned}\max_{w,b} \quad & \frac{1}{||w||} \\ \text{s.a} \quad & y_i(w \cdot x_i + b) \geq 1, \ i = 1, \ldots, N.\end{aligned}$$

Para evitar problemas de diferenciablidad, en realidad se ocupa la siguiente formulazión equivalente

$$\begin{aligned}\min_{w,b} \quad & \frac{1}{2}||w||^2 \\ \text{s.a} \quad & y_i(w \cdot x_i + b) \geq 1, \ i = 1, \ldots, N.\end{aligned}$$

Ahora estudiaremos el lagrangiano del problema (Y en consecuencia su dual). Este problema tiene una mejor estructura para optimizar, además que nos servirá para generalizar al caso no lineal.

El lagrangiano es

$$L(w, b, \alpha) = \frac{1}{2}||w||^2 - \sum_{i=1}^{N} \alpha_i (y_i(w \cdot x_i + b) - 1)$$

Las condiciones de primer orden nos dicen que

$$\frac{\partial L}{\partial w} = 0 \Rightarrow w = \sum_{i=1}^{N} \alpha_i y_i x_i \tag{252}$$

$$\frac{\partial L}{\partial b} = 0 \Rightarrow \sum_{i=1}^{N} \alpha_i y_i = 0 \tag{253}$$

Reemplazando (2) y (3) en $L(w, b, \alpha)$, maximizando (por ser dual) e imponiendo holgura complementaria, el problema queda

$$\begin{aligned} \max_{\alpha} \quad & \sum_{i=1}^{N} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{N} \alpha_i \alpha_j y_i y_j \langle x_i, x_j \rangle \\ \text{s.a} \quad & \sum_{i=1}^{N} \alpha_i y_i = 0 \\ & \alpha_i \geq 0 \end{aligned}$$

Este problema es de la forma QP (Quadratic Programming) y existen diferentes métodos para resolverlo de manera óptima. Notar la participación del producto punto $\langle \cdot, \cdot \rangle$ en este problema, pues este será el punto de partida del SVM no lineal.

Una vez resuelto el dual, la predicción de un nuevo punto x^* es de la forma

$$\hat{y}(x^*) = \text{sgn}\left(\left[\sum_{i=1}^{N} \alpha_i y_i \langle x_i, x^* \rangle \right] - b \right)$$

Por holgura complementaria tenemos que si x_i no está en el márgen, entonces $\alpha_i = 0$ de modo que x_i no aporta en la predicción $\hat{y}$. Esta propiedad es la que mencionabamos al principio: La predicción solo depende de los vectores de soporte.

Esta propiedad ayuda a resolver el problema de optimización de manera más rápida, ya que en realidad solo algunas variables duales α_i serán no nulas. Normalmente se ocupan heurísticas para encontrar rápidamente que vectores no son de soporte y así resolver un problema mas pequeño (Buscar, por ejemplo, el método de Sequential Minimal Optimization)

7.4. Soft Margin

El planteamiento anterior tiene dos debilidades. Una, la más obvia, es que no siempre podemos tener datos separables y la segunda es que, incluso si los datos son linealmente separables, nuestro clasificador puede ser muy sensible a nuevos datos. Ver por ejemplo la Fig.30.

Como se ve en la imagen, la llegada del nuevo dato (encerrado en rojo) cambia drásticamente nuestro modelo.

Para solucionar estos problemas (parcialmente) agregamos variables de holgura, que nos permitirán clasificar mal algunos datos. La idea es que, sacrificando algunos datos (muy probablemente outliers), podamos clasificar mejor la *mayoría* de los datos (Clasificarlos todos era claramente muy ambicioso) y así tener un modelo mas robusto.

Dicha formulazión del problema es la siguiente:

$$\begin{aligned} \min_{w,b,\xi} \quad & \frac{1}{2}||w||^2 + c\sum_{i=1}^{N} \xi_i \\ \text{s.a} \quad & y_i(w \cdot x_i + b) \geq 1 - \xi_i, \ i = 1, \ldots, N. \end{aligned}$$

donde c es un parámetro (Ya hablaremos de él).

Los ξ indican que tan mal clasificado está un punto. Si $\xi_i = 0$, entonces el dato x_i está al lado correcto del plano. Si $0 < \xi_i < 1$, entonces x_i está al lado correcto del plano, pero viola la restricción del margen. Finalmente si $\xi_i > 1$, entonces el punto esta al lado incorrecto del plano (Ver la siguiente figura)

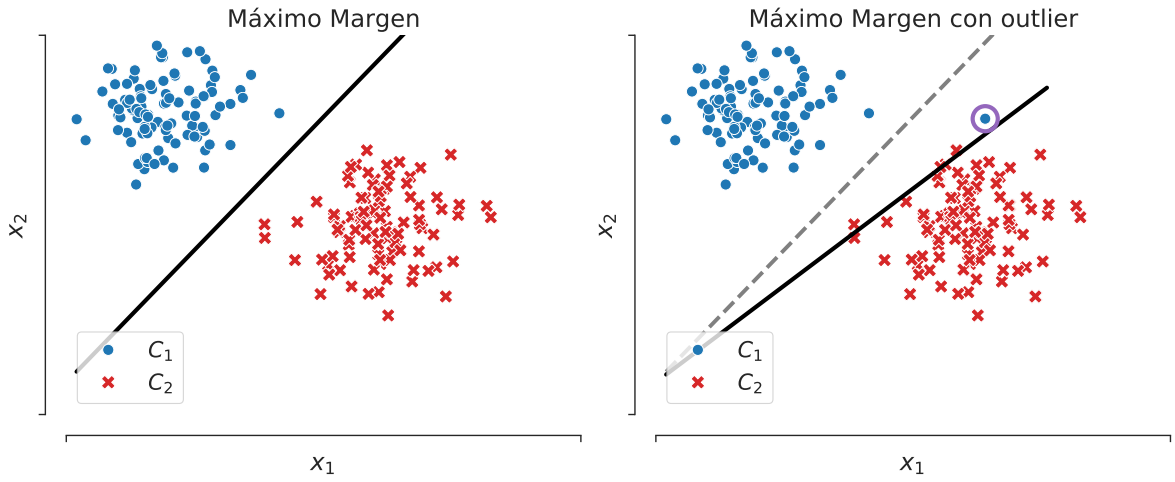


Fig. 30. Sensibilidad del máximo margen al agregar un nuevo dato.

El dual de este problema es (Usando Lagrangiano etc.)

$$\begin{aligned} \max_{\alpha} \quad & \sum_{i=1}^{N} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{N} \alpha_i \alpha_j y_i y_j \langle x_i, x_j \rangle \\ \text{s.a} \quad & \sum_{i=1}^{N} \alpha_i y_i = 0 \\ & 0 \leq \alpha_i \leq c \end{aligned}$$

Volveremos a este dual cuando veamos métodos no lineales.

Por último, ¿Qué hacer con el parámetro c? Guarda relación con el *bias-variance tradeoff*. La variable c es un parámetro de tolerancia. A mayor c, encontraremos un mayor márgen, lo que significa que nuestro modelo no estará sumamente ajustado a los datos (Lo que lo vuelve mas insesgado), pero probablemente se equivoque más (Tenga mayor varianza). Al revés, un c muy pequeño nos llevará más cerca del caso inicial, donde nuestro margen será más pequeño, y por ende, raramente violado (Poca varianza), pero nuestro modelo podría quedar sobreajustado a nuestros datos, logrando un mayor sesgo.En la práctica se usa *cross-validation* para encontrar un buen parámetro c.

7.5. Kernel Methods

Los métodos anteriores se ven interesantes, pero podría desmotivar el hecho que sólo funciona para datos linealmente separables (Con un poco de holgura en el caso de soft margin).

Un caso particular es el problema de clasificación XOR que se muestra en la siguiente figura

Dichos datos no son linealmente separables en $\mathbb{R}^2$, pero si consideramos el siguiente mapeo a $\mathbb{R}^3$

$$\phi(x_1, x_2) = (x_1, x_2, x_1 x_2) \tag{254}$$

Entonces es claro que el plano $z = 0$ en $\mathbb{R}^3$ será capaz de separar dichos puntos (Pues si $x_1 x_2 > 0$ entonces dicho punto será rojo y estará por sobre el plano $z = 0$, y análogamente sucede algo similar con los puntos azules).

La función ϕ se conoce normalmente como **feature map**. Es decir una función que entrega ciertos atributos de los datos (En este caso, por la forma del problema, un atributo importantes era justamente $x_1 x_2$).

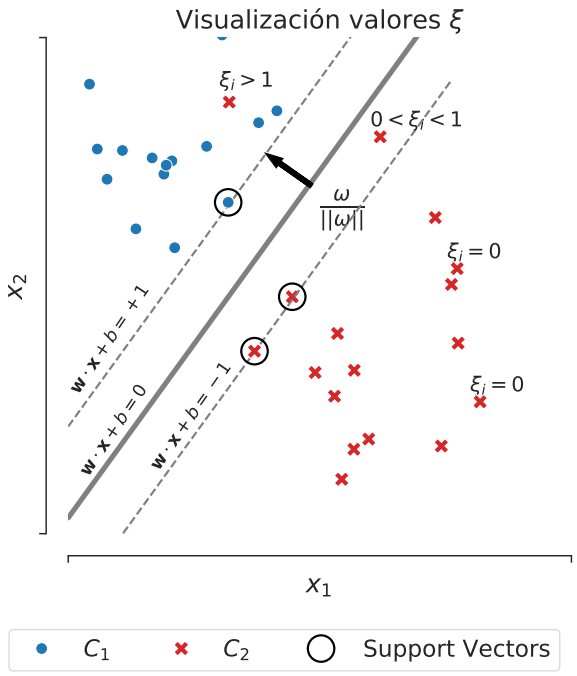


Fig. 31. Visualización de los valores de ξ

Dicha función puede ser muy díficil de definir en un problema particular, de hecho ya la función anterior dificilmente nos funcionará en otro problema que no sea XOR.

Sin embargo, muchas veces no nos interesa la forma explícita de la función, si no que más bien nos interesa trabajar con ella, en particular nos basta poder calcular $\phi(x)^T\phi(x)$(Ya vimos en las partes anteriores que dicho término, el producto punto de los datos, es relevante).

Para ello se ocupan los llamados **kernels**.

Def: Un (Mercer) Kernel es una función $K : X \times X \to \mathbb{R}$ tal que

- Es simétrica $K(x_1, x_2) = K(x_2, x_1)$
- Es definida positiva, es decir

$$\int_{X^2} K(x_1, x_2)g(x_1)g(x_2)dx_1dx_2 \geq 0$$

  para toda g continua.

Lema: Todo (mercer) Kernel se puede descomponer de la forma

$$K(x_1, x_2) = \langle \phi(x_1), \phi(x_2) \rangle$$

donde $\phi : X \to \mathbb{R}^D$ donde D puede ser ∞.

Es decir un kernel nos permite evaluar el producto punto (Qué será todo lo que necesitemos) en un espacio de alta dimensión, sin necesidad de encontrar explícitamente el feature map. Esto es lo que se conoce en machine learning como el **kernel trick**.

Distintos tipos de kernels, implicarán distintos tipos de feature maps. Por ejemplo

- **Radial Basis Function kernel:** Este kernel se defne por

$$K_{rbf}(x_1, x_2) = \sigma^2 \exp\left(-\frac{(x_1 - x_2)^2}{2l^2}\right)$$

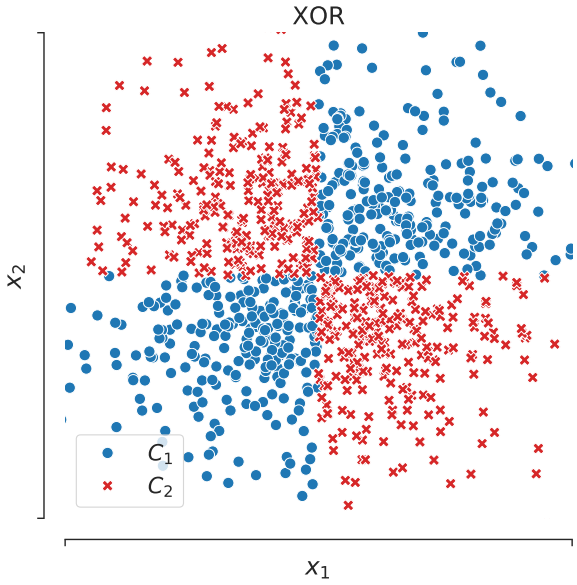


Fig. 32. Datos XOR no son linealmente separables

Las fronteras que entrega son suaves (infinitamente diferenciables) y tiene dos parámetros. El parámetro σ^2 es un parámetro de escala, y el parámetro l controla la oscilación de la curva.

- **Periodic kernel:**

$$K_{per}(x_1, x_2) = \sigma^2 \exp\left(-\frac{2\,\text{sen}^2(\pi|x_1 - x_2|/p)}{l^2}\right)$$

Este kernel es capaz de rescatar features periódicos en los datos (Controlados por el parámetro p). Los otros parámetros cumplen la misma función que el kernel anterior.

