UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO

PROGRAMA DE MAESTRÍA Y DOCTORADO EN CIENCIAS MATEMÁTICAS Y DE LA ESPECIALIZACIÓN EN ESTADÍSTICA APLICADA.

REDES NEURONALES BAYESIANAS: UNA MIRADA AL APRENDIZAJE ESTADÍSTICO EN EL PROBLEMA DE CLASIFICACIÓN

TESINA
QUE PARA OPTAR POR EL GRADO DE:
MAESTRO EN CIENCIAS

PRESENTA:
ANDRÉS ASTORGA ESPRIELLA

DIRECTORA DE LA TESINA
DRA. RUTH SELENE FUENTES GARCÍA, FACULTAD DE CIENCIAS

MÉXICO, D. F. 14 DE MAYO DEL 2015.

Redes Neuronales Bayesianas: una mirada al aprendizaje estadístico en el problema de clasificación

Andrés Astorga Espriella
Universidad Nacional Autónoma de México

1. Introducción

El *Aprendizaje estadístico* es una rama de la ciencia que nace del llamado *aprendizaje de máquina* y alude a una gran variedad de métodos estdísticos para utilizar y comprender la información obtenida a través de grandes cantidades de datos e información. En términos matemáticos, suponga que observamos una variable de respuesta cuantitativa Y, y un número p de predictores distintos $X_1, ..., X_p$ y asumimos que existe una relación de la respuesta con los predictores $X := (X_1, ..., X_p)$, la cual puede estar escrita de manera general por:

$$Y = f(X) + \epsilon. \tag{1.1}$$

En la expresión anterior, f es una función desconocida de los datos y ϵ es un número aleatorio que representa el error de la aproximación, el cual tienen media cero y es independiente de X. Y puede ser un vector m dimensional. En escencia, el *aprendizaje estadístico* se refiere a un conjunto de módelos y métodos que nos permiten estimar a f. Aunque el término "aprendizaje estadístico" es reciente, los conceptos y métodos en los que se basa este campo han sido desarrollados hace tiempo; a principios del siglo diecinueve, Legendre y Gauss publicaron trabajos relacionados a lo qué conocemos como *regresión lineal*, y en 1936 ya se conocía el concepto de *análisis del discriminante lineal* dado por Fisher. Y dos años después, varios autores propusieron otro tipo de alternativas como la *regresión logística*. Para finales de 1970, muchas técnicas de aprendizaje fueron desarrolladas; sin embargo, eran exclusivamente métodos lineales, pues ajustar de forma *no lineal* la información era casi imposible con el poder computacional de aquellos días. En la actualidad, el poder computacional es superior y ajustar modelos no lineales a los datos se ha vuelto más accesible. No obstante, si buscamos conseguir predicciones e inferencias exactas de problemas complejos, necesitamos ajustar los módelos con cantidades enormes de datos que incluso con la tecnología de ahora no pueden ser manejados eficientemente. El aprendizaje estadístico (como el aprendizaje de máquina) tienen como objetivo de encontrar estiamciones exactas de f utilizando grandes cantidades de información.

Para estimar la función f, se utiliza una muestra de nuestros predictores $S := \{x_1, ...x_N\}$ donde N es suficinetemente grande. A diferencia de la estadística tradicional, en donde se busca estimar la función f utilizando toda la información S, el aprendiaje estadístico toma un subconjunto S_0 de S de tamaño significativamenete menor, a cual se le llama *el conjunto de entrenamiento*. Haciendo uso de este conjunto S_0, se estima a f con algún método estadístico, para después corroborar la efectividad del método con el subconjunto $S_1 := S \setminus S_0$, llamado el *conjunto de prueba*. Es por esa razón que uno utliza el término "aprendizaje" cuando se refiere a este tipo de ejemplos pues simula de cierta forma la manera en la que aprendemos; cuando somos pequeños y queremos diferenciar a un perro de un gato por ejemplo. No necesitamos una muestra de 100 gatos y 90 perros para poder clasificarlos, sólo necesitamos un par de cada uno y un buen proceso cognitivo . Se dice que el aprendizaje es supervisado cuando nuestra muestra

S viene incluida con la respuesta y_i de (1.1) para cada observación, y el método estadástico utiliza esa información para la estimación de f. En otro caso, se dice que el *aprendizaje es no supervisado*.

Al estimar f se busca que esta estimación sea lo suficientemente "flexible" y ajuste de manera correcta los datos, con el fin de obtener predicciones precisas de nuestra variable Y; sin embargo, si sólo buscamos que ajuste todos los datos de entrenamiento, el módelo (1.1) pierde interpretabilidad, pues si queremos repetir el método con una muestra distinta, obtendríamos una función f diferente. A éste fenómeno se le conoce como *sobreajuste*. Esta disyuntiva entre flexibilidad y sobreajuste se le llama *compensación entre varianza-sesgo*. La *varianza* mide que tanto cambia la funcón f cuando estimamos con distintos conjuntos de entremiento, mientras que el *sesgo* se refiere al error que se introduce en la estimación al tratar de aproximar un problema de la "vida real". Por ejemplo, si utilizamos regresión lineal, esperamos obtener un sesgo muy alto pues no existe un problema en la vida real que entre las variables se comporte linealmente. El objetivo del aprendizaje de máquina es obtener un estimador f cuya varianza y sesgo sean pequeños para todo punto.

1.1. El problema de Clasificación

En muchas situaciones, la variable respuesta Y de nuestro modelo (1.1) es una variable *cualitativa* o *categórica* en lugar de cuantitativa. Tomemos por ejemplo, el color de los ojos de una persona. En este caso Y puede tomar los valores café, azul o verde. Predecir una variable cualitativa por medio de observaciones puede ser referido como el problema de *Clasificación*, pues trata de asignar categorias o clases $\mathcal{C}_1, ..., \mathcal{C}_k$ a nuevas observaciones.

Problemas de clasificación ocurren recurrentemente:

- Una persona entra a una sala de emergencia con una serie de síntomas que están relacionados a tres tipos de enfermedades. ¿Con cuál de las tres condiciones vamos a diagnosticar al paciente?
- Un empresa bancaria quiere saber si las transacciones de uno de sus usuarios es fraudulenta o no basándose en el historial de compras de la persona.
- A un laboratorista le gustaría conocer qué mutaciones del ADN son una posible causa de una enfermedad y cuales no, de un número de pacientes que presenten o no este padecimiento.

Ingenuamente puede intentarse estimar f de la expresión (1.1) en el problema de clasificación usando la regresión lineal, pero eso no lleva a resultados satisfactorios en este contexto debido a: Supongamos que nuestra muestra son vectores p dimensionales; es decir tenemos p predictores en nuestros datos. Entonces, si queremos aplicar regresión lineal en este problema, estamos suponiendo que la varible de respuesta Y satisface:

$$Y := \beta_0 + \beta^T X.$$

Posteriormente, ajustaríamos a partir de nuestras observaciones los parámetros β_0, β, donde β posiblemente sea un vector multidimensional, y haciendo uso del método de mínimos cuadrados. El problema de este procedimiento es que si suponemos que la variable Y tiene un carácter cualitativo, se presentan demasiadas ambiguedades; por ejemplo, en el problema de diagnosticar una conmoción física en un paciente, Y puede ser de la forma:

$$Y = \begin{cases} 1 & \text{si} \quad \text{Epilepsia} \\ 2 & \text{si} \quad \text{Sobredosis} \\ 3 & \text{si} \quad \text{Asfixia} \end{cases}$$

el resultado con esta Y va a ser completamente distinto al caso en el que Y fuera:

$$Y = \begin{cases} 1 & \text{si} \quad \text{Asfixia} \\ 2 & \text{si} \quad \text{Epilepsia} \\ 3 & \text{si} \quad \text{Sobredosis} \end{cases}$$

pues la asigmnamos un peso distinto a cada clase, además de que la solución por medio de minimos cuadrados nos daría soluciones cuyos valores son números reales, por lo que tenemos que encontrar un forma eficiente de interpretar dichos valores.

