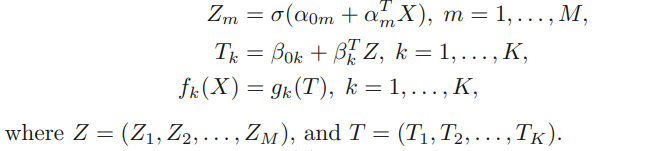
**11.3 Redes neuronales**

El término red neuronal ha evolucionado para abarcar una gran clase de modelos y métodos de aprendizaje. Aquí describimos la red neuronal "vainilla" más utilizada, a veces llamada red de retropropagación de capa única oculta o perceptrón de capa única. Ha habido un gran revuelo en torno a las redes neuronales, lo que las hace parecer mágicas y misteriosas. Como aclaramos en esta sección, son solo modelos estadísticos no lineales, muy parecidos al modelo de regresión de búsqueda de proyecciones discutido anteriormente.

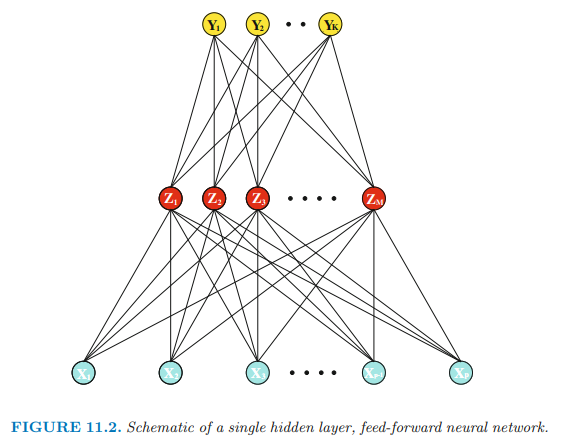
Una red neuronal es un modelo de clasificación o regresión de dos etapas, típicamente representado por un diagrama de red como en la Figura 11.2. Esta red se aplica tanto a la regresión como a la clasificación. Para la regresión, normalmente K = 1 y solo hay una unidad de salida Y1 en la parte superior. Sin embargo, estas redes pueden manejar múltiples respuestas cuantitativas sin problemas, por lo que nos ocuparemos del caso general.

Para la clasificación de clase K, hay K unidades en la parte superior, y la k-ésima unidad modela la probabilidad de la clase k. Hay K medidas de objetivo Yk, k = 1, ..., K, cada una codificada como una variable 0 - 1 para la k-ésima clase. Las características derivadas Zm se crean a partir de combinaciones lineales de las entradas, y luego el objetivo Yk se modela como una función de combinaciones lineales de Zm,



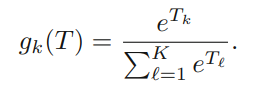
La función de activación σ (v) generalmente se elige para que sea el sigmoide σ (v) = 1 / (1 + e − v); consulte la Figura 11.3 para ver una gráfica de 1 / (1 + e − v). A veces, las funciones de base radial de Gauss (Capítulo 6) se utilizan para σ (v), produciendo lo que se conoce como una red de funciones de base radial.

Los diagramas de redes neuronales como la Figura 11.2 a veces se dibujan con una unidad de polarización adicional que ingresa a cada unidad en las capas ocultas y de salida.



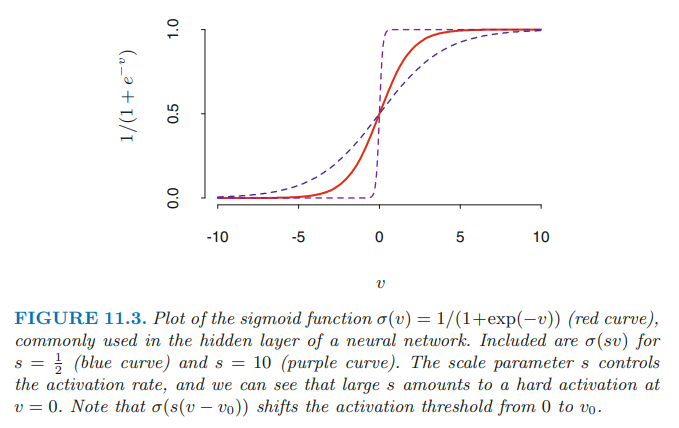
Pensando en la constante “1” como una característica de entrada adicional, esta unidad de sesgo captura las intersecciones α0m y β0k en el modelo (11.5).