7.6. Ejemplo

Usemos el kernel trick para kernelizar un algoritmo ya conocido. EL rpoblema de ridge-regression consiste en que tienes un set de datos (y, X) con X una matriz e y un vector y queremos resolver

$$\min_{w \in \mathbb{R}^d} \frac{1}{n}||y - Xw||_2^2 + \lambda \sum_{i=1}^{d} w^2$$

donde λ es sólo un parámetro de regularización.
El resultado de este problema es explícito y dado por

$$\hat{w} = (X^T X + \lambda I)^{-1} X^T y$$

Para un nuevo punto (x', y') la predicción es:

$$\hat{y}' = y^T (XX^T + \lambda I)^{-1} X x'$$

Es aquí donde podemos ocupar el kernel trick, pues XX^T corresponde a los productos punto posibles entre las entradas de x. Podemos mapear a un espacio de alta dimensionaldiad cambiando a XX^T por una matriz K tal que

$$K_{ij} = K(x_i, x_j)$$

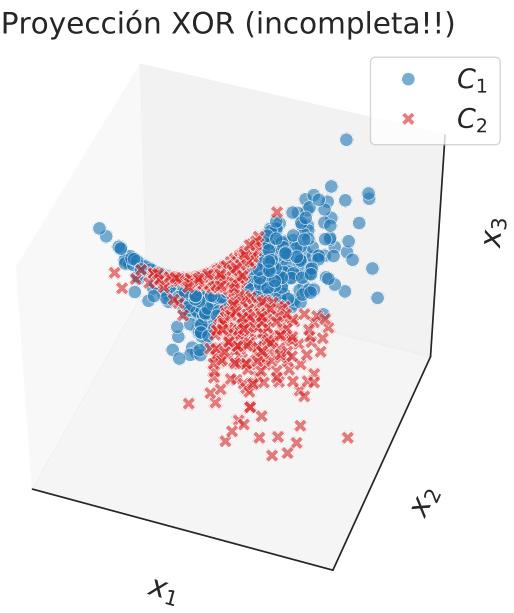


Fig. 33. Los puntos rojos y azules corresponden a los datos mapeados a través de ϕ. El plano $z = 0$ es claramente capaz de separar puntos rojos y puntos auzles en este nuevo espacio.

donde K es un kernel que nosotros estimamos conveniente. También el término Xx' lo cambiamos por un término k tal que $k_i = K(x_i, x')$.

Y así la predicción kernelizada queda

$$\hat{y}' = y^T(K + \lambda I)^{-1}k$$

Notar que una de las desventajas del método, a medida que entran más datos, deberemos invertir una matriz cada vez más grande.

7.7. Kernel SVM

Ahora los datos de entrenamiento los llevaremos a un espacio de dimensionalidad alta (donde si será posible separarlos linealmente) a través de un feature map $\phi(\cdot)$. Encontrar un feature map acorde es en general dificil e intractable.

El truco del kernel consiste en no tener que definir dicho feature map, si no que solo trabajar con el producto punto en dicho espacio de dimensionalidad alta el cual esta dado por un kernel

$$K(x_i, x_j) = \phi(x_i) \cdot \phi(x_j)$$

Dicho Kernel sí es conocido y tiene hiperparámetros los cuales podemos ajustar a nuestro problema. Para ello usaremos todo lo visto en las partes anteriores.

Mapeando los datos a través de ϕ y sin utilizar soft margin, el primal nos queda

$$\begin{aligned} \min_{w,b} \quad & \frac{1}{2}||w||^2 + c\sum_{i=1}^{N}\xi_i \\ \text{s.a} \quad & y_i(w \cdot \phi(x_i) + b) \geq 1 - \xi_i,\ i = 1, \ldots, N. \end{aligned}$$

Pasandonos al dual, tendremos que

$$\begin{aligned} \max_{\alpha} \quad & \sum_{i=1}^{N} \alpha_i - \frac{1}{2} \sum_{i=1}^{N} \alpha_i \alpha_j y_i y_j < \phi(x_i), \phi(x_j) > \\ \text{s.a} \quad & \sum_{i=1}^{N} \alpha_i y_i = 0 \\ & 0 \leq \alpha_i \leq c \end{aligned}$$

Notar que en este problema el único término donde aparece $\phi(x)$ es en el producto punto $\langle \phi(x_i), \phi(x_j) \rangle$ el cual es igual a $K(x_i, x_j)$. Esto es justamente el Kernel Trick.

El problema así nos queda

$$\begin{aligned} \max_{\alpha} \quad & \sum_{i=1}^{N} \alpha_i - \frac{1}{2} \sum_{i=1}^{N} \alpha_i \alpha_j y_i y_j K(x_i, x_j) \\ \text{s.a} \quad & \sum_{i=1}^{N} \alpha_i y_i = 0 \\ & 0 \leq \alpha_i \leq c \end{aligned}$$

El cual si es tratable: una vez calculados todos los $\binom{n}{2}$ productos de la forma $K(x_i, x_j)$, lo unico que queda es resolver un problema QP (El cual, por las condiciones de Mercer del kernel, tendrá solución, pues la matriz será definida positiva).

En la siguiente imagen vemos un ejemplo de SVM con dos kernels diferentes.

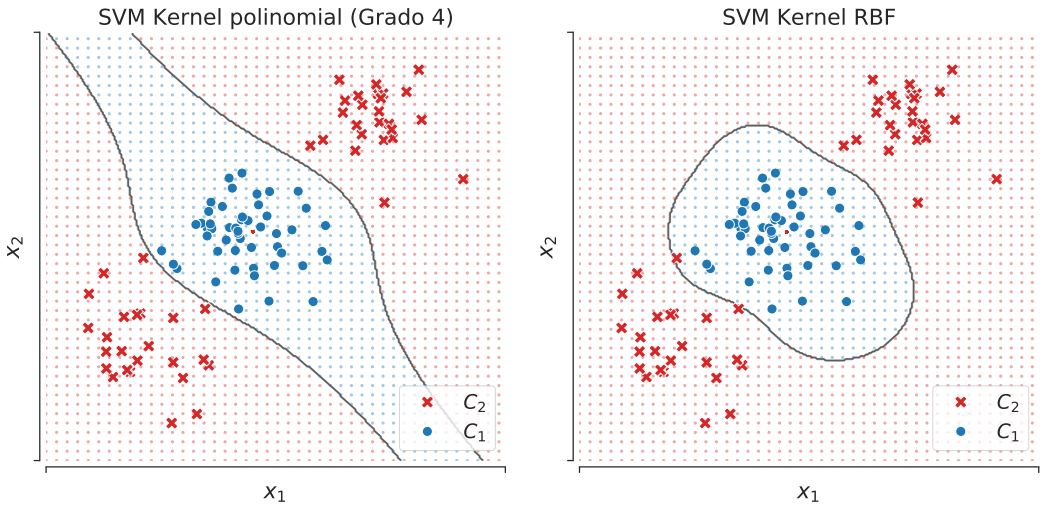


Fig. 34. Clasificación con distinto kernels.

A la izquierda se ocupo un kernel polinomial de grado 3, dicho kernel permite un poco mas de curvatura en los límites del clasificador. A la derecha se ocupó un kernel RBF, el cual es capaz de encontrar fronteras de decisión más curvadas (Pero infinitamente diferenciables).

Tener en cuenta que esta manera de resolver el problema tiene una tremenda ventaja computacional. Mover los datos a un espacio más grande mediante explícitamente definiendo $\phi(x)$, puede ser muy costoso, o incluso imposible en el caso que el espacio de llegada sea infinitamente dimensional. Usando funciones kernel dicha función queda implícita en el problema, dejandonos incluso trabajar en el caso infinito-dimensional (En el caso de RBF por ejemplo)

8. Procesos Gaussianos

Como se vio en capítulos anteriores, el problema de regresión busca encontrar una función $y = f(x)$, dado un conjunto de pares de la forma $\mathcal{D} = \{(x_i, y_i)\}_{i=1}^N$. Dento de los métodos vistos para resolver el problema de regresión, se vio el de regresión lineal, lineal en los parámetros y no lineals. Una característica en común que tienen estos métodos es que el proceso de entrenamiento consiste en encontrar un número **fijo** de parámetros, que minimicen cierta función objetivo, donde la cantidad de parámetros y la forma del modelo es parte del diseño. A este tipo de modelos se les llama *paramétricos*.

En contraste, se encuentran los modelos *no paramétricos* los cuales **no** tienen un número fijo de parámetros, donde pueden llegar a ser en algunos casos infinito. Un ejemplo es el algoritmo k-vecinos más cercanos (KNN), donde los puntos del conjunto de entrenamiento son usados para clasificar nuevas muestras. Otro ejemplo son las máquinas de soporte vectorial (SVM) donde al entrenar se obtienen los vectores de soporte.

Es importante hacer la distinción entre parámetros que se aprenden y los parámetros de modelo, muchas veces llamados hyperparámetros, donde estos últimos pueden ser fijos independiente si el método es paramétrico y no paramétrico; esto se ve en el caso de SVM donde los parámetros serían los vectores de soporte, y los hyperparámetros serían el tipo de kernel, los parámetros de dicho kernel y los demás valores elegidos de antemano que definen el tipo de modelo.

En este capítulo introduciremos un método no paramétrico probabilístico de regresión no lineal, llamados Procesos Gaussianos ($\mathcal{GP}$), este en vez de encontrar un candidato único de la función a estimar, define una distribución sobre funciones $\mathbb{P}(f)$, donde $f(\cdot)$ es una función de un espacio de entrada $\mathcal{X}$ a los reales, $f : \mathcal{X} \to \mathbb{R}$. Esto tiene la virtud que permitirá cuantificar la incertidumbre puntual que existe en la predicción de nuestro modelo que servirá en forma de intervalos de confianza para la distribución Gaussiana.

Partiremos definiendo un $\mathcal{GP}$ como una distribución a priori sobre funciones, y mostraremos que la densidad posterior se puede encontrar de forma exacta y que esta también es un $\mathcal{GP}$, conservando sus propiedades.

$$\mathbb{P}(f|\mathcal{D}) = \frac{\mathbb{P}(f)\mathbb{P}(\mathcal{D}|f)}{\mathbb{P}(\mathcal{D})} \tag{255}$$

Es de notar que si $\mathcal{X}$ tiene cardinalidad infinita (por ejemplo $\mathcal{X} = \mathbb{R}$), $f(\cdot)$ puede ser visto como un vector infinito dimensional. Como en la práctica no podemos trabajar con un vector infinito dimensional, dados n puntos $\{x_i\}_1^n \subset \mathcal{X}$ podemos definirnos el vector n-dimensional de valores de la funciones evaluados en dichos puntos $f(\mathbf{x}) = [f(x_1), \ldots, f(x_n)]$.

Definición: Un Proceso Gaussiano ($\mathcal{GP}$) es una colección de variables aleatorias, tal que para cualquier subconjunto finito de puntos, estos tienen una distribución conjuntamente Gaussiana.

Al aplicar esta definición a nuestro caso anterior, $\mathbb{P}(f)$ será un $\mathcal{GP}$ y para cualquier conjunto finito $\{x_i\}_1^n \subset \mathcal{X}$, la distribución de $\mathbb{P}(f(\mathbf{x}))$ es Gaussiana multivariada $f(\mathbf{x}) = [f(x_1), \ldots, f(x_n)])$. En este caso las variables aleatorias representan el valor de la función $f(x_i)$ en la posición x_i.

Un $\mathcal{GP}$ queda completamente caracterizado por su función de media $m(\cdot)$ y función de covarianza $K(\cdot, \cdot)$, de esta forma para cualquier conjunto finito podemos encontrar la distribución. Definimos estas funciones como

$$m(x) = \mathbb{E}\left\{f(x)\right\} \tag{256}$$
$$K(x, x') = \mathbb{E}\left\{\left(f(x) - m(x)\right)\left(f(x') - m(x')\right)\right\} \tag{257}$$

Y de esta forma podemos escribir el proceso como:

$$f \sim \mathcal{GP}(m(\cdot), K(\cdot, \cdot)) \tag{258}$$

Donde para un conjunto finito tenemos que la marginal resulta de la forma:

$$f(\mathbf{x}) \sim \mathcal{N}(m(\mathbf{x}), K(\mathbf{x}, \mathbf{x})) \tag{259}$$

Hasta el momento hemos hablado del espacio de entrada $\mathcal{X}$ como genérico, un caso común es definir los $\mathcal{GP}$ sobre el tiempo ($\mathbb{R}^+$), es decir que los x_i son instantes de tiempo. Es de notar que este no es el único caso, y se podría definir sobre un espacio más general, por ejemplo $\mathbb{R}^d$.

Otro punto a notar es que como estamos hablando de una colección (no necesariamente finita) de variables aleatorias, es necesario que se cumpla la propiedad de marginalización (o llamada consistencia[14]). Esta propiedad se refiere a que si un $\mathcal{GP}$ define una distribución multivariada para digamos dos variables $(y_1, y_2) \sim \mathcal{N}(\mu, \Sigma)$ entonces también debe definir $y_1 \sim \mathcal{N}(\mu_1, \Sigma_{11})$ donde μ_1 es la componente respectiva del vector μ y Σ_{11} la submatriz correspondiente de Σ. En otras palabras, el tomar un subconjunto más grande de puntos no cambia la distribución de un subconjunto más pequeño. Y podemos notar que esta condición se cumple si tomamos la función de covarianza definida anteriormente.