En el caso de dos clases $\mathcal{C}_0, \mathcal{C}_1$, en vez de tratar de usar una regresión de la forma (1.1), podemos mejor pensar que nuestro modelo es:

$$P(Y = 0|X = x) = \beta_0 + \beta^T x, \tag{1.2}$$

es decir, en vez de tratar de dar una predicción de una variable cualitativa, trataremos de predecir la probabilidad de que un valor caiga en la clase $\mathcal{C}_0$. El problema de esta interpretación es que la estimación por medio del método de mínimos cuadrados difícilmente nos va a dar una respuesta que este en el intervalo $[0, 1]$. En la siguiente sección veremos métodos estadísticos que atacan eficientemente este problema: *regresión logística*, y *análisis del discriminante*, los cuales generan fronteras lineales que separan a cada clase. Existe otro método que no está relacionado con estadística conocido como el *perceptron*, el cual también genera fronteras lineales de decisión y que inspiró uno de los métodos más celebrados en el aprendizaje de máquina: *las redes neuronales*.

En el capítulo tres explicaremos lo que es una red neuronal y una red neuronal bayesiana, que a diferencia de los modelos del capítulo dos, resultan ser módelos no lineales. En el capítulo cuatro implementaremos una red neuronal bayesiana de una sola neurona. Con esta finalidad introduciremos los métodos de muestreo *Markov Chain Monte Carlo* y *Monte Carlo Hamiltoniano* que resultan indispensables en la simulación de la red.

2. Métodos lineales para clasificación

En el contexto del aprendizaje de máquina, cuando la variable de respuesta y está modelada de la siguiente forma:

$$y(x) = \sigma(\beta_0 + \beta^T x), \tag{2.1}$$

se dice que es parte de los *métodos lineales para clasificación*. Por ejemplo, si σ es la identidad tenemos el módelo de regresión lineal, sin embargo, la expresión (2.1) no tiene por qué ser lineal como lo veremos en esta sección.

Supongamos que cada una de nuestras observaciones son vectores de $\mathbb{R}^N$. Una manera de resolver el problema de la clasificación es encontrar la manera de describir o aproximar geométricamente cada una de las regiones $\mathcal{C}_k \subset \mathbb{R}^n$ que representan cada clase. Cuando representamos a cada clase por regiones que están separadas o delimitadas por conjuntos de la forma

$$\left\{x \in \mathbb{R}^n : \beta_0 + \beta^T x = 0\right\}$$

decimos que nuestro método de clasificación genera *fronteras lineales de decisión*, pues cada una de estas fronteras está dada por espacios lineales afines.

Los siguientes métodos nos dan resultados cuya frontera de decisión son lineales.

2.1. Regresión Logística

La regresión logística es un método el cual modela la probabilidad (1.2) como variable de respuesta. Este modelo utiliza a la función *logśtica* o la *sigmoide*:

$$\sigma(x) := \frac{\exp x}{1 + \exp x} = \frac{1}{1 + \exp(-x)},$$

para corregir el hecho de que las probabilidades caigan en el intervalo $[0, 1]$. La Figura 2.1 muestra un esquema de la gráfica de la sigmoide.

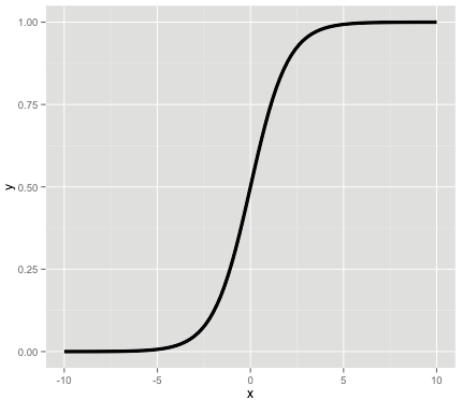

Figura 2.1: Función sigmoide.

En nuestro caso, lo que tenemos es entonces

$$P(Y = 0|X) := P(X) = \frac{\exp(\beta_0 + \beta^T X)}{1 + \exp(\beta_0 + \beta^T X)}. \tag{2.2}$$

Obsérvese que esta expresión anterior tiene la forma de la ecuación (2.1), con σ una función no lineal. Aplicando el logaritmo a (2.2) obtenemos:

$$\log(\frac{(P(X)}{1 - P(X)}) := \beta_0 + \beta^T X, \tag{2.3}$$

la cual, resulta ser una funcón lineal en los parámteros. La regresión logística busca estimar cada uno de los parámteros de (2.3) a partir de las observaciones. En la Figura 2.2 se puede observar cómo la función logística modela mucho mejor la probabilidad $P(Y = 0|X)$ que el modelo (1.2).

Del hecho de que modelamos nuestras probabilidad utilizando (2.2), podemos utilizar la función de verosimilitud y encontrar el estimador máximo verosímil de nuestros parámetros. En nuestro caso, la función de verosimilitud está dada por:

$$l(\beta_0, \beta; x_1, ..., x_N) : \prod_{i:y_i=1} P(x_i) \prod_{j:y_j=0} (1 - P(x_j)).$$

Con y_i la variable que vale 1 si la observación x_i pertence a la clase $\mathcal{C}_0$ y 0 en el otro caso. Diferenciando el logaritmo de la función de verosimilitud e igualando a cero el gradiente se obtiene:

$$\nabla log(l) = \sum_{i=1}^{N} x_i(y_i - P(x_i; \beta_0, \beta)) = 0,$$

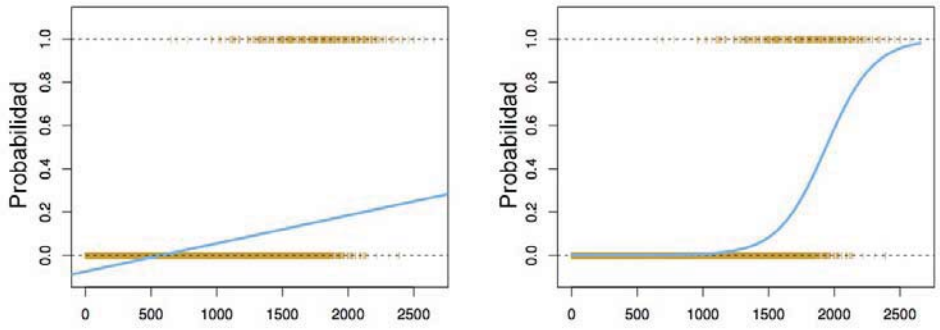

Figura 2.2: A la izquierda vemos la probabilidad estimada utilizando regresión lineal. A la derecha utilizamos regresión logística. Los puntos gruesos en color naranja representan los valores de nuestras observaciones, en 0 los de la clase $\mathcal{C}_0$ y en 1 los de la clase $\mathcal{C}_1$.

Encontrar una solución explícita al problema anterior es sumamente complicado, por lo que se utilizan métodos numéricos para aproximar la solución. Quizás el más utilizado es el método Newton-Raphson [2], este algortimo es bastante eficaz cuando nuestro problema es clasificar nuestra información en sólo dos clases. Obsérvese que cuando obtenemos los parámetros, podemos asignar a nuestro conjunto de prueba la clase que le corresponde; si $P(Y = 0|X = x) > 0.5$ entonces $x \in \mathcal{C}_0$, y si $P(Y = 0|X = x) < 0.5$, $x \in \mathcal{C}_1$. De lo anterior tenemos que nuestra frontera de decisión está dada por la ecuación.

$$P(Y = 0|X = x) = \frac{\exp(\beta_0 + \beta^T x)}{1 + \exp(\beta_0 + \beta^T X)} = 0.5.$$

Cuyas soluciones satisfacen la ecuación lineal

$$\beta_0 + \beta^T x = 0,$$

por lo que la frontera de decisión en este método es lineal.

Existen maneras de generalizar todo este proceso cuando tenemos k clases, pero que no hacen uso de la función logística. Una de ellas utliza la función multivariada *softmax*:

$$\sigma(z)_j := \frac{\exp(z_j)}{\sum_{i=1}^{k} \exp(z_i)}.$$

para modelar la probabilidad de que cada observación se encuentre en su respectiva clase $\mathcal{C}_j$.

2.2. Análisis de Discriminante Lineal

A diferencia de la regresión logística, es decir la que utiliza la sigmoide, el *análisis del discriminante lineal* permite clasificar nuestros datos en más de dos clases. Análogamenete, buscamos modelar probabilidades del estilo de (1.2), pero haciendo uso del teorema de Bayes, es decir:

$$P(Y = k|X = x) = \frac{\pi_k f_k(x)}{\sum_{l=1}^{K} \pi_l f_l(x)}, \tag{2.4}$$

donde π_k es la probabilidad inicial de que una observación cualquiera pertenzca a la clase $\mathcal{C}_k$ y $f_k(x)$ es la función de densidad de la probabilidad $P(X = x|Y = k)$.