La función de salida gk (T) permite una transformación final del vector de salidas T. Para la regresión normalmente elegimos la función de identidad gk (T) = Tk. Los primeros trabajos en la clasificación de la clase K también utilizaron la función de identidad, pero luego se abandonó en favor de la función softmax.



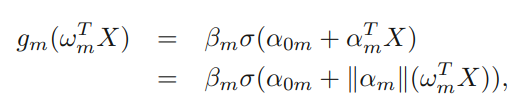
Por supuesto, esta es exactamente la transformación utilizada en el modelo multilogit (sección 4.4) y produce estimaciones positivas que suman uno. En la Sección 4.2 discutimos otros problemas con las funciones de activación lineal, en particular efectos de enmascaramiento potencialmente severos.

Las unidades en el medio de la red, que calculan las características derivadas Zm, se denominan unidades ocultas porque los valores Zm no se observan directamente. En general, puede haber más de una capa oculta, como se ilustra en el ejemplo al final de este capítulo. Podemos pensar en Zm como una expansión de la base de las entradas originales X; la red neuronal es entonces un modelo lineal estándar, o modelo lineal multilogit, utilizando estas transformaciones como entradas. Sin embargo, existe una mejora importante con respecto a las técnicas de expansión básica discutidas en el Capítulo 5; aquí los parámetros de las funciones básicas se aprenden a partir de los datos.



Observe que si σ es la función de identidad, entonces todo el modelo se colapsa a un modelo lineal en las entradas. Por tanto, se puede pensar en una red neuronal como una generalización no lineal del modelo lineal, tanto para regresión como para clasificación. Al introducir la transformación no lineal σ, amplía enormemente la clase de modelos lineales. En la figura 11.3 vemos que la tasa de activación del sigmoide depende de la norma de αm, y si alfa es muy pequeño, la unidad estará operando en la parte lineal de su función de activación.

Observe también que el modelo de red neuronal con una capa oculta tiene exactamente la misma forma que el modelo de búsqueda de proyección descrito anteriormente. La diferencia es que el modelo PPR usa funciones no paramétricas gm (v), mientras que la red neuronal usa una función mucho más simple basada en σ (v), con tres parámetros libres en su argumento. En detalle, viendo el modelo de red neuronal como un modelo PPR, identificamos:

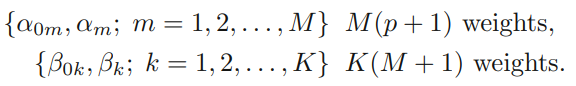


donde ωm = αm / αm es el m-ésimo vector unitario. Dado que σβ, α0, s (v) = βσ (α0 + sv) tiene menor complejidad que una g (v) no paramétrica más general, no es sorprendente que una red neuronal pueda usar 20 o 100 de tales funciones, mientras que el modelo PPR típicamente utiliza menos términos (M = 5 o 10, por ejemplo).

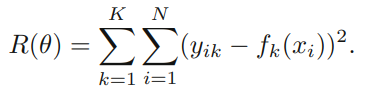
Por último, observamos que el nombre "redes neuronales" se deriva del hecho de que se desarrollaron por primera vez como modelos para el cerebro humano. Cada unidad representa una neurona y las conexiones (enlaces en la Figura 11.2) representan sinapsis. En los primeros modelos, las neuronas se activaban cuando la señal total que pasaba a esa unidad excedía un cierto umbral. En el modelo anterior, esto corresponde al uso de una función escalonada para σ (Z) y gm (T). Más tarde, la red neuronal fue reconocida como una herramienta útil para el modelado estadístico no lineal y, para este propósito, la función de paso no es lo suficientemente suave para la optimización. Por tanto, la función escalón fue reemplazada por una función de umbral más suave, la sigmoidea de la figura 11.3.