8.1. Muestreo de un prior $\mathcal{GP}$

Como fue mencionado, un $\mathcal{GP}$ define un *prior* sobre funciones, por lo que, antes de ver ningún dato se podría obtener una muestra de este proceso dado una función de media y covarianza. Un supuesto común es asumir la función de media $m(\cdot) = 0$ por lo que solo nos queda definir una función de covarianza o kernel, un ejemplo de kernel es el *Exponencial Cuadrático* (Square Exponential), también conocido como RBF (Radial Basis function) o Kernel Gaussiano (en general se evita este nombre porque este kernel no tiene relación con la distribución de los datos).

$$K_{SE}(x, x') = \sigma^2 \exp\left(-\frac{(x - x')^2}{2\ell^2}\right) \tag{260}$$

Donde en este caso los parámetros son interpretables (y como veremos más adelante pueden ser aprendidos a través de un conjunto de entrenamiento) donde σ^2 es la varianza de la función, notar que esta es la diagonal de la matriz covarianza. El parámetro ℓ es conocido como el *lenghtscale* que determina que tan lejos tiene influencia un punto sobre otro, donde en general un punto no tendrá influencia más allá de ℓ unidades alrededor.

Como sabemos, las funciones definidas por el $\mathcal{GP}$ son vectores infinito dimensionales, por lo que no podemos muestrear de toda la función, pero tomando una cantidad suficiente de puntos podemos graficar muestras de un $\mathcal{GP}$ dada una función de covarianza. Tomando un $\mathcal{GP}$ con media cero ($m(\cdot) = 0$) y kernel SE, muestras del proceso para distintos valores de ℓ obtenemos la Fig.35, donde el área sombreada corresponde al intervalo de confianza del 95 %. Se puede ver que el parámetro ℓ controla que tan erráticas son las funciones, donde a medida que va a aumentando las muestras se vuelven funciones más suaves.

[14]Para más detalles puede ver el teorema de consistencia de Kolmogorov

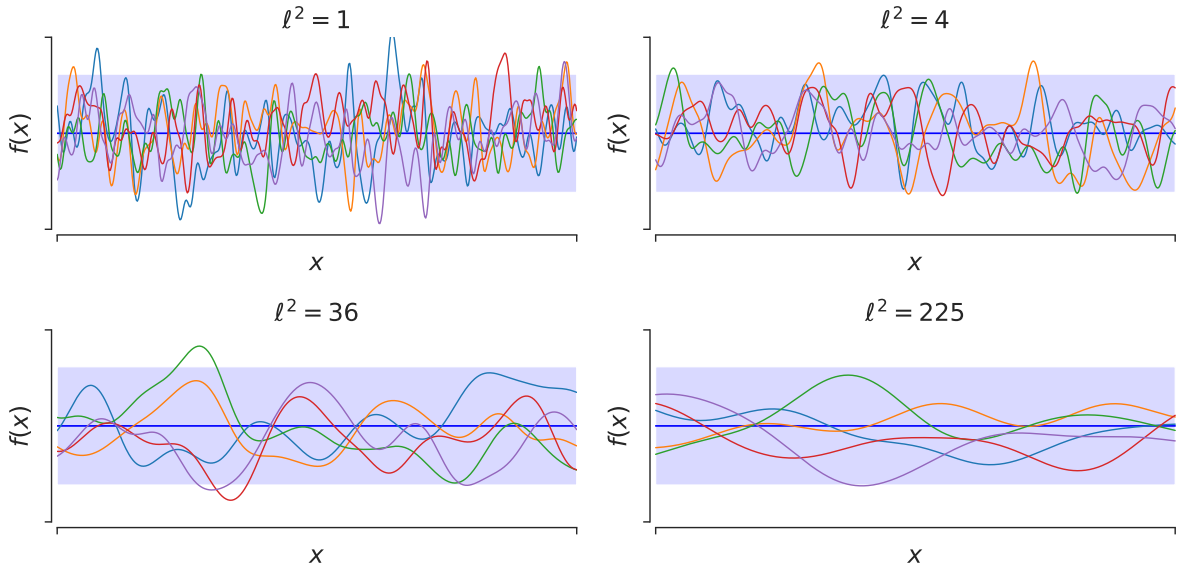


Fig. 35. Muestras de un prior $\mathcal{GP}$ con kernel SE, para distintos *lenghtscales* (ℓ) y función media $m(\cdot) = 0$, la parte sombreada corresponde al intervalo de confianza del 95 %. Se puede ver que a mayor ℓ las funciones se van volviendo más suaves.

8.2. Incorporando Información: Evaluación sin ruido

Ahora que ya podemos muestrear de nuestro prior, nos interesaría incorporar las observaciones que tenemos de la función a nuestro modelo. Para esto, primero consideramos el caso en que las observaciones son sin ruido, es decir tenemos observaciones de la forma $\{x_i, f(x_i)\}_1^n$, donde tenemos el valor real de nuestra función en los puntos $[x_1, \ldots, x_n] = X$.

Luego, tenemos el par de entradas y observaciones $(X, f(X))$ digamos que queremos realizar una predicción en el conjunto X_* de n_* puntos, luego la distribución conjunta es de la forma:

$$\begin{bmatrix} f(X) \\ f(X_*) \end{bmatrix} \sim \mathcal{N}\left(\begin{bmatrix} m(X) \\ m(X_*) \end{bmatrix}, \begin{bmatrix} K(X,X) & K(X,X_*) \\ K(X_*,X) & K(X_*,X_*) \end{bmatrix} \right) \tag{261}$$

Donde la submatriz $K(X, X_*)$ es de $n \times n_*$, $K(X, X)$ de $n \times n$ y así respectivamente para cada submatriz.

Pero a nosotros nos interesa encontrar la densidad posterior de $f(\cdot)$, y dada las observaciones y una función de covarianza podemos evaluar la verosimilitud, que también es Gaussiana, lo que nos lleva a un punto clave de los Processos Gaussianos.

Dado un prior $\mathcal{GP}$ sobre $f(\cdot)$ y una verosimilitud Gaussiana, la posterior sobre $f(\cdot)$ es también un $\mathcal{GP}$.

Luego, podemos condicionar sobre las observaciones $(X, f(X))$ y obtenemos

$$f(X_*)|f(X), X \sim \mathcal{N}(m_{X_*|X}, \Sigma_{X_*|X}) \tag{262}$$

Donde la media y covarianza son:

$$m_{X_*|X} = m(X_*) + K(X_*, X)K^{-1}(X, X)(f(X) - m(X)) \tag{263}$$

$$\Sigma_{X_*|X} = K(X_*, X_*) - K(X_*, X)K^{-1}(X, X)K(X, X_*) \tag{264}$$

Ahora, dado un conjunto de observaciones y dada una función de covarianza, podemos obtener la densidad posterior. Para mostrar esto tomemos el caso de hacer regresión para una función conocida, para la cual tenemos observaciones sin ruido muestreadas no uniformemente, con estas observaciones queremos encontrar la función real de las que provienen; para esto usamos un prior $\mathcal{GP}$ con función media nula y kernel SE (por el momento tendrá parámetros fijos), nos damos un rango donde queremos hacer predicción y condicionamos en las observaciones usando la Ec.(263). En este caso las observaciones corresponden al 15 % de los puntos generados por nuestra función sintética.

Esto se muestra en la Fig.36, donde podemos ver que la media de la posterior pasa por las observaciones sin incertidumbre asociada, es decir que para estos puntos se tiene una posterior degenerada pues no hay varianza. Se puede ver que a medida que la predicción se aleja de las observaciones el intervalo de confianza (al que lo podemos asociar con incertidumbre del modelo) va creciendo.

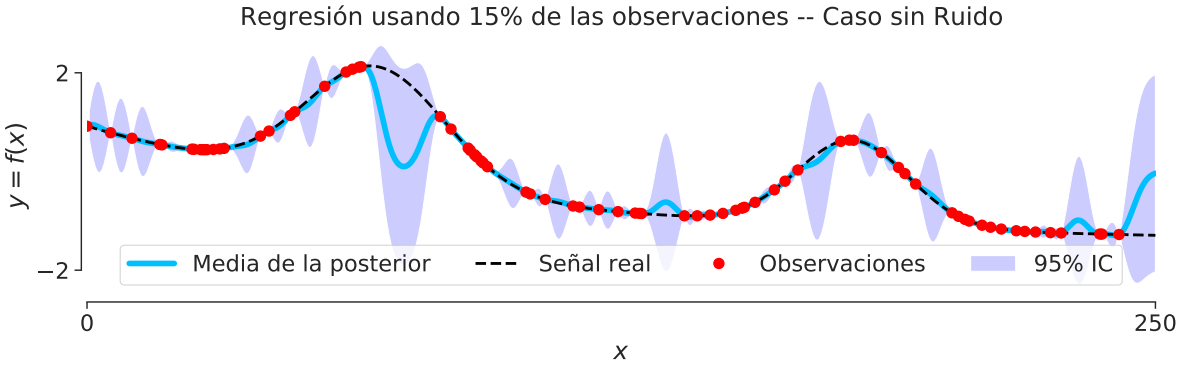


Fig. 36. Regresión con $\mathcal{GP}$ para señal sintetica usando el 15 % de los datos muestreados de forma no uniforme, utilizand un $\mathcal{GP}$ de media nula y kernel SE.

8.3. Incorporando Información: Evaluación con ruido

En aplicaciones reales el caso de tener observaciones sin ruido es poco habitual, por lo que si queremos incorporar la incertidumbre real debemos tomar en cuenta errores de medición en nuestro modelo. Tomemos el caso de tener observaciones contaminadas con ruido i.i.d (independientes idénticamente distribuidas) donde las observaciones serán de la forma $y_i = f(x_i) + \eta$ donde $\eta \sim \mathcal{N}(0, \sigma_n^2)$, donde ahora nuestro conjunto de observaciones es de la forma (X, Y) donde $Y = f(X) + \eta$.

Lo que en nuestro modelo equivale a agregar un término a la función de covarianza

$$cov(Y) = K(X, X) + \sigma_n^2 \mathbb{I} \tag{265}$$

Donde si tenemos el mismo caso anterior, observaciones (X, Y) y queremos evaluar en X_*, la conjunta queda

$$\begin{bmatrix} Y \\ f(X_*) \end{bmatrix} \sim \mathcal{N}\left(\begin{bmatrix} m(X) \\ m(X_*) \end{bmatrix}, \begin{bmatrix} K(X,X) + \sigma_n^2 \mathbb{I} & K(X, X_*) \\ K(X_*, X) & K(X_*, X_*) \end{bmatrix} \right) \tag{266}$$

Notemos que el termino de ruido solo es agregado al subbloque correspondiente a las observaciones, no se agrega el termino en los otros subbloques pues buscamos hacer una predicción de la función latente $f(\cdot)$ y no una versión ruidosa de esta. Igual que en el caso sin ruido, podemos condicionar esta conjunta a las observaciones y obtenemos

$$f(X_*)|Y, X \sim \mathcal{N}(m_{X_*|X}, \Sigma_{X_*|X}) \tag{267}$$

Donde la media y covarianza son:

$$m_{X_*|X} = m(X_*) + K(X_*, X)[K(X, X) + \sigma_n^2 \mathbb{I}]^{-1}(Y - m(X)) \tag{268}$$

$$\Sigma_{X_*|X} = K(X_*, X_*) - K(X_*, X)[K(X, X) + \sigma_n^2 \mathbb{I}]^{-1} K(X, X_*) \tag{269}$$

Si tomamos el mismo ejemplo anterior, pero añadimos el ruido al modelo, obtenemos la predicción de la Fig.37. En este caso podemos ver que la media de la posterior no necesariamente coincide su valor con el de la observación, pues se toma en cuenta la incertidumbre en las observaciones mismas, también se ve que no se obtienen soluciones degeneradas incluso en zonas donde hay observaciones aglomeradas.

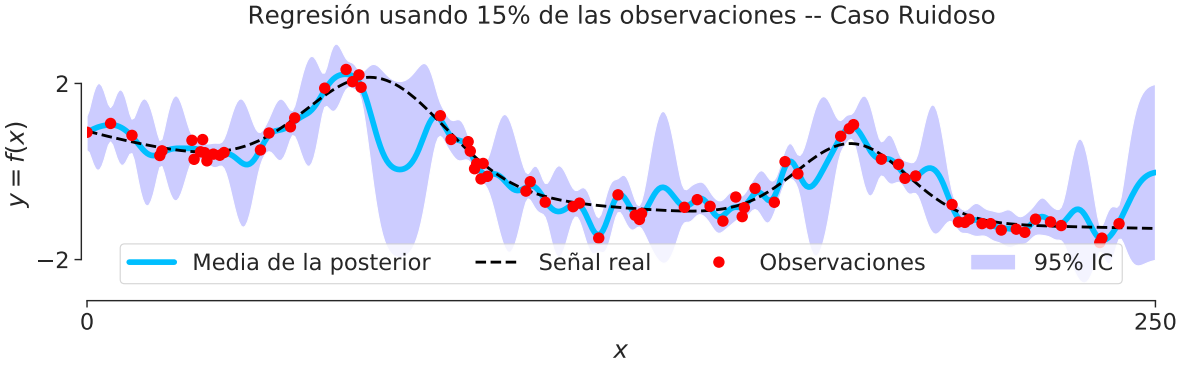


Fig. 37. Regresión con $\mathcal{GP}$ para señal sintetica usando el 15 % de los datos muestreados de forma no uniforme y contaminados con ruido Gaussiano, utilizando un $\mathcal{GP}$ de media nula y kernel SE.