Usualmente, para este tipo de método, se supone que la distribución de $P(X = x|Y = k)$ es *normal* o *Gaussiana*, es decir

$$f_k(x) := \frac{1}{(2\pi)^{\frac{p}{2}}|\Sigma|^{\frac{1}{2}}} e^{-\frac{1}{2}(x-\mu_k)^T\Sigma^{-1}(x-\mu_k)}.$$

Para comparar a que clase debe pertencer una observación x, lo único que tenemos que hacer es tomar el logaritmo del cociente de probabilidades:

$$\log\Big(\frac{P(Y=k|X=x)}{P(Y=l|X=x)}\Big) = \log(\frac{\pi_k}{\pi_l}) - \frac{1}{2}(\mu_k+\mu_l)^T\Sigma^{-1}(\mu_k-\mu_l) + x^T\Sigma^{-1}(\mu_k-\mu_l), \tag{2.5}$$

la cual es una función lineal del valor x. Así pues, nuestras fronteras de decisión que definen a cada clase son lineales. La fórmula (2.5) también nos ayuda a encontrar la manera de clasificar a cualquier observación; un dato x estará en la clase $\mathcal{C}_j$ si j es aquel valor donde se maximiza la función

$$\delta_j(x) := \log(P(Y=j|X=x)) = x^T\Sigma^{-1}\mu_j - \frac{1}{2}\mu_j^T\Sigma^{-1}\mu_j + \log(\pi_j), \tag{2.6}$$

a la que se le suele llamar la función *discriminante*.

Nótese que todavía nos falta encontrar los valores de nuestros parametros π_k, μ_k, Σ. Éstos usualmente se estiman busacando los parámetros máximo verosímiles de la probabilidad posterior (2.4), los cuales están dados de la siguiente forma:

$$\hat{\pi}_k := \frac{N_k}{N},$$

$$\hat{\mu_k} := \sum_{y_i=k} \frac{x_i}{N_k},$$

$$\hat{\Sigma} := \sum_{k=1}^{K}\sum_{y_i=k} \frac{(x_i-\hat{\mu_k})(x_i-\hat{\mu_k})}{N-K},$$

donde N_k es el numḿero de las observaciones $x_1, ..., x_N$ que pertencieron a la clase $\mathcal{C}_k$. Obsérvese que $\hat{\mu_k}$ es la media muestral de las muestras que pertencen a la clase $\mathcal{C}_k$ y $\hat{\Sigma}$ es la varianza muestral. En el caso de que Σ_k no sea la misma covarianza para todas las funciones f_k, el discriminante (2.6) no se complica tanto, de hecho solamente se vuelve cuadrático en x; el problema es tratar de dar soluciones explícitas para los estimadores máximo verosímiles.

Cuando las clases están bien separadas, o si n es pequeña y la distribución de $P(X = x - |Y = k)$ es aproximadamente normal, el método de discriminante lineal es más estable y efectivo que el método de regresión logística. Se puede probar que la regresión logística y el análisis del discriminante comparten el mismo modelo lineal cuando sólo tenemos dos clases, es decir, ambos son de la forma:

$$\frac{P(Y=0|X=x)}{1-P(Y=0|X=x)} = \beta_0 + \beta^T x.$$

Sin embargo, la regresión logística maximiza la verosimilitud condicional, mientras que el análisis del discriminante maximiza la verosimilitud completa.

2.3. El Perceptrón de Rossenblat

El perceptrón es, quizás, uno de los algoritmos más trascendentes en el estudio del reconocimiento de patrones y clasificación. Entre 1980 a 1990, este método fue el fundador de la teora de las *redes neuronales* y es, en la actualidad una de las teorías más importantes del *aprendizaje de máquina*.

El perceptrón es un algoritmo para encontrar fronteras lineales de decisión, el cual está más enfocado en el área de optimización y en el análisis numérico que en el área de estadística, pero igualmente con las redes neuronales, nos sirve para entender los fundamentos del aprendizaje estadístico.

Antes de llegar a explicar como funciona el perceptrón, debemos recordar ciertos resultados elementales de álgebra lineal (o geometría analítica) acerca de hiperplanos afines:

1. Para cualesquiera dos puntos x_1, x_2 en el hiperplano $\mathbf{L}$, $\beta^T(x_1 - x_2) = 0$, por lo que el vector normal unitario a la superficie $\mathbf{L}$ es $\frac{1}{\|\beta\|}\beta$.

2. Para todo punto x_0 en L, $\beta^T x_0 = -\beta_0$.

3. La distancia (con signo) de cualquier punto x a $\mathbf{L}$ está dada por

$$\beta^T(x - x_0) = \frac{1}{\|\beta\|}(\beta^T x + \beta_0).$$

Suponga que tenemos $x_1, ..., x_N$ puntos perfectamente clasificados en dos clases: $\mathcal{C}_0$ y su complemento $\mathcal{C}_1$. El perceptrón es un clasificador que intenta encontrar un hiperplano que separe a los puntos de ambas clases y que de cierta forma minimice la distancia de los puntos que se encuentren "*mal clasificados*".

El algoritmo funciona de la siguiente manera: Decimos que $y_i := 1$ si $x_i \in \mathcal{C}_0$ y $y_i = -1$ en otro caso. Empezamos con un vector cualquiera $(\beta, \beta_0) \in \mathbb{R}^n \times \mathbb{R}$. Buscamos que nuestro vector (β, β_0) cumpla la siguiente propiedad: los puntos de la clase $\mathcal{C}_0$ deberán satisfacer $\beta^T x + \beta_0 > 0$ y los que no estén en esta clase deberán satisfacer la desigualdad contraria. Designaremos a un punto x_i como *mal clasificado* si $y_i = 1$ con $\beta^T x + \beta_0 < 0$. Asimismo si $y_i = -1$ y se satisface la desigualdad contraria, x_i será un punto mal clasificado. Sea $\mathcal{M}$ los índices de los puntos x_i mal clasificados. Nuestro objetivo es minimizar la siguinte función:

$$D(\beta, \beta_0) := -\sum_{i \in \mathcal{M}} y_i(\beta^T x_i + \beta_0).$$

Observe que por la definición de y_i y de los puntos mal clasificados, esta función es no-negativa y es proporcional a la distancia de los puntos malclasificados a la frontera de decisión, definida por $\beta^T x + \beta_0 = 0$. Suponiendo que $\mathcal{M}$ es fijo, el gradiente de D está dado por

$$\partial \frac{D}{\partial \beta} = -\sum_{i \in \mathcal{M}} y_i x_i,$$

$$\partial \frac{D}{\partial \beta_0} = -\sum_{i \in \mathcal{M}} y_i.$$

El algoritmo del perceptrón usa el método de *gradiente descendente estocástico* para encontrar numéricamente un mínimo de D, esto quire decir, actualizaremos el vector de pesos (β, β_0) en la dirección contraria al gradiente de la siguiente forma:

$$(\beta, \beta_0) \leftarrow (\beta, \beta_0) - \rho \nabla D = (\beta, \beta_0) - \rho(y_i x_i, y_i),$$

asimismo los valores de los parámetros para cada una de las observaciones mal clasificadas (índices $i \in \mathcal{M}$), de aquí proviene el término estocástico en el nombre del algoritmo. El valor ρ es un parámetro en el intervalo $[0, 1]$ que optimiza el algoritmo, usualmente a esta constante se le llama el *parámetro de aprendizaje*. Puede demostrarse que el algoritmo converge si los puntos x_i son linealmente separables (es decir, si existe un hiperplano que separe las observaciones).

No obstante persisten algunos problemas:

- Cuando los datos son separables por un hiperplano, el algoritmo puede converger a una infinidad de soluciones, y éstas dependeran de como cambiemos el punto inicial del algoritmo.

- El número de pasos puede ser muy grande para encontrar los parámetros.

- Si los datos no son linealmente separables el algoritmo no converge.

Por último, veamos que este modelo es lineal. Una vez estimado nuestros parámetros, nuestra variable respuesta, que en este caso es cualitativa, viene dada por:

$$y(x) = \sigma(\beta_0 + \beta^T x),$$

donde σ es la función:

$$\sigma(a) = \begin{cases} +1 & \text{si} \quad a > 0, \\ -1 & \text{si} \quad a < 0. \end{cases} \tag{2.7}$$

Los modelos lineales nos brindan propiedades útiles para calcular estimaciones y analizar resultados, no obstante, son bastante limitados pues su desempeño a la hora de clasificar es muy pobre cuando la base de datos es muy grande o presenta alta dimensionalidad. En la siguiente sección veremos una generalización de los métodos lineales en donde la ecuación (2.1) resulta ser altamente no lineal, estos modelos suelen hacer un mejor trabajo en problemas a gran escala.