**11.4 Adaptación de redes neuronales**

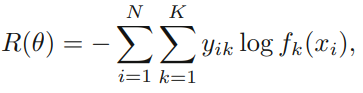
El modelo de red neuronal tiene parámetros desconocidos, a menudo llamados pesos, y buscamos valores para ellos que hagan que el modelo se ajuste bien a los datos de entrenamiento. Denotamos el conjunto completo de pesos por θ, que consta de



Para la regresión, usamos errores de suma de cuadrados como nuestra medida de ajuste (función de error)



Para la clasificación usamos el error al cuadrado o la entropía cruzada (desviación):

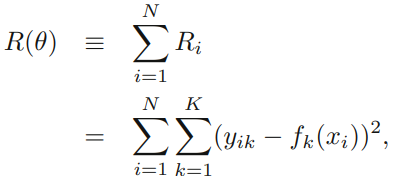


y el clasificador correspondiente es G (x) = argmaxkfk (x). Con la función de activación softmax y la función de error de entropía cruzada, el modelo de red neuronal es exactamente un modelo de regresión logística lineal en las unidades ocultas, y todos los parámetros se estiman por máxima verosimilitud.

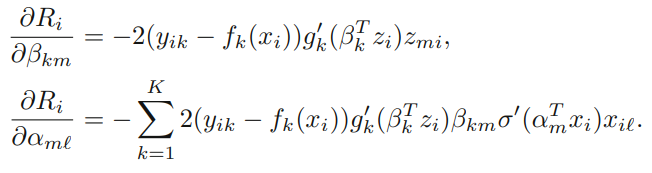
Por lo general, no queremos el minimizador global de R (θ), ya que es probable que sea una solución de sobreajuste. En cambio, se necesita cierta regularización: esto se logra directamente a través de un plazo de penalización, o indirectamente mediante una parada anticipada. Los detalles se dan en la siguiente sección.

El enfoque genérico para minimizar R (θ) es por descenso de gradiente, llamado retropropagación en este entorno. Debido a la forma de composición del modelo, el gradiente se puede derivar fácilmente usando la regla de la cadena para la diferenciación. Esto se puede calcular mediante un barrido hacia adelante y hacia atrás en la red, manteniendo un registro solo de las cantidades locales de cada unidad.

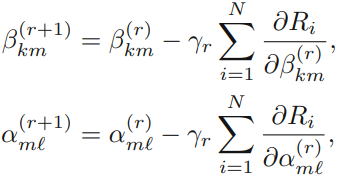
Aquí se muestra la propagación hacia atrás en detalle para la pérdida de error al cuadrado. Sea zmi = σ (α0m + αT mxi), de (11.5) y sea zi = (z1i, z2i, ..., zM i). Entonces tenemos



Con derivadas:

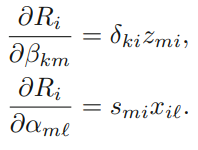


Dadas estas derivadas, una actualización de descenso de gradiente en la (r + 1) st iteración tiene la forma:

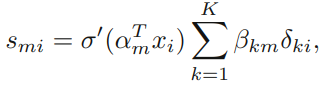


donde γr es la tasa de aprendizaje, que se analiza a continuación.