8.4. Entrenamiento y optimización de un $\mathcal{GP}$

hasta el momento nos hemos dado la función de covarianza y sus parámetros, por lo que aún no hemos hecho realizado el *aprendizaje* de nuestro $\mathcal{GP}$, que es lo que haremos a continuación.

¿Qué es entrenar un $\mathcal{GP}$?

Vimos que dada una función de covarianza podemos representar el proceso, y podemos encontrar analíticamente la densidad posterior de nuestra función $f(\cdot)$ condicionando a las observaciones. Pero la forma que tendrá la posterior y la función fuera de las observaciones dependerá fuertemente en nuestra función kernel escogida, en este sentido, para un kernel dado nos gustaría encontrar los parámetros de este que mejor representen nuestra función a estimar.

Nos referiremos a entrenar u optimizar un $\mathcal{GP}$ cuando queremos obtener los hyperparámetros, es decir los parámetros del kernel (los denotamos θ) y la varianza del ruido (la denotamos σ_n^2) si es que aplica.

Para esto nos gustaría poder comparar funciones de covarianza, o mejor aún un funcional que podamos optimizar, afortunadamente podemos usar la *verosimilitud marginal*, obtenida marginalizando sobre la función $f(\cdot)$, donde dado un conjunto de entrenamiento $(X, Y) = \{(x_i, y_i)\}_{i=1}^n$, esta dada por

$$\mathbb{P}(Y|X, \theta, \sigma) = \int \mathbb{P}(Y|f, X, \theta, \sigma_n) p(f|X, \theta, \sigma) df \tag{270}$$

$$= \frac{1}{(2\pi|\mathbf{K}_y|)^{\frac{n}{2}}} \exp\left(-\frac{1}{2}(Y - \mathbf{m})^T \mathbf{K}_y^{-1}(Y - \mathbf{m})\right) \tag{271}$$

Donde $\mathbf{m} = m(X)$ y $\mathbf{K}_y = K_\theta(X,X) + \sigma_n^2 \mathbb{I}$, la matriz de covarianza dados los parámetros θ agregando el término de la diagonal correspondiente al ruido. De la misma forma que lo hacemos con otros modelos probabilísticos, en vez de maximizar la verosimilitud, en conveniente minimizar la log-verosimilitud negativa (NLL) dada por la expresión:

$$NLL = -\log \mathbb{P}(Y|X,\theta,\sigma_n) \tag{272}$$

$$NLL = \underbrace{\frac{1}{2}\log|\mathbf{K_y}|}_{\text{Penalización por complejidad}} + \underbrace{\frac{1}{2}(Y-\mathbf{m})^T\mathbf{K}_y^{-1}(Y-\mathbf{m})}_{\text{Data fit (Única parte que depende de } Y)} + \underbrace{\frac{n}{2}\log 2\pi}_{\text{Constante de normalización}} \tag{273}$$

De izquierda a derecha, el primer término tiene el rol de penalizar por la complejidad del modelo, y vemos que depende solo del kernel y las entradas; el segundo término cuantifica que tan bien se ajusta el modelo a los datos, y es la única componente que depende de las observaciones Y (ruidosas) de la función, el último término es una constante de normalización.

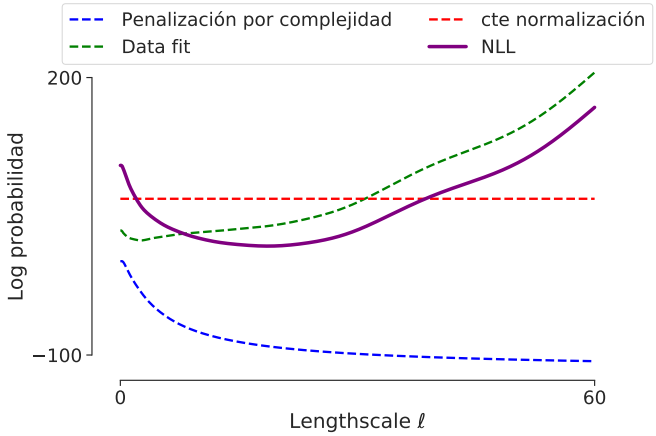


Fig. 38. Log verosimilitud marginal negativa (NLL) en función del *lengthscale* (ℓ) para señal sintetica, se mantienen constantes los otros parámetros del $\mathcal{GP}$.

Siguiendo con los ejemplos anteriores, para el mismo conjunto de observaciones ruidosas, se calculan las tres componentes de la NLL, utilizando un kernel SE, para este caso, se dejan fijo tanto la varianza de la señal como la varianza del ruido y se varia el *lengthscale* ℓ del kernel, en la Fig.38.

Al ir variando ℓ se ve que la penalización por complejidad va disminuyendo, pues a mayor ℓ menos complejas son las funciones (recordar Fig.35, a mayor ℓ más suaves y regulares las muestras), también vemos que la componente del ajuste de datos comienza a decaer y luego se mantiene en incremento, pues a mayor *lengthscale* el modelo se vuelve menos flexible, por lo que no es capaz de ajustarse de manera correcta a los datos. De esta forma, el proceso de entrenamiento dará preferencia a funciones que se ajusten bien a los datos sin ser tan complejas, ahora viendo el valor total del NLL vemos que este alcanza su mínimo en ℓ en el rango de $[10-30]$.

Ya tenemos una noción de que es entrenar un $\mathcal{GP}$, queremos minimizar la NLL (Ec.(273)) y encontrar los parámetros del kernel y del ruido (si aplica). Ahora discutiremos formas de optimizar este funcional.

¿Cómo se entrena?

Como contamos con una expresión cerrada para la NLL, podemos utilizar métodos clásicos de optimización, una opción es calcular el gradiente de esta función objetivo y aplicar algún método basado

en gradiente, como L-BFGS; otra es utilizar el método de Powell que no requiere que la función sea diferenciable, por lo que no utiliza gradiente.

Siguiendo ejemplos anteriores, usando la misma señal sintética y las mismas observaciones ruidosas de la Fig.37, ahora optimizamos nuestro $\mathcal{GP}$ utilizando L-BFGS, lo que nos entrega como resultado el mostrado en la Tabla.2 donde comparamos los parámetros del $\mathcal{GP}$ sin entrenar usado en la Fig.37, podemos ver que no difiere mucho en cuanto a las varianzas, pues como es una señal sintética poseíamos control sobre la varianza del ruido, el valor obtenido luego de optimizar es cercano al valor real (0.2), vemos que el *lengthscale* óptimo es concordante con lo analizado en la Fig.38.

Por último, podemos ver la predicción usando este $\mathcal{GP}$ optimizado en la Fig.39 donde vemos que se tiene una del proceso más concordante con la función real, esto se ve especialmente en la zona con pocas observaciones, entre 50 y 100 para x, donde el $\mathcal{GP}$ sin entrenar presentaba mucha incertidumbre en esa zona, en cuanto ahora se tiene un intervalo de confianza más bien limitado pues la señal no era muy compleja.

	σ_{ruido}	ℓ	$\sigma_{\text{señal}}$	NLL
Sin entrenar	0.2	3.1622	1	**55.3538**
Entrenado	0.2067	18.7267	0.9956	**17.6945**

Tabla 2. Resultado optimización $\mathcal{GP}$ y comparación con los parámetros sin entrenar.

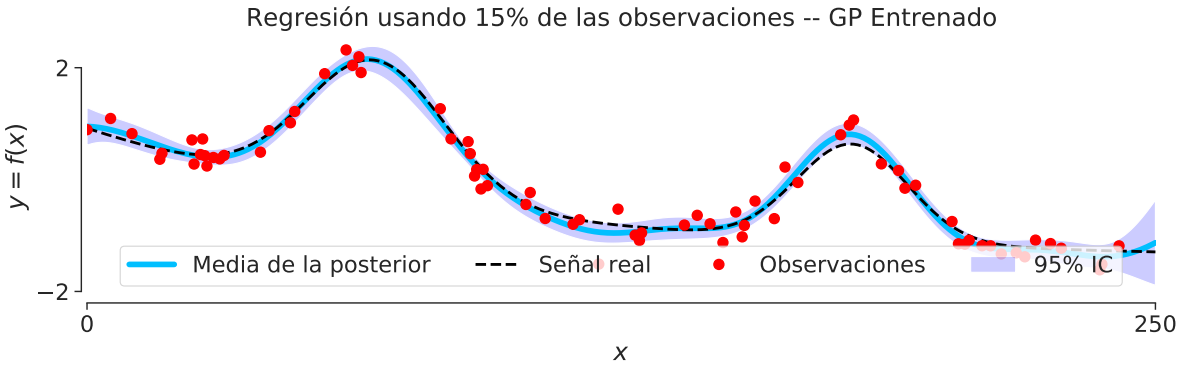


Fig. 39. Regresión con $\mathcal{GP}$ para señal sintetica usando el 15 % de los datos muestreados de forma no uniforme y contaminados con ruido Gaussiano, utilizando un $\mathcal{GP}$ de media nula y kernel SE; Modelo optimizado utilizando L-BFGS.

La desventaja de utilizar métodos tradicionales de optimización es que estos entregan una solución puntual (en el sentido de un valor para cada parámetro, el $\mathcal{GP}$ sigue siendo una distribución sobre funciones), en distintas aplicaciones puede que un valor fijo de parámetros no represente de manera idónea el proceso latente a estudiar. Una opción sería computar la densidad posterior de los parámetros del kernel condicionado a las observaciones. Lamentablemente en la mayoría de los casos no existe una expresión cerrada para esta posterior, por lo que debemos recurrir a métodos de *inferencia aproximada* como Markov Chain Monte Carlo (MCMC) o Inferencia Variacional.

8.4.1. Complejidad Computacional

Es importante reconocer una de las principales desventajas de utilizar un $\mathcal{GP}$ cuando se cuenta con una gran cantidad de datos, esto es, su costo computacional. Recordando, cuando queremos entrenar nuestro $\mathcal{GP}$ vamos a minimizar la log verosimilitud marginal negativa (NLL), mostrada en la Ec.(273), donde al observar en segundo término vemos que es la operación más costosa siendo $\mathcal{O}(n^3)$ con respecto

al número de puntos de entrenamiento n. Hay que tomar en cuenta que la evaluación de la NLL se debe hacer en cada iteración del método de optimización elegido.

En cuanto a la evaluación, esta está dada por la Ec.(268), donde en este caso vemos que es de orden cuadrático $\mathcal{O}(n^2)$ con respecto al número de puntos de test. Es de notar que esta vez no tomamos en cuenta la inversa de la matriz de Gram del conjunto de entrenamiento, pues puede ser precalculada y ser usada para múltiples conjuntos de test.

Lo anterior mencionado limita la cantidad de problemas que se pueden abordar usando estos procesos, sin embargo existen formas de obtener aproximaciones que siguen manteniendo la estructura y deseables propiedades de los $\mathcal{GP}$ a un menor costo computacional, estos reducen el orden de $\mathcal{O}(n^3)$ a $\mathcal{O}(nm^2)$ en entrenamiento y $\mathcal{O}(m^2)$ en test, con $m < n$. Un ejemplo de estos son los *Sparse Gaussian Processes* que utilizan una aproximación de rango bajo para la matriz de covarianza, utilizando *pseudo-inputs* en vez el conjunto de entrenamiento completo.

8.5. Funciones de covarianza (Kernels)

Una función de covarianza es una función de dos argumentos donde cualquier matriz que se obtiene como resultado en la evaluación de un conjunto de puntos (la llamaremos matriz de Gram) es semidefinida positiva.

Hasta el momento solo hemos utilizado un tipo de función de covarianza, el llamado kernel *Squared Exponential* (SE), conocido también como kernel RBF, dado por la Ec.(274). En esta sección mostraremos distintos tipos de funciones de covarianza y los distintos tipos de funciones que generan.

$$K_{SE}(x, x') = \sigma^2 \exp\left(-\frac{(x - x')^2}{2\ell^2}\right) \tag{274}$$

Es importante denotar tipos de familias de funciones de covarianza, si la función solo depende de la diferencia, es decir $k(x, x') = k(x - x')$ se le llamará *kernel estacionario*, más aún, si depende solo de la norma de la diferencia $k(x, x') = k(|x - x'|)$ se le llamará *kernel isométrico*, un ejemplo de esto es el kernel SE.

Es de notar que kernels estacionarios hacen que la covarianza entre puntos sea invariante a traslaciones en el espacio de entradas. Una noción importante es que un kernel puede ser visto como una medida de similaridad entre puntos, y en el caso de kernels estacionarios, mientras más cercanos estén dos puntos más símiles serán.