3. Redes Neuronales: un método no lineal

3.1. Redes Neuronales

El término de red neuronal ha estado involucrado en una gran cantidad de ramas de la ciencia relacionadas a métodos de aprendizaje que han tenido mucho auge en estos últimos años, y en algunos casos se les hace ver como mágicas y misteriosas; sin embargo, se pueden estudiar como un módelo estadśtico no lineal. La idea central de una red es extraer combinaciones lineales de nuestros datos y considerarlas como nuevos *imputs* que llamaremos *unidades escondidas*, y así modelar nuestras variables de respuesta en función de estas *unidades*.

En términos más generales, una red neuronal es un módelo de regresión o clasificación de dos étapas, típicamente representado por un diagrama como el que se muestra en la Figura 3.1. Por este tipo de diagramas viene la nomenclatura de *red neuronal*. Para regresión, usualemente sólo tenemos una respuesta, es decir en nuestro diagrama tendríamos $K = 1$, sin embargo, las redes se utilizan con más frecuencia en el contexto de la clasificación, en el cual, tenemos K respuestas, las cuales se observan arriba en nuestro diagrama, una por cada clase codificada por un 0 o 1.

Como mencionamos, nuestras unidades escondidas Z_m son derivados de combinaciones lineales de nuestros datos $X = (X_1, ..., X_p)$, y luego nuestra variable respuesta Y_k es modelada en función de combinaciones lineales de nuestras variables Z_m. En resumen tenemos lo siguiente:

$$\begin{aligned} Z_m &:= \sigma(\alpha_{0,m} + \alpha_m^T X), \quad m = 1, ..., M, \\ T_k &:= \beta_{0,k} + \beta_k^T Z \quad k = 1, ..., K, \\ Y_k &:= f_k(X) = g_k(T) \quad k = 1, ...K, \end{aligned} \tag{3.1}$$

donde $Z = (Z_1, ..., Z_M)$ y $T = (T_1, ..., T_K)$.

A σ se le llama la función de activación que se escoge usualmente como una función sigmoide

$$\sigma(v) := \frac{1}{1 + \exp(-v)}.$$

Usualmente en los diagramas de redes neuronales es común dibujar un vértice más de las Z_i y Y_i, que viene representando a los parámetros $\alpha_{0,m}$ y $\beta_{0,k}$ en el módelo (3.1).

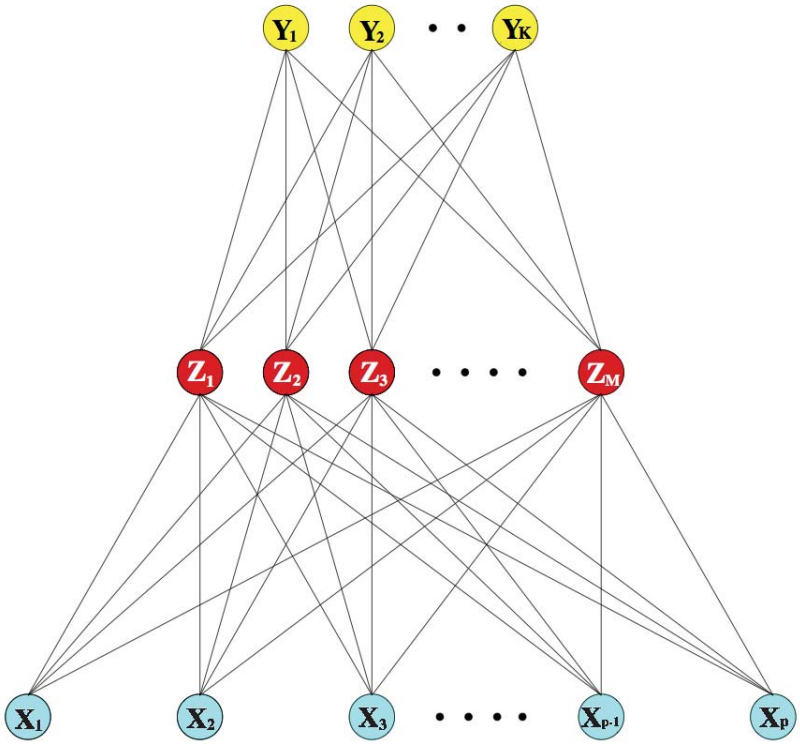

Figura 3.1: Esquema de una red neuronal abstracta.

La salida $g_k(T)$ permite una transformación final al vector de respuesta T. En estos últimos años, al igual que en el módelo de regresión logística, en el problema de clasificación con K clases, se obtienen muy buenos resultados utilizando la función *softmax*:

$$g_k(T) := \frac{exp(T_k)}{\sum_{l=1}^{K} exp(T_l)},$$

observe que esta función produce estimaciones de la variable de respuesta Y_k de valores en el intervalo $[0, 1]$ que suman uno.

Como mencionames anteriormenete, las variables Z_m se les conoce como unidades escondidas, pues nunca son observadas. Nótese que si σ es la sigmoide y si sólo tenemos una respuesta g_1, con g_1 la función identidad, el módelo (3.1) nos regresa el módelo lineal de regresión logística. Por lo tanto, las redes neuronales representan una generalización no lineal de ambos modelos lineales. En el caso que σ sea la identidad y g_1 la función (2.7), obetenemos el perceptrón.

Así como en nuestros modelos anteriores, tenemos que encontrar los parámetros que mejor encajen utilizando nuestro conjunto de entrenamiento. Sean:

$$\theta_0 := (\alpha_{0,1}, ..., \alpha_{0,M}, \beta_{0,1}, ..., \beta_{0,K}),$$
$$\theta := (\alpha_1, ..., \alpha_M, \beta_1, ..., \beta_K).$$

Para estimar los parámetros se utiliza la suma de cuadrados del error que está dada de la siguiente forma:

$$R(\theta_0, \theta) := \sum_{k=1}^{K} \sum_{i=1}^{N} (y_{ik} - f_k(x_i))^2. \tag{3.2}$$

Asimismo, en el problema de clasificación suele usarse el error de sumas al cuadrado, pero se acostumbra utilizar el negativo del logaritmo de verosimilitud, también conocido como *entropía cruzada* que viene dado por:

$$R(\theta_0, \theta) := -\sum_{k=1}^{K} \sum_{i=1}^{N} y_{ik} \log(f_k(x_i)), \tag{3.3}$$

donde $y_{ik} = 1$ si x_i pertenece a la clase k. La función (3.3) se deduce de manera similar a la regresión logística y los parámetros se estiman por máxima verosimilitud. En esta parte únicamente consideraremos a la función R como la obtenida por le ecuación (3.2), pues el problema de clasificación lo vamos a atacar utilizando la estadística bayesiana como lo veremos en la siguiente sección.

Típicamente, no buscamos un mínimo global de R, pues en la mayoría de las veces los mínimos globales resultarán en un *sobreajuste* del modelo, por lo que se procede por un método conocido como *propagación hacia atrás*, descrito de la siguiente forma: sean $\{x_1, ..., x_N\}$ nuestro conjunto de entrenamiento y $z_{mi} := \sigma(\alpha_{0m} + \alpha_m^T x_i)$, sea $z_i := (z_{1i}, z_{2i}, ..., z_{Mi})$. Entonces, tenemos que si

$$R_i := \sum_{k=1}^{K} (y_{ik} - f_k(x_i))^2 \quad i = 1, ..., N.$$

Entonces, las derivadas parciales de R_i están dadas de la siguiente manera

$$\frac{\partial R_i}{\partial \beta_{km}} = -2(y_{ik} - f_k(x_i)) g_k'(\beta_k^T z_i) z_{mi},$$

$$\frac{\partial R_i}{\partial \alpha_{ml}} = -\sum_{k=1}^{K} 2(y_{ik} - f_k(x_i)) g_k'(\beta_k^T z_i) \beta_{km} \sigma'(\alpha_m^T x_i) x_{il}.$$