Kernel *Rational Quadratic* (RQ)

Este kernel viene dado por la Ec.(275), este puede ser interpretado como una suma infinita de kernels SE con distintos *lenghtscale*, donde α es un parámetro de variación de escala, notar que cuando $\alpha \to \infty$ el kernel tiende a uno SE. En la Fig.40 se muestra el kernel RQ, a la izquierda se ve el valor de la covarianza en función de la diferencia de los argumentos $x - x'$, a la derecha se muestran diferentes muestras de funciones con este kernel.

$$K_{RQ}(x, x') = \sigma^2 \left(1 + \frac{(x - x')^2}{2\alpha\ell^2}\right)^{-\alpha} \tag{275}$$

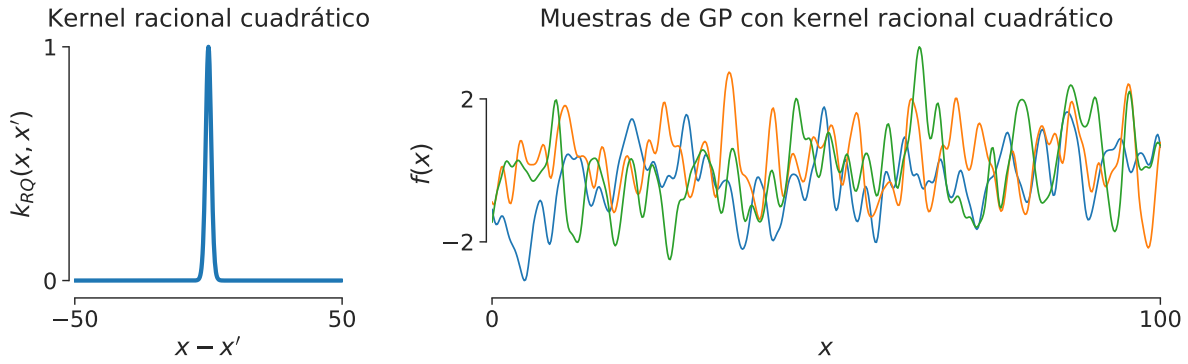


Fig. 40. Kernel *Rational Quadratic*, en la izquierda se muestra la covarianza en función de su argumento $\tau = x - x'$, a la derecha de un $\mathcal{GP}$ usando un kernel RQ.

Kernel Periódico

Como su nombre lo indica, este kernel, dado por la Ec.(276), permite modelar funciones periódicas, donde el parámetro p controla el periodo de la función. Una extensión de este el kernel localmente periódico, dado por la Ec.(277) que es un kernel SE ponderado por uno periódico, este da la libertad de tener funciones con una estructura periódica local.

$$K_P(x, x') = \sigma^2 \exp\left(-\frac{2\sin^2(\pi|x - x'|/p)}{\ell^2}\right) \tag{276}$$

$$K_{LP}(x, x') = \sigma^2 \exp\left(-\frac{(x - x')^2}{2\ell^2}\right) \exp\left(-\frac{2\sin^2(\pi|x - x'|/p)}{\ell^2}\right) \tag{277}$$

En la Fig.41 se muestra el kernel periódico, a la izquierda se ve el valor de la covarianza en función de la diferencia de los argumentos $x - x'$, a la derecha se muestran diferentes muestras de funciones con este kernel.

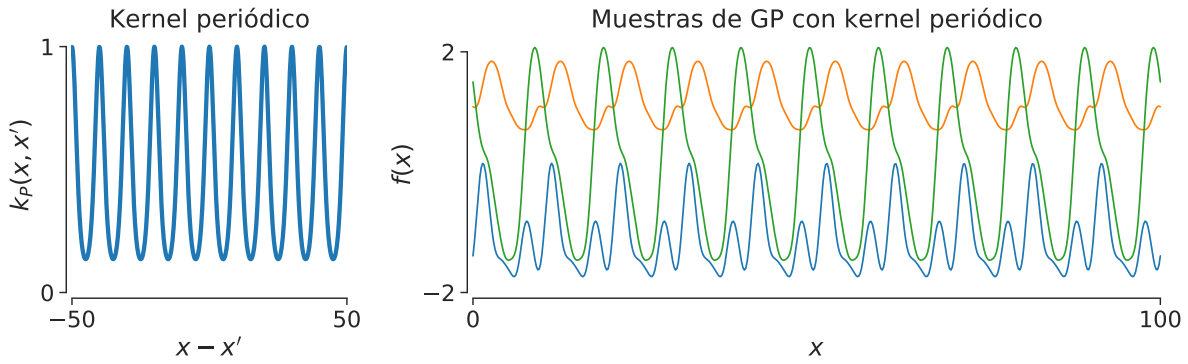


Fig. 41. Kernel periódico, en la izquierda se muestra la covarianza en función de su argumento $\tau = x - x'$, a la derecha de un $\mathcal{GP}$ usando un kernel periódico.

8.5.1. Operaciones con kernels

A la hora de diseñar un $\mathcal{GP}$ y elegir una función de covarianza, no se está completamente limitado a kernels conocidos, sino que también se pueden combinar para obtener distintas funciones de covarianza y representar mejor el proceso.

Tanto la suma y multiplicación de kernels da como resultado un kernel válido que puede ser utilizado, también la exponencial de un kernel es también es un kernel, es decir $\exp(k_1(\cdot, \cdot))$ con k_1 un kernel válido.

8.5.2. Representación espectral

Un teorema importante para las funciones de covarianza en procesos débilmente estaicionarios es el teorema de Wiener–Khinchin, el cual dice que si para un proceso débilmente estacionario existe una función de covarianza $k(\tau)$ finita y definida para cualquier $\tau = x - x'$, entonces existe una función $S(\xi)$ tal que:

$$k(\tau) = \int S(\xi) e^{2\pi i \xi \cdot \tau} d\xi, \quad S(\xi) = \int k(\tau) e^{-2\pi i \xi \cdot \tau} d\tau \tag{278}$$

Donde i es la unidad imaginaria. $S(\xi)$ es conocida como la densidad espectral de potencia (PSD), en otras palabras, la función de covarianza $k(\tau)$ y la PSD $S(\xi)$ son duales de Fourier el uno del otro. Esto es extremadamente útil a la hora de diseñar funciones de covarianza pues este puede ser llevado a cavo en el dominio de la frecuencia y luego llevado a una función de covarianza usando la transformada inversa de Fourier.

8.6. Extensiones para un $\mathcal{GP}$

A continuación veremos un número de extensiones que se le pueden hacer a un $\mathcal{GP}$ para abordar distintos tipos de problemas.

8.6.1. $\mathcal{GP}$ de clasificación

Hasta el momento hemos visto como usar un $\mathcal{GP}$ para regresión, pero este también puede usado para clasificación, para esto simplemente "pasamos" nuestro $\mathcal{GP}$ por una función logística, para así obtener un prior sobre $\sigma(f(x))$ donde σ es la función logística. Sin embargo esto trae consigo un problema, pues ahora la distribución posterior a las observaciones no se tiene de forma analítica como para el caso de regresión, esto lleva a que tengamos que recurrir a métodos aproximado de inferencia. Una solución simple es utilizar la aproximación de Laplace, pero si se quieren aproximaciones más fidedignas métodos más complejos pueden ser usados como *Expectation Maximization* y métodos MCMC. En la Fig.42 se muestra un ejemplo con datos sintéticos, a diferencia de la mayoría de los clasificadores vistos, este entrega naturalmente una densidad de probabilidad para la función de decisión.

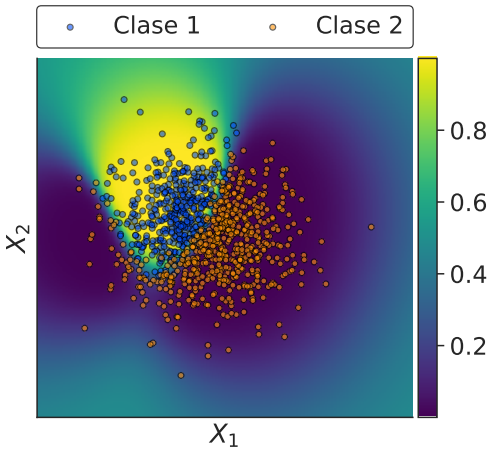


Fig. 42. $\mathcal{GP}$ de clasificación utilizando datos sintéticos. Este clasificador entrega una densidad de probabilidad en vez de una sola función de decisión.

8.6.2. Selección automática de relevancia (ARD) (*Selección automática de features*)

Un $\mathcal{GP}$ define una densidad de probabilidad sobre funciones, donde estas funciones son del tipo $f : \mathbb{R}^D \to \mathbb{R}$, con D es finito, este es nuestra dimensión de entrada o "características". Haciendo un pequeño cambio en nuestra función kernel podemos hacer que esta automáticamente seleccione las entradas más relevantes con el problema, es decir realice una selección de características automática.

Si tomamos el kernel de la Ec.(279) vemos que es una multiplicación de kernels SE, donde se tiene un *lenghtscale* por cada entrada ℓ_d, sabemos que mientras más grande es este ℓ_d menos flexible será el $\mathcal{GP}$ respecto a cambios en ese eje, haciendo que las funciones del proceso dependan cada vez menos de la componente d a medida que $\ell_d \to \infty$. De esta forma se puede controlar de forma automática la relevancia de cada eje del conjunto de entrada, pues los parámetros del kernel se obtienen en el entrenamiento. De esta forma estamos optimizando también en que grado afecta cada variable en nuestra predicción.

$$k(x, x') = \sigma^2 \exp\left(-\sum_{d=1}^{D} \frac{(x_d - x'_d)^2}{2\ell_d^2}\right) \tag{279}$$

8.6.3. Multi output $\mathcal{GP}$

Hasta el momento solo hemos hablado de $\mathcal{GP}$ cuando nuestro proceso es solo una dimensión de salida. Se pueden extender los procesos Gaussianos a funciones de más de una salida o canal, donde ahora la función de covarianza $k(x, x')$ no entrega un escalar sino una matriz definida positiva, donde la diagonal corresponde a la covarianza del canal o autocovarianza y los elementos fuera de la diagonal corresponden a las covarianzas cruzadas o cross-covarianza. Este tipo de procesos Gaussianos aumentan considerablemente de complejidad al diseñar funciones de covarianza.

Dado un número m de canales, se tendrán m funciones de autocovarianza y ahora $m(m-1)/2$ funciones de covarianza y $k(x, x')$ será una matriz de $m \times m$. El desafio está en diseñar o escoger estas funciones de tal forma que para cualquier par de puntos x, x' la matriz $k(x, x')$ sea definida positiva.

Una opción simple es asumir que los canales son independientes entre sí, lo que equivale a entrenar independientemente m procesos Gaussianos, uno para cada canal, esto facilita el diseño de las funciones de covarianza pero hace que se pierdan relaciones entre los canales.

8.7. Diferentes Interpretaciones de un $\mathcal{GP}$

8.7.1. De regresión lineal a $\mathcal{GP}$

Partiendo de la regresión lineal donde el modelo es:

$$y_i = wX_i + \epsilon_i \tag{280}$$

Donde $\epsilon_i \sim \mathcal{N}(0, \sigma_n^2)$, luego podíamos extender este modelo usando M funciones base ϕ_m y obtener:

$$y_i = \sum_{m=1}^{M} w_m \phi(x_i) + \epsilon_i \tag{281}$$

El paso siguiente es la regresión Bayesiana en base de funciones, donde agregamos un prior sobre los pesos $w_m \sim \mathcal{N}(0, \lambda_m^2)$, donde si obtenemos la covarianza de este proceso y marginalizamos por los pesos w_m obtenemos:

$$Cov(x, x') = k(x, x') = \sum_{m=1}^{M} \lambda_m^2 \phi_m(x)^T \phi_m(x') + \delta_{x,x'} \sigma_n^2 \tag{282}$$

Donde $\delta_{x,x'}$ el delta de Kronecker. Si nos damos cuenta esto define un $\mathcal{GP}$ con la función covarianza con un número finito de funciones bases. En general los $\mathcal{GP}$ corresponderán a covarianzas con infinitas funciones bases, como veremos a continuación.

Tomando un modelo sin ruido, con un número M de funciones base ϕ_m, donde sobre los pesos se define un prior i.i.d $w_m \sim \mathcal{N}(0, \sigma^2)$, tenemos:

$$f(x) = \sum_{m=1}^{M} w_m \phi(x) \tag{283}$$

Y tomando $\phi = \exp(-\frac{1}{2\ell^2}(x - c_i)^2)$ donde c_i son los centros de estas bases, y luego haciendo tender el número de funciones base M a infinito tenemos que la covarianza es:

$$\mathbb{E}\left\{f(x)f(x')\right\} = \sigma^2 \sum_{m=1}^{M} \phi_m(x)\phi_m(x') \qquad \text{tomando } M \to \infty \tag{284}$$

$$\mathbb{E}\left\{f(x)f(x')\right\} \to \sigma^2 \int e^{-\frac{1}{2\ell^2}(x-c)^2} e^{-\frac{1}{2\ell^2}(x'-c)^2} dc \tag{285}$$

$$= \sigma^2 \sqrt{\pi\ell^2} e^{-\frac{1}{4\ell^2}(x-x')^2} \tag{286}$$

$$= k_{SE}(x, x') \tag{287}$$

Donde vemos que efectivamente el kernel SE es una función de covarianza para una composición infinita de funciones base.

8.7.2. Nota sobre RKHS

Dado un conjunto de entrenamiento $(X, Y) = \{x_i, y_i\}_{i=1}^n$ condicionando el proceso solo a una muestra de test $X_* = x_*$ y asumiendo una función media nula, de la Ec.(263) para un $\mathcal{GP}$ la posterior de la media está dada por:

$$\bar{f}(x_*) = m_{x_*|X} = k(X, x_*)(k(X, X) + \sigma_n \mathbb{I})^{-1} Y \tag{288}$$

Y tomamos el vector $\boldsymbol{\alpha} = (k(X, X) + \sigma_n \mathbb{I})^{-1} Y$ obtenemos la expresión:

$$\bar{f}(x_*) = \sum_{i=1}^{n} \alpha_i k(x_i, x_*) \tag{289}$$

Donde, a pesar de que el proceso esta descrito por (posiblemente infinita) funciones base, aún así es la suma de finitos términos, cada uno centrado en un punto de entrenamiento, esto es debido al teorema del representante de los Espacios de Hilbert de Kernel Reproductor (RKHS). La intuición detrás de esto es que incluso si el $\mathcal{GP}$ induce una distribución conjunta sobre todos los $y = f(x)$, una para cada x en el dominio, al hacer predicciones en el punto x_* solo nos interesa la distribución $(n+1)$-dimensional definida por los puntos de entrenamiento más este punto de test.

Resumen

¿Que es un Proceso Gaussiano?

Un Proceso Gaussiano ($\mathcal{GP}$) es un método no paramétrico que define un *prior sobre funciones*, lo cual se traduce en una *posterior* sobre funciones si la verosimilitud es Gaussiana. Este proceso se encuentra caracterizado completamente por su función de media $m(\cdot)$ (generalmente se asume 0) y función de covarianza o kernel $K_\theta(\cdot,\cdot)$ que puede depender o no de parámetros (θ).

¿Cómo se entrena?

Dado un conjunto de entrenamiento $(X,Y) = \{(x_i, y_i)\}_{i=1}^n$, donde y_i pueden estar contaminadas o no por ruido (generalmente se asume ruido Gaussiano de varianza σ_n^2) se *minimiza* la log-verosimilitud marginal negativa (NLL), que para el caso de $m(\cdot) = 0$ es:

$$NLL = -\log \mathbb{P}(Y|X,\theta,\sigma_n) \tag{290}$$

$$NLL = \underbrace{\frac{1}{2}\log|\mathbf{K_y}|}_{\substack{\text{Penalización}\\ \text{por}\\ \text{complejidad}}} + \underbrace{\frac{1}{2}Y^T\mathbf{K}_y^{-1}Y}_{\text{Data fit}} + \underbrace{\frac{n}{2}\log 2\pi}_{\substack{\text{Constante de}\\ \text{normalización}}} \tag{291}$$

Donde $\mathbf{K}_y = K_\theta(X,X) + \sigma_n^2\mathbb{I}$.

¿Cómo se evalúa en nuevos puntos?

Si queremos evaluar en el conjunto de test X_*, la evaluación es:

$$f(X_*)|Y, X \sim \mathcal{N}(0, \Sigma_{X_*|X}) \tag{292}$$

Donde la media y covarianza son:

$$m_{X_*|X} = K(X_*,X)[K(X,X) + \sigma_n^2\mathbb{I}]^{-1}Y \tag{293}$$

$$\Sigma_{X_*|X} = K(X_*,X_*) - K(X_*,X)[K(X,X) + \sigma_n^2\mathbb{I}]^{-1}K(X,X_*) \tag{294}$$

¿Qué costo tiene?

- Entrenamiento: $\mathcal{O}(n^3)$ respecto al número de muestras del conjunto de **entrenamiento**.
- Evaluacion: $\mathcal{O}(n^2)$ respecto al número de muestras del conjunto de **test**.

Notas sobre la implementación:

Como la matriz de Gram ($K(X,X)$) es definida positiva y simétrica, para calcular su inversa de forma más eficiente se utiliza la descomposición de Cholesky, donde si $K = LL^T$, entonces $K^{-1} = (L^T)^{-1}L^{-1}$

9. Aprendizaje No Supervisado

9.1. Reducción de dimensionalidad

9.1.1. Análisis de componentes principales (ACP)

Consideremos un conjunto de observaciones de $\{\mathbf{x}_i\}_{i=1}^n \subset \mathbb{R}^N$, donde denotamos $\mathbf{x}_i = [x_{i1}, x_{i2}, \ldots, x_{iN}]^\top$. Podemos entender que el elemento x_{ij} corresponde al valor del atributo j para la observación i.

ejemplos

dar un ejemplo como el de nutrición, series de tiempo, geo-algo

Notemos que cada observación puede descomponerse en la base canónica $\{\mathbf{e}_i\}_{i=1}^N$ de $\mathbb{R}^N$ de la forma

$$\mathbf{x}_i = x_{i1}\mathbf{e}_1 + x_{i2}\mathbf{e}_2 + \cdots + x_{iN}\mathbf{e}_N \tag{295}$$

Notemos que es posible representar cada vector $\mathbf{x}_i$ mediante una cantidad $N' < N$ de términos truncando la representación anterior, es decir,

$$\mathbf{x}_i \approx \sum_{j=1}^{N'} x_{i\sigma(j)}\mathbf{e}_{\sigma(j)} \tag{296}$$

donde $\sigma : \{1, 2, \ldots, N\} \mapsto \{1, 2, \ldots, N\}$ es una permutación. Dicha aproximación de las observaciones $\{\mathbf{x}_i\}_{i=1}^n$ es una versión de baja dimensión, entonces, naturalmente nos podemos hacer la siguiente pregunta: dado una dimensión $N' < N$ ¿es efectivamente un subconjunto de los vectores canónicos la mejor base para descomponer as observaciones? ¿cómo encontramos la *mejor* base?

La respuesta a la primera pregunta es, en la mayoría de los casos, negativa. Esto es porque los elementos de la base canónica, por sí solos, conllevan poca información estructural que puede ser encontrada en los vectores observados. En particular, consideremos el caso en donde solo se dispone de dos observaciones $\{\mathbf{x}_1, \mathbf{x}_2\}$, si $N' = 2$, entonces una descomposición que garantiza error nulo es simplemente elegir $\mathbf{x}_1$ y $\mathbf{x}_2$ como bases de la nueva descomposición, donde los coeficientes estarían dados por $[1,\ 0]^\top$ y $[0,\ 1]^\top$.

Para encontrar la *mejor* base, lo primero que se requiere es definir qué se entiende por *mejor*. Nos enfocaremos en determinar una base cuyos componentes **ordenados** $\mathbf{c}_1, \mathbf{c}_2, \ldots$ capturan las N' direcciones (ortogonales) de máxima variabilidad de nuestros datos. Es decir, el primer elemento de la nueva base estará dado por

$$\mathbf{c}_1 = \underset{||\mathbf{c}_1||=1}{\arg\max} \langle \mathbf{c}_1, \mathbf{x} \rangle \tag{297}$$

Este criterio es conocido como **análisis de componentes principales (ACP)**. Notemos que la restricción $||\mathbf{c}_1|| = 1$ es porque recordemos que estamos la dirección de máxima varianza y la expresión anterior (cuadrática en $\mathbf{c}_1$) no es maximizable en $\mathbf{c}_1$. Por esta razón, sin pérdida de generalidad podemos fijar la norma de $\mathbf{c}_1$ en 1 y buscar una base *ortonormal*.

Asumiremos además que nuestros datos están estandarizados, es decir, tienen un vector de medias nulo y varianzas marginales unitarias. Esto es para reducir los sesgos por variables que pueden ser introducidos en la maximización de la expresión anterior: Si una dimensión tiene una varianza marginal mayor que el resto, ésta será más importante en la determinación de la dirección de máxima varianza solo por su magnitud y no por la relación entre variables.

Como en general no contamos con la distribución de las observaciones $p(\mathbf{x})$, podemos considerar una aproximación muestral de la varianza en la ecuación (297) y resolver

$$\mathbf{c}_1 = \underset{||\mathbf{c}_1||=1}{\arg\max} \sum_{i=1}^{N} \langle \mathbf{c}_1, \mathbf{x} \rangle^2. \tag{298}$$

Podemos ahora usar la siguiente notación

$$X = [\mathbf{x}_1, \mathbf{x}_2, \ldots, , \mathbf{x}_n] = \begin{bmatrix} x_{11} & \ldots & x_{n1} \\ \ldots & \ddots & \ldots \\ x_{1N} & \ldots & x_{nN} \end{bmatrix},$$

y reescribir la ecuación (298) como

$$\mathbf{c}_1 = \underset{||\mathbf{c}_1||=1}{\arg\max} ||\mathbf{c}_1^\top X||^2 = \underset{||\mathbf{c}_1||=1}{\arg\max}\, \mathbf{c}_1^\top X X^\top \mathbf{c}_1 = \underset{\mathbf{c}_1}{\arg\max}\, \frac{\mathbf{c}_1^\top X X^\top \mathbf{c}_1}{\mathbf{c}_1^\top \mathbf{c}_1}. \tag{299}$$

donde la última igualdad incorpora la normalización de $\mathbf{c}_1$ la cual permite formular el problema de optimización irrestricto. La expresión de la derecha se llama cuociente de Rayleigh para la matriz simétrica $XX^\top$, donde sabemos que la maximización de este cuociente está dado por el vector propio de la matriz $XX^\top$ asociado al mayor valor propio. Observe que $XX^\top$ es la matriz de correlación muestreal del conjunto de observaciones $\{\mathbf{x}_i\}_{i=1}^n$, consecuentemente, la proyección de una observación $\mathbf{x}_i$ en la dirección de máxima varianza, o bien la *primera componente principal*, está dada por

$$\mathbf{x}_i^{(1)} = \langle \mathbf{x}_i, \mathbf{c}_1 \rangle \tag{300}$$

donde $\mathbf{c}_1$ es el vector propio asociado al mayor valor propio de la matriz de covarianza muestreal $XX^\top$.

El cálculo de las siguientes componentes se realiza de forma iterativa sobre los residuos del conjunto de observaciones con respecto a las componentes anteriores. De esta forma, ACP encuentra una nueva base ortonormal tal que las componentes maximicen la variabilidad donde en algunos casos se puede perder intepretabilidad de las nuevas características generadas, pues son combinaciones lineales de las características originales de los datos. A pesar de esto, utilizando las primeras 2 o 3 componentes PCA se pueden visualizar datos de alta dimensionabilidad de forma ilustrativa.

9.1.2. Kernel PCA

El método Kernel PCA es similar a PCA, pero esta vez se utiliza el truco del kernel para proyectar los datos. En ese sentido, en vez de calcular la matriz de covarianza empírica, se utiliza la matriz de Gram.

$$K_{ij} = K(x_i, x_j) = \phi(x_i)\phi(x_j)^T$$

Luego, se prosigue de la misma forma que PCA linear.

En la figura 43 se puede observar un ejemplo en que el resultado de PCA linear no es suficiente, puesto que el problema es simétrico, mientras que KPCA realiza una correcta separación de ambos clusters.

9.1.3. Probabilistic PCA

PCA probabilístico (PPCA, por su sigla en inglés) tiene su inspiración en que PCA se puede expresar como la solución via máxima verosimilitud de un modelo probabilístico de variable latente. De este modo, PPCA propone un método iterativo para obtener la solución evaluando solo cierto número de componentes, sin necesidad de calcular la matriz de covarianza empírica.

El modelo probabilístico para PCA en que se inspira PPCA es el siguiente:

Sea $x_i \in \mathbb{R}^m$ los elementos observados, inputs o variables y $z \in \mathbb{R}^l$ una variable latente explícita correspondiente al espacio de las componentes principales. Se define un prior para z:

$$p(z) = N(0, I) \tag{301}$$

De este modo, la distribución condicional de x dado z también es Gaussiana:

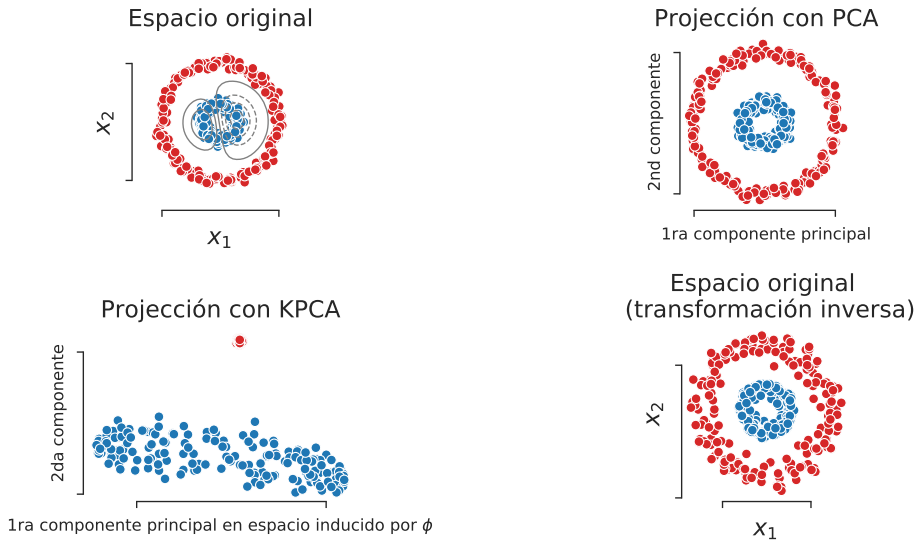


Fig. 43. Ejemplo en que kernel PCA sobre un conjunto de datos que no es linealmente separable.

$$p(x|z) = N(Wz + \mu, \sigma^2 I) \tag{302}$$

Donde $W \in \mathbb{R}^{M\times l}$ y $\mu \in \mathbb{R}^m$ son parámetros a determinar. Notemos que no se pierde generalidad tomar el prior para z con media cero y varianza unitaria, puesto que si se toma otro prior más general, se produce el mismo modelo.