Sean,

$$\delta_{ki} := -2(y_{ik} - f_k(x_i)) g_k'(\beta_k^T z_i)$$

$$s_{mi} := -\sum_{k=1}^{K} 2(y_{ik} - f_k(x_i)) g_k'(\beta_k^T z_i) \beta_{km} \sigma'(\alpha_m^T x_i),$$

así pues

$$\frac{\partial R_i}{\partial \beta_{km}} = \delta_{ki} z_{mi},$$

$$\frac{\partial R_i}{\partial \alpha_{ml}} = s_{mi} x_{il}.$$

$$s_{mi} := \sigma'(\alpha_m^T x_i) \sum_{k=1}^{K} \beta_{km} \delta_{ki},$$

a esta última expresión se le conoce como *ecuaciones de propagación*. El algoritmo funciona como sigue: primero, empezamos con parámetros (θ_0, θ) cualesquiera y calculamos la respuesta $\hat{f}_k(x_i)$ por medio de

la fórmula (3.1). En el siguiente paso se calculan los valores δ_{ki}, s_{mi} usando las ecuaciones de propagación y, por último, se actualizan los parámetros de manera similar al usado en el perceptrón, por medio del método de gradiente descendente:

$$\hat{\beta_{km}} := \beta_{km} - \gamma \sum_{i=1}^{N} \frac{\partial R_i}{\partial \beta_{km}},$$

$$\hat{\alpha_{ml}} := \alpha_{ml} - \gamma \sum_{i=1}^{N} \frac{\partial R_i}{\partial \alpha_{ml}},$$

donde γ es la *constante de aprendizaje*, la cual se usa por cuestiones de convergencia del algoritmo. El procedimiento anterior se repite con los parámetros actualizados hasta obtener convergencia. Usualmente, el parámetro γ es actualizado para cada iteración del algoritmo por un número que se va a aproximando a cero conforme avanzamos en el mismo.

No obstante, existen algunos problemas con las redes neuronales:

- La elección de los valores iniciales de los parámetros (θ_0, θ) puede afectar la efectividad del módelo y lleva a soluciones pobres. Se suele tomar los parámetros cerca de cero, y así, el algoritmo empieza aproximadamente lineal y conforme avanza la no linealidad aumenta y los mismos empiezan a aumentar de valor.
- Si se repite la iteración un número grande de veces es muy común que el módelo quede sobreajustado pues converge a un mínimo global de R. Existen métodos para evitar esto deteniendo la iteración del algoritmo en un número óptimo.
- El número de *unidades escondidas* afecta mucho la efectiviadad de la red; entre menos unidades el algoritmo converge más rápido pero la red se vuelve poco flexible y, por esta razón, se acostumbra a usar una cantidad grande de unidades escondidas aunque la convergencia sea lenta.

3.2. Redes Neuronales Bayesianas

Como vimos en la sección anterior, las redes neuronales se pueden considerar como un problema de estadística no lineal, de hecho, a este tipo de modelos se les llama *no paramétricos*, es decir, un modelo que tiene muchos parámetros que en su mayoría no pueden ser interpretados a diferencia de los paramétricos. Usualmente, los tipos de modelos no paramétricos son estudiados desde el punto de vista de la estadística bayesiana, pues en ocasiones con esta perspectiva podemos encontrar relaciones de independencia (o dependencia) entre las variables.

En el contexto de clasificación, una red bayesiana sigue el mismo modelo que en la ecuación (3.1), pero ahora suponemos que las funciones f_k no simulan una respuesta del modelo, si no más bien, como los modelos estadśticos de la sección anterior, modelan las probabilidades $P(Y = k|x) = f_k(x)$.

Para predecir a que clase pertence una nueva observación que no se encuentra en nuestro conjunto de entrenamiento, x_{N+1}, tenemos que estimar los parámetros (θ_0, θ) para encontrar el valor de:

$$P(Y = k|x_{N+1}).$$

Para esto usamos la teoría de estadística bayesiana, similarmente a lo que hicimos en análisis del discriminante lineal. Sea $T = \{(x_1, y_1), ..., (x_N, y_N)\}$ nuestro conjunto entrenamiento con sus debidas respuestas. Supongamos que tenemos una distribución incial de nuestros parámetros $P(\theta_0, \theta)$, entonces la inferencia de nuestros parámetros (θ_0, θ) está dada por:

$$P(\theta_0, \theta|T) = \frac{P(T|\theta_0, \theta)P(\theta_0, \theta)}{\int P(T|\theta_0, \theta)P(\theta_0, \theta)d\theta_0 d\theta}, \tag{3.4}$$

donde $P(T|\theta_0, \theta)$ es la función de verosimilitud.

Teniendo el valor de (3.4), podemos predecir el valor de $P(Y_{N+1} = k|x_{n+1}, T)$ por medio de la fórmula:

$$P(Y_{N+1} = k|x_{N+1}, T) = \int P(Y_{N+1} = k|x_{N+1}, \theta_0, \theta)P(\theta_0, \theta|T)d\theta_0 d\theta, \tag{3.5}$$

donde

$$P(Y_{N+1} = k|x_{N+1}, \theta_0, \theta) = f_k(x_{N+1}).$$

En la práctica, la distribución inicial de los parámetros $P(\theta_0, \theta)$ depende también de otros parámetros, digamos α, que también tienen una distribución en incial $P(\alpha)$, la cual se considera en el análisis anterior a la hora de calcular la distribución posterior de los parámetros; en este caso la distribución nos quedaría de la siguiente forma:

$$P(\theta_0, \theta, \alpha|D) \propto P(\theta_0, \theta|D, \alpha)P(\alpha) \propto P(D|\theta_0, \theta)P(\theta_0, \theta|\alpha)P(\alpha).$$

A α se le llama *hiperparámetro*, para ser diferenciado de los parámetros del modelo sobre los cuales hacemos el análisis. Existen dos razones de por las que se usan estos hiperparámetros: con una distribución inicial conjugada, la distribución posterior será de la misma familia paramétrica que la incial, pero tendrá distintos hiperparámetros, los cuales reflejan la información contruibuida por los datos en la distribución posterior. La otra razón va en el sentido del *sobreajuste* del modelo; al darle una distribución inicial a los hiperparámetros, uno puede variarlos de manera sencilla y observar qué tan sensible es la distribución posterior de el modelo con esa distribución inicial, si el modelo varía mucho uno puede sospechar que esa distribución inicial no es la adecuada.

En resumen, podemos saber la probabilidad de que una nueva observación pertenezca a la clase $\mathcal{C}_k$ si encontramos la manera de calcular la integral (3.5). En la siguiente sección hablaremos de los métodos de *Markov Chain Monte Carlo*, los cuales han sido escenciales a la hora de implmentar la teoría bayesiana en modelos reales. Estos métodos nos permiten hacer buenas estimaciones de este tipo de integrales.

4. Métodos Monte Carlo

Como vimos, en las redes neuronales bayesianas el objetivo es hacer predicciones encontrando las probabilidades predictivas (3.5). Esta tarea requiere que evaluemos la esperanza de una función respecto a la distribución posterior de los parámetros del modelo. Si escribimos como $Q(\theta)$ a la densidad de la probilidad posterior, la esperanza de una funcón $a(\theta)$ es:

$$E[a] := \int a(\theta)Q(\theta)d\theta,$$

en nuestro caso, buscamos calcular la esperanza de la función $a(\theta) := f_k(x_{n+1}; \theta)$.

Tales esperanzas pueden ser estimadas por métodos *Monte Carlo*, usando valores de una muestra de la distribución Q:

$$E[a] \approx \frac{1}{N}\sum_{i}^{N} a(\theta_i), \tag{4.1}$$

con $\theta_1, ..., \theta_N$ generados por un proceso que hace que cada uno de estos valores tengan una distribución definida por Q.

Dicha serie de valores θ_i puede ser generada por una *Cadena de Markov* que tiene a Q como su distribución estacionaria. La cadena es definida por una distribución inicial para el primer estado de la cadena, θ_1, y un conjunto de *densidades de transición* para un nuevo estado θ_{i+1}, seguido del estado

presente, θ_i. La densidad de las probabilidades de transición se denotan como $f(\theta_{i+1}, \theta_i)$. Una *distribución estacionaria*, Q, es una distribución que persiste una vez que se estabiliza el proceso (esto es, si θ_i tiene la distribución dada por Q, entonces $\theta_{i'}$ tendrá la misma distribución Q para toda $i' > i$). Esta condición estacionaria se puede escribir como sigue:

$$Q(\theta') = \int f(\theta'|\theta)Q(\theta)d\theta.$$

Una cadena de Markov es *ergódica* si tiene una única distribución estacionaria, su *equilibrio*, en la cual converge para cualquier estado inicial. Si podemos encontrar una cadena de Marokov que tiene como equilibrio a Q, podemos estimar esperanzas con respecto a Q usando la expresión (4.1), con $\theta_1, ..., \theta_N$ siendo estados de la cadena; quizá con algunos de los primeros estados descartados, pues posiblemente éstos no son representativos del equilibrio. Debido a las dependencias entre los θ_i, el número de valores θ que necesitamos para que nuestra estimación logre un cierto nivel de precisión puede ser muy grande. La cadena también quizas requiera mucho tiempo para alcanzar cierto punto de la distribución en donde es una buena aproxiamción al equilibrio.

Para usar el método de *Markov Chain Montecarlo* para estimar la esperanza con respecto a cierta distribución, Q, necesitamos construir una cadena de Markov que sea ergódica, que tenga a Q como a su distribución equilibrio, que converga a esa distribución la más rápido posible, y que una vez alcanzada la distribución, los estados que se obtengan no resulten ser altamente dependientes.

Cuando la distribución Q no es complicada, existen métodos que presentan todas las caratrísticas que se mencionaron anteriormente, y son bastante sencillos de implementar. Estos métodos son los siguientes:

- *Método de Inversión*: Trata de encontrar la inversa de la función de distribución comulativa. Funciona con distribuciones uniformes o normales.
- *Muestreo de Rechazo*: La idea básica de este método consiste en muestrear a partir de una distribución instrumental y rechazar muestras con baja probabilidad que provengan de la distribución que nos interesa. Funciona con distribuciones gamma y normales.
- *Muestreo de Importancia*: Al igual que el muestreo de rechazo, se utiliza una distribución instrumental y se muestrea con esa distribución, pero se utilizan pesos para corregir el hecho de que estamos tomando muestras de una distribución que no es la buscada. Se utiliza para estimar ezperanzas de un grupo particular de distribuciones, pero la eficiencia de esa estimación depende, en mayor parte, de la selcción de la distribución instrumental. Encontrar dicha distribución en dimensiones altas resulta ser muy complicado.

Como mencionamos anteriormente, implementar estos métodos de muestreo resulta ser muy sencillo, pero son poco generalizables para cualquier tipo de distribución. En la siguiente sección discutiremos dos métodos que resultan ser fundamentales en el proceso de estimar (4.1) por el modelo de redes nueronales bayesianas y llegan a funcionar para cualquier distribución de interés.

4.1. Método del muestreo de Gibbs

El muestreo de Gibbs es aplicable si buscamos encontrar muestras de una distribución multivariada. Supongamos que la tarea de encontrar muestras $\theta := (\theta^{(1)}, ..., \theta^{(p)})$ de la distribución Q es poco viable, pero supongamos que podemos generar muestras de la distribución condicional (bajo Q) para una de las componentes de θ, dadas los valores de las otras componentes. Esto perimite realizar una cadena de Markov θ_{i+1} a partir de θ_i como sigue:

- Encuentra una muestra $\theta_{i+1}^{(1)}$ de la distribución de $\theta^{(1)}$ dada $\theta_i^{(2)}, ..., \theta_i^{(p)}$.

- Encuentra una muestra $\theta_{i+1}^{(2)}$ de la distribución de $\theta^{(2)}$ dada $\theta_{i+1}^{(1)}, \theta_i^{(3)}, ..., \theta_i^{(p)}$.
- ...
- Encuentra una muestra $\theta_{i+1}^{(j)}$ de la distribución de $\theta^{(j)}$ dada $\theta_{i+1}^{(1)}, ..., \theta_{i+1}^{(j-1)}, \theta_i^{(j+1)}, ..., \theta_i^{(p)}$.
- ...
- Encuentra una muestra $\theta_{i+1}^{(p)}$ de la distribución de $\theta^{(p)}$ dada $\theta_{i+1}^{(1)}, \theta_{i+1}^{(2)}, ..., \theta_{i+1}^{(p-1)}$.

Bajo condiciones de positividad de la distribución Q, se puede probar que la cadena de Markov generada por el proceso dado por el muestreo de Gibbs es ergódica (consúltese [6]).

El hecho de que el muestreo de Gibbs sea útil para la inferencia bayesiana depende del hecho de que las distribuciones posteriores de cada una de las distribuciones de los parámetros, dados los valores de los otros, sean fáciles de muestrear y así aplicar los métodos vistos en la sección anterior. Para redes neuronales bayesiana, usualmente se utiliza el muestreo de Gibbs para obtener muestras de los hiperparámetros, ya discutidos con anterioridad.

4.2. El Algorítmo Metrópolis

La cadena de Markov definida por el algorítmo Metrópolis es similar al muestreo de importancia; un nuevo estado θ_{i+1}, es generado del estado previo θ_i primero utilizando una distribución instrumental especificada, y después, decidimos si aceptamos a este candidato basándonos en la densidad de probabilidad relativa al estado anterior, con repecto a la distribución invariante deseada, Q. Si el candidato se acepta, éste se vuelve un nuevo estado de la cadena de Markov, si se rechaza se busca a otro candidato de la misma forma.

En detalle, la transición θ_i a θ_{i+1} está explicada como sigue:

1. Genera un candidato θ^* a partir de una distribución instrumental, la cual puede depender del estado anterior θ_i. Supongamos que la densidad de esta distribución es $S(\theta^*|\theta_i)$.
2. Si $(\theta^*) \geq Q(\theta_i)$, se acepta el nuevo candidato; si no, se acepta el candidato con probabilidad $\frac{Q(\theta^*)}{Q(\theta_i)}$.
3. Si el candidato es aceptado, sea $\theta_{i+1} = \theta^*$; si el candidato se rechaza, sea $\theta_{i+1} = \theta_i$.

La distribución $S(\theta'|\theta)$ debe de ser simétrica. En algunos contextos, $Q(\theta)$ se define en términos de una función de "energía", $E(\theta)$, y $Q(\theta) \propto \exp(-E(\theta))$. En el paso (2), se acepta el candidato con baja energía, y si el candidato tiene alta energía, se acepta con probabilidad $\exp(-(E(\theta^*) - E(\theta_i))$.

Para mostrar que estas transiciones mantienen invariante a Q uno puede ver lo siguiente, si $\theta \neq \theta'$ entonces las densidades de transición satisfacen lo siguiente

$$T(\theta'|\theta) = S(\theta'|\theta)\,\text{mín}\{1, \frac{Q(\theta')}{Q(\theta)}\},$$

y por lo tanto

$$\begin{aligned} T(\theta'|\theta)Q(\theta) &= S(\theta'|\theta)\,\text{mín}\{1, \frac{Q(\theta')}{Q(\theta)}\}Q(\theta) \\ &= S(\theta'|\theta)\,\text{mín}\{Q(\theta), Q(\theta')\} \\ &= S(\theta|\theta')\,\text{mín}\{Q(\theta'), Q(\theta)\} \end{aligned}$$

$$= S(\theta|\theta') \,\text{mín}\{1, \frac{Q(\theta)}{Q(\theta')}\} Q(\theta')$$

$$= T(\theta|\theta') Q(\theta').$$

Por lo que el algoritmo Metrópolis mantiene invariante a Q. Bajo suposiciones acerca de las distribuciones Q y S, se puede probar que la cadena es ergódica (consúltese [6]).

Muchas elecciones de la distribución S se pueden hacer para este algoritmo. Una simple elección es tomar una distribución normal centrada en θ con varianza escogida, de tal forma que la probabilidad de aceptar a un nuevo candidato es alta. Si la distibución Q es muy compleja y la dimensión de θ es muy grande, la desviación estandar de la distribución auxiliar tiene que ser pequeña, comparada con la extención de Q, ya que grandes cambios llevan con gran certeza a regiones con poca probabilidad. Esto hace que el algoritmo tenga una dependencia alta de los estados sucesivos, ya que muchos pasos van a ser necesitados para moverse a una distancia considerable de la distribución.

Debido a este tipo de problemas, versiones simples del algoritmo Metrópolis pueden ser muy lentas, por lo que no se usan con mucha frecuencia en la estimación de los parámetros en las redes neuronales bayesianas.

4.3. Métodos Montecarlo Hamiltonianos

Una manera de ver a los métodos Monte Carlo Hamiltonianos es como una composición del método de muestreo de Gibbs y particularmente una versión elaborada del algoritmo Metrópolis. En el caso de redes neuronales bayesianas, la versión elaborada del algoritmo Metrópolis está relacionada con métodos de dinámica estocástica, los cuales están basados en formular los problemas de muestreo en términos de una "función de energía".