Dado que tenemos modelo paramétrico probabilístico, podemos estimar los parámetros con máxima verosimilitud. Dado los datos $X = \{x_i\}_{i=1}^n$. La log-verosimilitud está dada por:

$$\log p(X|W, \mu, \sigma^2) = \sum_{i=1}^{n} \log p(x_i|W, \mu, \sigma^2) \tag{303}$$

$$= \frac{nl}{2}\log(2\pi) - \frac{n}{2}log|C| - \frac{1}{2}\sum_{i=1}^{n}(x_i - \mu)^T C^{-1}(x_i - \mu), \tag{304}$$

con $C = WW^T + \sigma^2 I$.
Usando la condición de primer orden obtenemos

$$\mu = \bar{x} = \frac{1}{n}\sum_{i=1}^{n} x_i. \tag{305}$$

De esta manera tenemos la función de log-verosimilitud completa:

$$\log p(X, Z|W, \mu, \sigma^2) = \sum_{i=1}^{n}\{\log p(x_i|z_i) + \log p(z_i)\} \tag{306}$$

y evaluando en $\mu = \bar{x}$

$$\mathbb{E}[p(X,Z|W,\mu,\sigma^2)] = -\sum_{i=1}^{n}\Bigg\{\frac{l}{2}\log(2\pi\sigma^2) + \frac{1}{2}\mathrm{Tr}(\mathbb{E}[z_i z_i^T]) \frac{1}{2\sigma^2}||x_i-\mu||^2 - \frac{1}{\sigma^2}\mathbb{E}[z_i]^T W^T(x_i-\mu) \frac{1}{2\sigma^2}\mathrm{Tr}(\mathbb{E}[z_i z_i^T]W^T W)\Bigg\} \tag{307}$$

10. Clustering

10.1. k-means

Dado un entero $k \in \mathbb{N}$ y uin conjunto de observaciones $X = \{x_i\}_{i=1}$ con $x_i \in \mathbb{R}^D$ queremos separar los datos en k graupos, donde cada grupo se le asigna un centroide μ_k y cada elemento x_i se le asigna el grupo que tenga el centroide más cercano.

Sea r_{ik} la asignación, esta estará definida por:

$$r_{ik} = \begin{cases} 1 & \text{si } k = \text{argmin}||x_i - x_k|| \\ 0 & \text{si no.} \end{cases}$$

Es decir, para encontrar los centroides se debe minimizar la función:

$$J = \sum_{i=1}^{N}\sum_{k=1}^{K} r_{ik}||x_i - x_j||^2 \tag{308}$$

Para minimizar esta función utilizaremos un enfoque llamado *Expectation-Maximization*. Este es un método iterativo y como tal, tiene problemas con mínimo locales, pero para solucionar esto, basta inicializar el algoritmo muchas veces.

El algoritmo está dado por:

- **E-step:** En este paso, se calculan (actualizan) las asignaciones r_{ik}, dejando fijos μ_k. Lo que corresponde a asignar el dato x_i al centroide más cercano.
- **M-step:** El siguiente paso corresponde a actualizar los centroides μ_k dejando fijo las asignaciones r_{ik}.

  Como J es cuadrática en μ_k, entonces podemos utilizar la condición de primer orden:

  $$\mu_k = \frac{\sum_{i=1}^{N} r_{ik}x_i}{\sum_{i=1}^{N} r_{ik}} \tag{309}$$

  Lo que corresponde a asignar el centro del cluster al promedio de todas las muestras asignadas al antiguo cluster.

Ejemplo: En la figura 44 se observa un ejemplo de clustering utilizando kmeans. Como se puede notar, los clusters creados por kmeans son circulares, puesto que se utiliza distancia euclediana hace el centro del cluster.

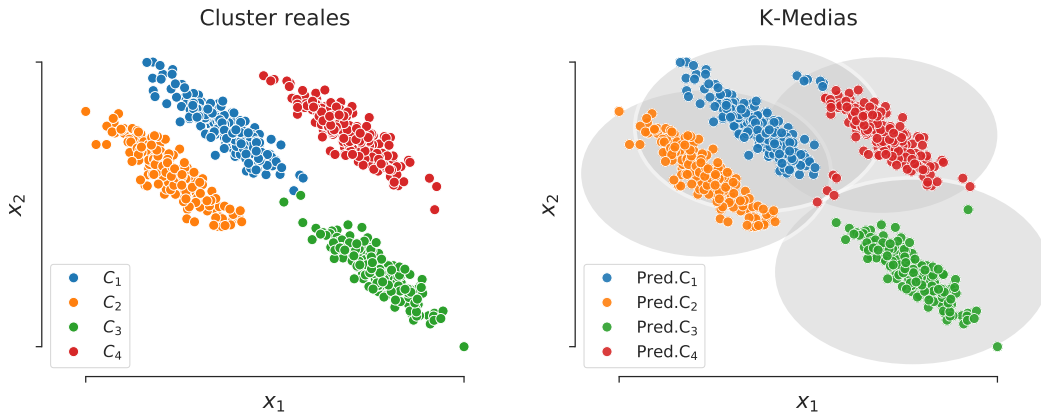


Fig. 44. (Izquierda) Datos reales con sus etiquetas correctas. (Derecha) Clusters encontrados por k-means.

10.2. Modelo de mezcla de gaussianas

La mezcla de gaussianas (GMM, por su sigla en inglés) es un caso general de k-means, en donde los clusters pueden tener una forma anisotrópica modelada por una Gaussiana con covarianza no necesariamente proporcional a la identidad. Además, si bien su solución puede ser muy similar a K-means los supuestos subyacentes al modelo CMM son diferentes y obedecen a un enfoque de modelo generativo.

Recordemos que una distribución de mezcla de gaussianas dada por

$$p(\mathbf{x}) = \sum_{k=1}^{K} \pi_k \mathcal{N}(\xi|\mu_k, \Sigma_k) \tag{310}$$

es un modelo para aproximar distribuciones mediante la consideración de varias componentes $K \in \mathbb{N}$. Nos referiremos a los parámetros de este modelo como

- π_k : coeficiente de mezcla del cluster k
- μ_k : media del cluster k
- Σ_k : matriz de covarianza del cluster k

En el caso de clustering, es necesario además saber de qué componente fue generada cada muestra. Con el mismo criterio que K-means, podemos asignar una variable latente, denotada $\mathbf{z} = [z_1, z_2, \ldots, z_K]^\top \in \{0,1\}^K$, que describe la asignación de cada muestra a cada cluster, es decir, $z_k = 1$ si la observación x_n pertenece al cluster k, y $z_k = 0$ si no.

Podemos denotar la distribución marginal del cluster k, es decir, de que una observación sea generada por la k-ésima componente como

$$p(z_k = 1) = \pi_k \tag{311}$$

con lo que la distribución marginal del vector $\mathbf{z}$ puede expresarse como

$$p(\mathbf{z}) = \prod_{k=1}^{K} \pi_k^{z_k} \tag{312}$$

pues recordemos que $\sum_{k=1}^{K} z_k = 1$ y $\sum_{k=1}^{K} \pi_k = 1$.

Adicionalmente, podemos describir la distribución condicional de $\mathbf{x}$, dado que su componente es la k-ésima, i.e., $z_k = 1$, como

$$p(\mathbf{x}|z_k = 1) = \mathcal{N}(\mathbf{x}|\mu_k, \Sigma_k) \tag{313}$$

y consecuente su distribución condicional como

$$p(\mathbf{x}|\mathbf{z}) = (\mathcal{N}(\mathbf{x}|\mu_k, \Sigma_k))^{z_k} \tag{314}$$

Multiplicando las ecuaciones (312) y (314), podemos obtener la distribución conjunta de $\mathbf{x}$ y $\mathbf{z}$, donde luego sumando c.r.a. k, recuperamos la distribución marginal de $\mathbf{x}$ de suma de gaussianas mediante el uso de la variable latente $\mathbf{z}$. Respectivamente:

$$p(\mathbf{x}, \mathbf{z}) = (\pi_k \mathcal{N}(\mathbf{x}|\mu_k, \Sigma_k))^{z_k} \tag{315}$$

$$p(\mathbf{x}) = \sum_{k=1}^{K} p(\mathbf{x}, \mathbf{z}) = \sum_{k=1}^{K} \pi_k \mathcal{N}(\mathbf{x}|\mu_k, \Sigma_k) \tag{316}$$

discusión

¿Por qué la conjunta es *mejor* que la marginal?
Como generar muestras: ancestral sampling

10.2.1. Posterior

Veamos ahora la forma que toma la distribución posterior de la variable latente z, usando el teorema de Bayes, tenemos

$$p(z_k = 1|\mathbf{x}) = \frac{p(\mathbf{x}|z_k = 1)p(z_k = 1)}{p(\mathbf{x})} = \frac{\pi_k \mathcal{N}(\mathbf{x}|\mu_k, \Sigma_k)}{\sum_{k=1}^{K} \pi_k \mathcal{N}(\mathbf{x}|\mu_k, \Sigma_k)} \tag{317}$$

esta cantidad también se llama *responsabilidad* de la componente k por la observación $\mathbf{x}$.

10.2.2. Máxima verosimilitud

Recordemos que los parámetros del modelo de mezcla de gaussianas son los coeficientes de mezcla, las medias y varianzas de cada componente. Si disponemos de un conjunto de observaciones $\{\mathbf{x}_n\}_{n=1}^{N}$, entonces podemos encontrar dichos parámetros mediante máxima verosimilitud. Observe que la log-verosimilitud está dada por

$$\log p(\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_N) = \sum_{n=1}^{N} \log p(\mathbf{x}_n) = \sum_{n=1}^{N} \log \sum_{k=1}^{K} \pi_k \mathcal{N}(\mathbf{x}_n|\mu_k, \Sigma_k) \tag{318}$$

Observemos que hay una serie de complicaciones relacionados a la búsqueda de estos parámetros mediante la maximización de las ecuación (318). Primero, están las soluciones dadas por singularidades cuando una muestra es exactamente igual a una de las medias, en cuyo caso un término de la log-verosimilitud es proporcional a $1/\sigma_k$, con lo que la maximización te σ_k resulta en una log-verosimilitud infinita. En segundo lugar tenemos las redundancias de soluciones: para cada máximo local (o solución en general) de la log-verosimilitd existen $K!$ soluciones equivalente con la misma verosimilitud dadas por las permutaciones de las etiquetas de los clusters. Finalmente, optimizar la log-verosimilitud de la mezcla de gaussianas es desafiante porque la sumatoria aparece *dentro* del logaritmo, consecuentemente, el logaritmo no actúa directamente en la gaussiana reduciendo el funcional de optimización a una solución en forma cerrada. Por esta razón, es necesario considerar basados en gradiente.

discusión

¿Por qué no hay *singularidades* en el caso de una gaussiana?

10.2.3. Entrenamiento

Veamos las condiciones de primer orden para encontrar los parámetros, es decir, igualando el gradiente (con respecto a cada uno de los parámetros) de la log-verosimilitud igual a cero. Denotando $\gamma(z_{nk}) = p(z_{nk} = 1|\mathbf{x}_n)$, obtenemos

$$\mu_k = \frac{1}{R_k} \sum_{n=1}^{N} \gamma(z_{nk}) \mathbf{x}_i \tag{319}$$

$$\Sigma_k = \frac{1}{R_k} \sum_{n=1}^{N} \gamma(z_{nk})(\mathbf{x}_n - \mu_k)(\mathbf{x}_n - \mu_k)^\top \tag{320}$$

$$\pi_k = \frac{R_k}{R}, \tag{321}$$

donde

$$R_k = \sum_{i=n}^{N} \gamma(z_{nk}) \qquad \text{y} \qquad R = \sum_{k=1}^{K} R_k.$$

Observe que esto no constituye una solución en forma cerrada para los parámetros, pues la posterior $\gamma(z_{nk})$ depende de todos los parámetros de acuerdo a la ecuación (317). Sin embargo, podemos considerar un procedimiento iterativo en donde calculamos las posteriores $\gamma(z_{nk})$ (llamado paso E), para luego calcular los parámetros óptimos de acuerdo a las ecuaciones anteriores (llamado paso M).

Ejemplo: La figura 45 muestra un ejemplo de clustering utilizando GMM. En este caso, se puede observar directamente como GMM es una generalización de kmeans, en donde ahora los clusters tienen forma de gaussiana anisotrópica.

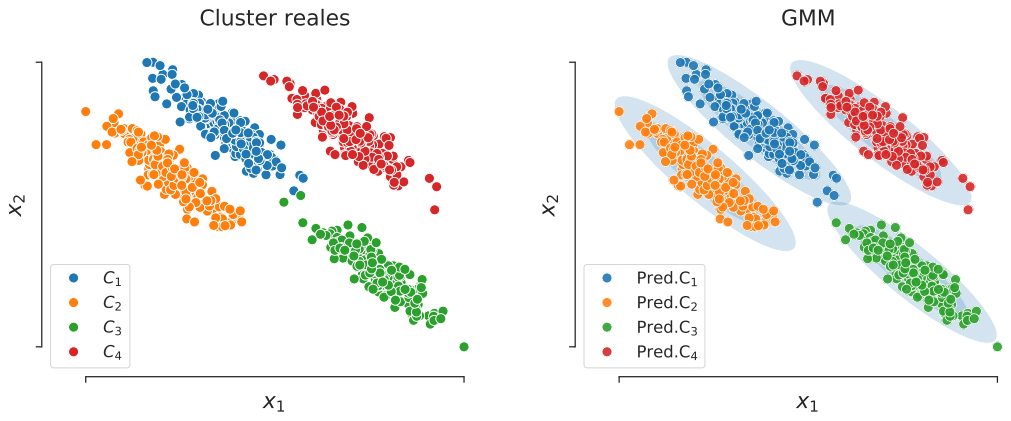


Fig. 45. (Izquierda) Datos reales con sus etiquetas correctas. (Derecha) Clusters encontrados por GMM.