Supongamos que buscamos extraer una muestra de una distribución para unas variables de "posición" q de n componentes. La función de densidad de probabilidad para estas variables bajo la distribución canónica está definida por:

$$P(q) \propto \exp(-E(q)) \tag{4.2}$$

donde $E(q)$ es la llamada energía potencial del sistema. Cualquier función de densidad que nunca se anula se puede poner de está forma, simplemente definiendo a $E(q) := -\log(P(q)) - \log(Z)$ para alguna Z conveniente.

Para utilizar la dinámica de los modelos de la mecánica analítica, introducimos variables de "momento", p, también de n componentes. La distribución canónica, comúnmente llamada *estado fase* de q y p, está dada por:

$$P(q,p) :=\propto \exp(-H(q,p)), \tag{4.3}$$

donde $H(q,p) := E(q) + K(p)$ es llamado el *Hamiltoniano*, el cual proporciona la energía total del sistema. A $K(p)$ se le llama la energía cinética a su herencia con el momento dinámico, y usualmente está dada por:

$$K(p) := \sum_{i=1}^{n} \frac{p_i^2}{2m_i}, \tag{4.4}$$

con m_i las masas de las partículas, que en este caso, asignaremos todas de masa unitaria. Suponemos que las variables q y p son variables independientes, por lo que, si contruimos una cadena de Markov que nos permita extraer muestras (q,p), ignoraremos a p en cada muestra y nos quedaremos sólo con la muestras q que al ser idempendientes de p, vendrán de una muestra de la densidad buscada (4.2).

El método estocástico dinámico consiste en simular la dinámica dada por el Hamiltoniano del sistema, la cual, está definida por el siguiente sistema de ecuaciones diferenciales ordinarias:

$$\begin{aligned} \frac{dq_i}{d\tau} &= \frac{\partial H}{\partial p_i} = p_i, \\ \frac{dp_i}{d\tau} &= \frac{\partial H}{\partial q_i} = -\frac{\partial E}{\partial q_i}, \end{aligned} \tag{4.5}$$

así pues, para realizar este método, necesitamos poder calcular las derivadas parciales de la función E.

Hay tres propiedades de las ecuaciones anteriores que son cruciales en el hecho de que tomemos el punto de vista de la dinámica Hamiltoniana para encontrar muestras (q, p). La primera es que el Hamiltoniano permanece constante conforme q y p varían de acuerdo a la dinámica dada por (4.5). La segunda es que la dinámica (4.5) preserva el volumen en regiones del espacio fase. Finalmente, la dinámica (4.5) es reversible, es decir, si seguimos la dinámica hacia adelante en el tiempo por algún periodo, podemos regresar al estado inicial si consideramos la dinámica hacia atrás y en el mismo lapso.

Estas tres propiedades implican que la distribución canónica de q y p son invariantes con respecto a la transición de seguir la trayectoria por un periodo de tiempo usando la dinámica Hamiltoniana. La probabilidad de que terminemos en una pequeña región después de la transición será la misma que en la pequeña región donde empezamos y que podemos encontrar si tomamos la dinámica hacia atrás. Por último, si la probabilidad de cada región está dada por la distribución (4.3), esta seguirá siendo la misma al seguir el flujo de las ecuaciones (4.5), pues H tiene el mismo valor para todos los puntos de la trayectoria.

En el método de dinámica estocástica, formamos una cadena de Markov de la siguiente forma: en el primer paso generamos una muestra de nuestra variable de momentos p_i^* con la distribución $exp(-K(p))$, con K descrito por (4.4), usando el método de muestreo de Gibbs; en el segundo, actualizamos nuestras muestras q_i y p_i^* simulando la dinámica Hamiltoniana de (4.5) en un periodo prestablecido.

En la práctica, la dinámica de Hamiltoniana no se puede simular de manera exacta, pero puede ser aproximada por discretizaciones en un número finito de pasos en el tiempo. El método utilizado usualmente se utiliza para simular la dinámica Hamiltoniana es el llamado de *leapfrog*, el cual encuentra aproximaciones de la posición y el momento en un tiempo $\tau + \epsilon$ como sigue:

$$\begin{aligned} p_i(\tau + \frac{\epsilon}{2}) &= p_i(\tau) - \frac{\epsilon}{2}\frac{\partial E}{\partial q_i}(q(\tau)), \\ q_i(\tau + \epsilon) &= q_i(\tau) + \epsilon p_i(\tau + \frac{\epsilon}{2}), \\ p_i(\tau + \epsilon) &= p_i(\tau + \frac{\epsilon}{2}) - \frac{\epsilon}{2}\frac{\partial E}{\partial q_i}(q(\tau + \epsilon)), \end{aligned} \tag{4.6}$$

cada uno de estos pasos es obtenido a partir de discretizar la derivada de cada una de estas variables con respecto al tiempo. Para seguir la dinámica en algún periodo $\Delta\tau$, se escoge un valor del incremento en el tiempo ϵ, y las expresiones (4.6) son aplicadas una cantidad

$$L = \frac{\Delta\tau}{\epsilon},$$

de veces para llegar al tiempo deseado. Se utiliza el método de *leapfrog* en lugar de otros, pues numéricamente es el método que mejor preserva el volumen, además de que permite revertir la dinámica simplemente aplicando el mismo número de pasos L con ϵ negativa.

Como mencionamos anteriormente, el Método Montecarlo Hamiltoniano es una combinación de el muestreo de Gibbs y una foma elaborada de algoritmo Metrópolis, el cual se decribe como sigue:

Sea ϵ el tamaño de los pasos del método *leapfrog* y sea L la cantidad de pasos que vamos a tomar, entonces:

- Se extrae una muestra de la distribución (4.4) de los momentos, p_i^*, dejando fijo el estado q_i.
- Empezando desde el punto $(q_i, p_i^*) = (q(0), p(0))$, se realizan L pasos de tamaño ϵ de método *leapfrog* (4.6), obteniendo los estados $(q^*, p^*) = (q_i(\epsilon L), p_i^*(\epsilon L))$.
- Hecho lo anterior, se acepta al candidato (q^*, p^*) como el siguiente estado de la muestra con probabilidad:

$$\text{mín}\{1, \exp(-(H(q^*, p^*) - H(q, p))\},$$

de otra forma, se define el nuevo estado como el anterior.

Cuando L es suficientemente grande se obtiene uno de los grandes beneficios de este método: evitar caminatas aleatorias, que es uno de los grandes problemas del algoritmo Métropolis. Por esta razón, en el aprendizaje de redes neuronales bayesianas, es usual utilizar este tipo de métodos. En el siguiente capítulo implementaremos una red neuronal bayesiana y utilizaremos el Método Monte Carlo Hamiltoniano con $L = 1$ (el cual se le conoce como el método de Langevin) pues nuestro modelo no presenta una significativa cantidad de parámetros en un número grande de pasos.

5. Implementación de una Neurona Bayesiana con dos pesos

Construiremos un clasificador de únicamente dos clases con una sola neurona bayesiana, lo cual resume el contenido visto en la sección anterior. Como ejemplo, usaremos la siguiente red neuronal:

$$f_0(x) = \frac{1}{1 - \exp(w_0 + w_1 x_1 + w_2 x_2)},$$

$$f_1(x) = 1 - f_0(x),$$

es decir, en la expresión (3.1), nosostros tenemos:

$$\sigma(v) := \frac{1}{1 - \exp(-v)},$$

$$T(Z) := T_0(Z) = T_1(Z) = Z,$$

$$g_0(T) = T,$$

$$g_1(T) = 1 - g_0(T),$$

que como mencionamos anteriormente, también nos brindan el caso de la regressión logística; sin embargo, nosotros consideraremos a los parámetros como variables aleatorias que provienen de una distribución inicial.

Tomemos a nuestro conjunto de entrenamiento de tamaño N, y supongamos $T := \{(x_1, t_1), ..., (x_N, t_N)\}$, donde

$$t_i = \begin{cases} 1 & \text{si } x_i \in \mathcal{C}_0, \\ 0 & \text{si} x_i \in \mathcal{C}_1. \end{cases}$$

Sea α un número mayor que cero y sea $w = (w_0, w_1, w_2)$ el vector de parámetros. Suponga que la distribución incial de w es una gaussiana multivariada, es decir:

$$P(w) := \frac{\alpha^{\frac{3}{2}}}{\pi^{\frac{3}{2}}} \exp(-\alpha E(w)),$$

donde $E(w) := \sum_{i=0}^{2} \frac{1}{2} w_i^2$. Entonces, nuestra verosimilitud quedaría de la forma:

$$P(T|w) = \exp(-G(w)),$$

con

$$G(w) = \sum_{i=1}^{N} t_i \log(f_0(x_i)) + (1 - t_i) f_1(x_i).$$

A partir de estas dos expresiones podemos calcular la distribución posterior de los parámetros:

$$P(w|T) = \frac{P(T|w)P(w)}{\int P(T|w)P(w)dw} = \frac{1}{C_M} \exp(-M(w)), \tag{5.1}$$

donde

$$M(w) := G(w) + \alpha E(w) \qquad \text{y} \qquad C_M := \int \exp(-M(w))dw.$$

Por lo tanto, si nos dan una nueva observación x_{N+1} y buscamos clasificarla, tenemos que encontrar la cantidad.

$$\begin{aligned} P(t_{N+1} = 0|x_{N+1}, T) &= \int P(t_{N+1} = 0|x_{N+1}, w)P(w|T)dw \\ &= \int f_0(x_{N+1})\Big(\frac{1}{C_M} \exp(-M(w))\Big)dw. \end{aligned} \tag{5.2}$$

Para finalizar este trabjo, utilizando el lenguaje de programación **R**, estimaremos la expresión (5.2) en nuestro conjunto de prueba utilizando la expresión (4.1), y haremos uso del método de Langevin para encontrar muestras de la distribución (5.1), el cual tiene forma de una exponencial elevada a la función de energía M.

5.1. Resultados

Con la libreria `mlbench.2dnormals`, generamos una cantidad de 50 de datos en el plano cuya distribución es normal bivariada, éstos se encuentran divididos en dos clases. Tomamos 30 de éstos como conjunto de entrenamiento y los 20 restantes como conjunto de prueba. Inmediatamente, programamos el método de Langevin con los valores $\alpha = 0.01$, el tamaño de los pasos *leapfrog* de $\epsilon := \sqrt{2\alpha}$ y como muestra inicial el vector de parámetros $w_0 = (0.5, 0.5, 0)$.

Después de 2000 iteraciones del método de Langevin, la muestra de la distribución (5.1) resultante en el paso dos mil, la cual denotamos w_{2000}, está dada por:

```
[1] -1.6596371 -2.0201868 -0.5333094
```

En la Figura 5.1 podemos ver como la pendiente de la recta varía conforme tomamos distintos pasos en el algoritmo de Langevin, y como cada una de estas rectas mejora la separación de cada clase.

Con el vector w_{2000} comparamos los resultados en nuestro conjunto de prueba. En la figura 5.2 podemos comprobar cómo la recta asociada al vector w_{2000} clasifica incorrectamente, en el conjunto de prueba, sólamente tres observaciones.

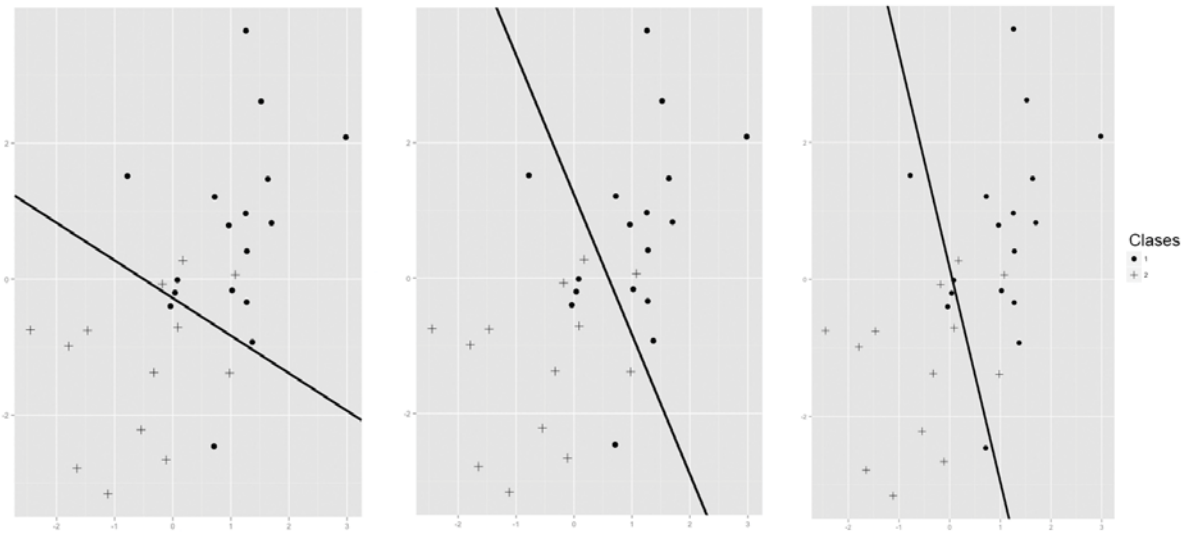

Figura 5.1: En las tres figuras tenemos graficados cada uno de los datos del conjunto de entrenamiento. A la izquierda se encuentra la recta correspondiente al vector de parámetros resultante de 500 iteraciones en el método de Langevin, es decir w_{500}. En medio tenemos a la recta correspondiente a w_{1000}. En la derecha tenemos la recta correspondiente a w_{2000}, la cual es la que mejor separa nuestras clases.

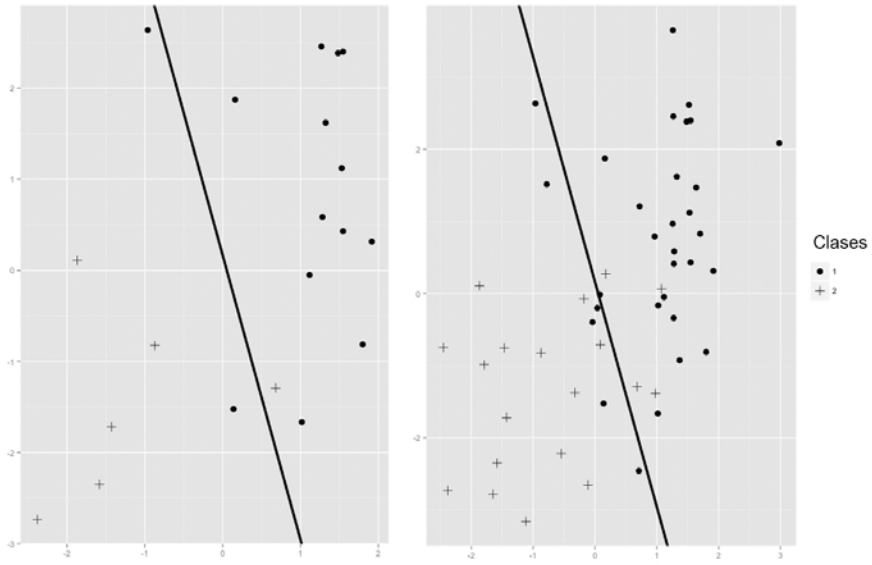

Figura 5.2: A la izquierda tenemos graficado el conjunto de prueba con la recta asociada al vector w_{2000}. En la derecha vemos la gráfica de las 50 observaciones con esa misma recta. En este caso, vemos que la recta clasifica incorrectamente a diez de los datos.

Por último, utlilizando las últimas 1800 muestras del algoritmo, estimamos la cantidad (5.2) con la fórmula (4.1). Posteriormente, usando nuestro conjunto de prueba, clasificamos las observaciones de

acuerdo a si la cantidad (5.2) es mayor que un medio o no. El siguiente vector contiene como primera entrada a la cantidad de vectores que fueron clasificados correctamente por la red neuronal bayesiana, y como segunda entrada los vectores que fueron mal clasificados.

```
[1] 14  6
```

En este caso, podemos observar que el algoritmo pudo clasificar, de manera correcta, casí tres cuartos del conjunto de entrenamiento.

Referencias

[1] Hastie Trevor, Tibshirani Robert, Jerome Friedman, *An Introduction to Statistical Learning*, Springer, 2013.

[2] Hastie Trevor, Tibshirani Robert, Jerome Friedman, *The Elemnets of Statistical Learning*, Springer, Second edition, 2008.

[3] Bishop Christopher M., *Pattern Recognition and Machine Learning*, 2006.

[4] MacKay David J.C., *Information Theory, Inference, and Learning Algorithms*, Cambridge University Press, 2003.

[5] Neal M. Radford, *Bayesian Learning for Neural Networks*, Springer, 1996.

[6] Cosma Ioana, Evers Ludgers, *Markov Chain and Monte Carlo Methods*, Lecture Notes, Cambridge University Press, 2010.

[7] Gelman Andrew, Carlin John B., Stern Hal, Rubin Donald B., *Bayesian Data Analysis*, Chapman Hall/CRC, 2004.