10.2.4. Expectation-Maximisation

En el caso general, donde tenemos *observables* $\mathbf{x}$, variables latentes $\mathbf{z}$ y parámetros θ, y un modelo descrito por una distribución marginal que puede ser expresada mediante

$$\log p(\mathbf{x}|\theta) = \log \int p(\mathbf{x}, \mathbf{z}|\theta) d\mathbf{z} \tag{322}$$

donde tenemos logaritmos de sumas, lo cual es complicado de optimizar, incluso cuando la distribución conjunta $p(\mathbf{x}, \mathbf{z}|\theta)$ está en la familia exponencial.

Asumamos por un momento que tenemos valores para la variable latente $\mathbf{z}$, si este fuese el caso, podríamos buscar los parámetros mediante la optimización de la log-verosimilitud completa, $\log p(\mathbf{x}, \mathbf{z}|\theta)$, la cual como en GMM puede tener una forma más simple de optimizar debido a que no hay una suma dentro del logaritmo.

En la práctica, sin embargo, no tenemos acceso al valor de las variables latentes $\mathbf{z}_n$ sino que únicamente podemos acceder a una estimación de éstas mediante la distribución posterior $p(\mathbf{z}|\mathbf{x}, \theta)$. Entonces, estrictamente hablado, la cantidad que nos gustaría maximizar $\log p(\mathbf{x}, \mathbf{z}|\theta)$ es aleatoria, por lo que podemos maximizar su esperanza con respecto a las observaciones (etapa de Maximización). Luego, con los valores obtenidos para los parámetros podemos recalcular la distribución posterior $p(\mathbf{z}|\mathbf{x}, \theta)$ (etapa de Esperanza) y seguir este proceso iterativamente. La motivación de este procedimiento, llamado *Expectation-Maximisation*, es que en primer lugar, con la cantidad $p(\mathbf{z}|\mathbf{x}, \theta)$ fija, el cálculo de los nuevos parámetros mediante máxima verosimilitud indiscutiblemente aumenta la verosimilitud, luego éstos mejores parámetros dan consecuentemente una mejor estimación de la distribución posterior $p(\mathbf{z}|\mathbf{x}, \theta)$, con lo cual la actualización de los parámetros usando esta mejorada aproximación de la posterior debe ser incluso mejor.

discusión

Bayesian GMM: ¿cómo elegir la cantidad de Gaussianas?

10.3. Density-based spatial clustering of applications with noise (DBSCAN)

Es un algoritmo de clustering propuesto por Martin Ester et al. el cual ha tenido mucha popularidad puesto que no requiere definir una cantidad inicial de número de clusters. Los hiper-parámetros de entrada del modelo son 2:

- Mínimo número de punto $minPts$.
- Radio o vecindad ϵ.

El algoritmo se basa en la idea de que dado 2 púntos x_i, x_j dentro de un mismo cluster, se dice que x_i *es alcanzable por* x_j, si siempre se puede llegar de x_i a x_j avanzando de punto en punto, donde la distancia entre cada uno es al menos menor que ϵ y además un punto intermedio es un punto núcleo. Con estos el algoritmo define tres tipos de puntos:

- **Puntos núcleo:** Son puntos x_i tales que en una vecindad ϵ tienen almenos $minPts$ vecinos.
- **Puntos borde:** Constituyen el *borde externo* de los cluster.
- **Outliers:** Puntos que no son alcanzables por ningún punto.

Notemos que con lo anterior, se desprende que todo cluster debe tener al menos un punto núcleo. El algoritmo para encontrar los clusters es el siguiente:

Algoritmo 3 Pseudo código de DBSCAN

```
1: function DBSCAN(D, eps, MinPts)
2:     C ← 0
3:     for cada punto P no visitado en D do
4:         marcar P como visitado
5:         if sizeOf(PuntosVecinos) ≤ MinPts then
6:             marcar P como RUIDO
7:         else
8:             C ← C+1
9:             expandirCluster(P,vecinos, C, eps, MinPts)
```

Algoritmo 4 Función para expandir cluster.

```
1: function expandirCluster(P, vecinosPts, C, eps, MinPts)
2:     agregar P al cluster C
3:     for cada punto P' en vecinosPts do
4:         if P' no fue visitado then
5:             marcar P' como visitado
6:             vecinosPts' ← regionDeConsulta(P', eps)
7:             if sizeof(vecinosPts') ≥ MinPts then
8:                 vecinosPts ← vecinosPts ∪ vecinosPts'
9:         if P' no tiene cluster asignado then
10:            P' se le asigna el cluster C
```

Algoritmo 5 Retorna los puntos de la vecindad de búsqueda para un punto.

```
1: function regionDeConsulta(P, eps)
       return Todos los puntos junto a P' que están a eps de distancia (incluyendo P)
```

Ejemplo: La figura 46 muestra un ejemplo de clustering utilizando DBSCAN. La figura 46(derecha) muestra en negro los puntos que son clasificados como ruido o *outliers* por el algoritmo. Por otro lado, los puntos núcleos son graficados como un punto grande, mientras que los puntos borde se grafican con un marcador pequeño.

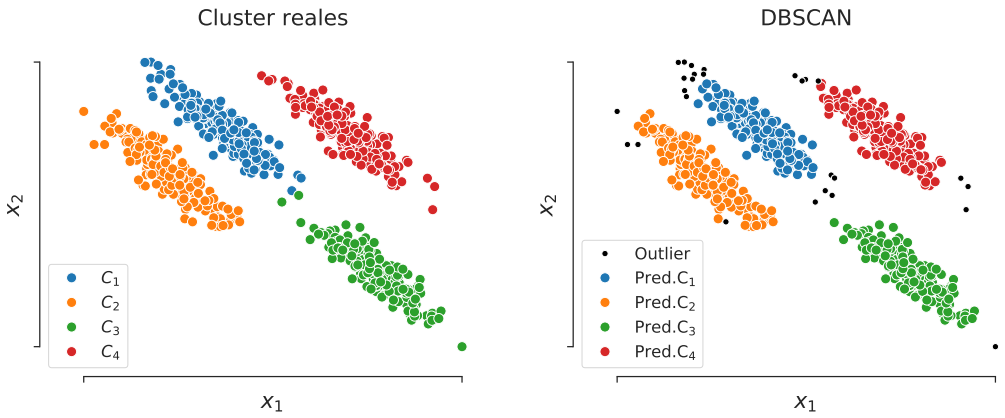


Fig. 46. (Izquierda) Datos reales con sus etiquetas correctas. (Derecha) Clusters encontrados por DBSCAN.

Referencias

Bayes, T. (1763). An essay towards solving a problem in the doctrine of chances. *Phil. Trans. R. Soc.*, *53*, 370–418. Descargado de `https://doi.org/10.1214/13-STS438` doi: 10.1098/rstl.1763.0053

Bellhouse, D. R. (2004). The reverend thomas bayes, frs: A biography to celebrate the tercentenary of his birth. *Statistical Science*, *19*(1), 3–32.

Bengio, Y. (2009). Learning deep architectures for ai. *Foundations and Trends® in Machine Learning*, *2*(1), 1-127.

Bengio, Y. (2016). *What's yoshua bengio's opinion on max welling's position paper äre ml and statistics complementary?̈* Descargado de `https://www.quora.com/Whats-Yoshua-Bengios-opinion-on-Max-Wellings-position-paper-Are-ML-and-Statistics-Complementary`

Ben-Israel, A., y Greville, T. (2006). *Generalized inverses: Theory and applications*. Springer New York.

Boser, B. E., Guyon, I. M., y Vapnik, V. N. (1992). A training algorithm for optimal margin classifiers. En *Proceedings of the fifth annual workshop on computational learning theory* (pp. 144–).

Bostrom, N. (2014). *Superintelligence: Paths, dangers, strategies*. Oxford University Press.

Breiman, L., Friedman, J. H., Olshen, R. A., y Stone, C. J. (1984). *Classification and regression trees*. Wadsworth & Brooks.

Davenport, T. H., y Patil, D. (2012). *Data scientist: The sexiest job of the 21st century.* Descargado de `https://hbr.org/2012/10/data-scientist-the-sexiest-job-of-the-21st-century`

Deng, J., Dong, W., Socher, R., Li, L.-J., Li, K., y Fei-Fei, L. (2009). ImageNet: A Large-Scale Hierarchical Image Database. En *Cvpr09.*

Farley, B., y Clark, W. (1954). Simulation of self-organizing systems by digital computer. *IRE Transactions on Information Theory*, *4*(4), 76-84.

Fayyad, U., Piatetsky-Shapiro, G., y Smyth, P. (1996). From data mining to knowledge discovery in databases. *AI Magazine*, *17*(3).

Ferguson, T. S. (1973). A bayesian analysis of some nonparametric problems. *Annals of Statistics*, *1*(2), 209– 230.

Fukushima, K. (1980). Neocognitron: A self-organizing neural network model for a mechanism of pattern recognition unaffected by shift in position. *Biol. Cybern.*, *36*, 193–202.

Gal, Y. (2015). *The science of deep learning.* Descargado de `YarinGals'swebsite:http://mlg.eng.cam.ac.uk/yarin/blog\_5058.html`

Ghahramani, Z. (2015). Probabilistic machine learning and artificial intelligence. *Nature*, *521*, 452–459.

Harari, Y. N. (2015). *Sapiens: A brief history of humankind*. Harper.

Hjort, N., Holmes, C., Müller, P., y Walker, S. (2010). *Bayesian nonparametrics*. Cambridge University Press.

Hoerl, A. E., y Kennard, R. W. (1970). Ridge regression: Biased estimation for nonorthogonal problems. *Technometrics*, *12*(1), 55-67.

Hopfield, J. J. (1982). Neural networks and physical systems with emergent collective computational abilities. *Proceedings of the National Academy of Sciences of the USA*, *79*, 2554–2558.

James, G., Witten, D., Hastie, T., y Tibshirani, R. (2014). *An introduction to statistical learning: With applications in r*. Springer.

Jordan, M. I., Ghahramani, Z., Jaakkola, T., y Saul, L. (1999). An introduction to variational methods for graphical models. *Machine Learning*, *37*, 183-233.

LeCun, Y., Boser, B., Denker, J. S., Henderson, D., Howard, R. E., Hubbard, W., y Jackel, L. D. (1989). Backpropagation applied to handwritten zip code recognition. *Neural Computation*, *1*(4), 541-551.

Lighthill, J. (1973). Artificial intelligence: A general survey. *Artificial Intelligence: a paper symposium*.

Minsky, M. (1952). *A neural-analogue calculator based upon a probability model of reinforcement* (Inf. Téc.). Harvard University Psychological Laboratories.

Minsky, M., y Papert, S. (1969). *Perceptrons: an introduction to computational geometry*. MIT.

Neal, R. M. (1993). *Probabilistic inference using markov chain monte carlo methods* (Inf. Téc.). University of Toronto, Department of Computer Science.

Pierce, G. (1949). *The song of insects.* Harvard College Press.

Rasmussen, C. E., y Williams, C. K. (2006). *Gaussian processes for machine learning.* The MIT Press.

Rosenblatt, F. (1958). The perceptron: A probabilistic model for information storage and organization in the brain. *Psychological Review*, *65*(6), 386–408.

Rumelhart, D. E., Hinton, G. E., y Williams, R. J. (1986). Learning representations by back-propagating errors. *Nature*, *23*(6008), 533-536.

Russell, S. J., y Norvig, P. (2009). *Artificial intelligence: A modern approach* (3.ª ed.). Pearson Education.

Salakhutdinov, G. E. (2006). Reducing the dimensionality of data with neural networks. *Science*, *313*, 504-507.

Shannon, C. (1950). Programming a computer for playing chess. *Philosophical Magazine*, *41*(324).

Silver, D., Huang, A., Maddison, C. J., Guez, A., Sifre, L., van den Driessche, G., ... Hassabis, D. (2016). Mastering the game of Go with deep neural networks and tree search. *Nature*, *529*, 484–503. Descargado de `http://www.nature.com/nature/journal/v529/n7587/full/nature16961.html`

Stigler, S. M. (2013, 08). The true title of bayes's essay. *Statist. Sci.*, *28*(3), 283–288. Descargado de `https://doi.org/10.1214/13-STS438` doi: 10.1214/13-STS438

Tibshirani, R. (1996). Regression shrinkage and selection via the Lasso. *Journal of the Royal Statistical Society: Series B (Methodological)*, *58*(1), 267-288.

Tikhonov, A. N., y Arsenin, V. Y. (1977). *Solution of ill-posed problems.* Washington: Winston & Sons.

Turing, A. (1950). Computing intelligence and machinery. *Mind*, *59*(236), 433-460.

Vapnik, V., y Chervonenkis, A. (1971). On the uniform convergence of relative frequencies of events to their probabilities. *Theory of Probability and its Applications*, *16*, 264-280.

Welling, M. (2015). Are ml and statistics complementary? *Roundtable discussion at the 6th IMSISBA meeting on " Data Science in the next 50 years".*

Werbos, P. (1974). *Beyond regression: New tools for prediction and analysis in the behavioral sciences* (Tesis Doctoral no publicada). Harvard University.