Ahora escriba (11.12) como:



Las cantidades δki y smi son "errores" del modelo actual en las unidades de capa oculta y de salida, respectivamente. De sus definiciones, estos errores satisfacen:



conocidas como ecuaciones de retropropagación. Con esto, las actualizaciones en (11.13) se pueden implementar con un algoritmo de dos pasos. En el pase hacia adelante, los pesos actuales son fijos y los valores predichos ˆfk (xi) se calculan a partir de la fórmula (11.5). En el paso hacia atrás, se calculan los errores δki y luego se propagan hacia atrás mediante (11.15) para dar los errores smi. Ambos conjuntos de errores se utilizan para calcular los gradientes de las actualizaciones en (11.13), a través de (11.14).

Este procedimiento de dos pasos es lo que se conoce como propagación hacia atrás. También se le ha llamado la regla delta (Widrow y Hoff, 1960). Los componentes computacionales de la entropía cruzada tienen la misma forma que los de la función de error de suma de cuadrados y se derivan en el ejercicio 11.3.

Las ventajas de la retropropagación son su naturaleza simple y local. En el algoritmo de retropropagación, cada unidad oculta pasa y recibe información solo hacia y desde unidades que comparten una conexión. Por lo tanto, se puede implementar de manera eficiente en una computadora de arquitectura paralela.

Las actualizaciones en (11.13) son una especie de aprendizaje por lotes, y las actualizaciones de los parámetros son una suma de todos los casos de entrenamiento. El aprendizaje también se puede llevar a cabo en línea: procesando cada observación de una en una, actualizando el gradiente después de cada caso de entrenamiento y recorriendo los casos de entrenamiento muchas veces. En este caso, las sumas de las ecuaciones (11.13) se reemplazan por un único sumando. Una época de entrenamiento se refiere a un barrido a través de todo el conjunto de entrenamiento. El entrenamiento en línea permite a la red manejar conjuntos de entrenamiento muy grandes y también actualizar los pesos a medida que ingresan nuevas observaciones.

La tasa de aprendizaje γr para el aprendizaje por lotes generalmente se toma como una constante y también se puede optimizar mediante una búsqueda de líneas que minimiza la función de error en cada actualización. Con el aprendizaje en línea, γr debería disminuir a cero a medida que la iteración r → ∞. Este aprendizaje es una forma de aproximación estocástica (Robbins y Munro, 1951); Los resultados en este campo aseguran la convergencia si γr → 0, r γr = ∞ y r γ2 r <∞ (satisfecho, por ejemplo, por γr = 1 / r).

La propagación hacia atrás puede ser muy lenta y, por esa razón, generalmente no es el método de elección. Las técnicas de segundo orden, como el método de Newton, no son atractivas aquí, porque la matriz de la segunda derivada de R (el hessiano) puede ser muy grande. Los mejores enfoques para el ajuste incluyen gradientes conjugados y métodos métricos variables. Éstos evitan el cálculo explícito de la segunda matriz derivada al tiempo que proporcionan una convergencia más rápida.

**11.5 Algunos problemas en el entrenamiento de redes neuronales**

Es todo un arte entrenar redes neuronales. El modelo generalmente está sobreparametrizado y el problema de optimización no es convexo e inestable a menos que se sigan ciertas pautas. En esta sección resumimos algunos de los temas importantes.

**11.5.1 Valores iniciales**

Tenga en cuenta que, si los pesos están cerca de cero, entonces la parte operativa del sigmoide (figura 11.3) es aproximadamente lineal y, por lo tanto, la red neuronal colapsa en un modelo aproximadamente lineal (ejercicio 11.2). Por lo general, los valores iniciales de los pesos se eligen para que sean valores aleatorios cercanos a cero. Por lo tanto, el modelo comienza casi lineal y se vuelve no lineal a medida que aumentan los pesos. Las unidades individuales se localizan según las direcciones e introducen no linealidades cuando es necesario. El uso de pesos cero exactos conduce a derivadas cero y simetría perfecta, y el algoritmo nunca se mueve. En cambio, comenzar con grandes pesos a menudo conduce a soluciones deficientes.

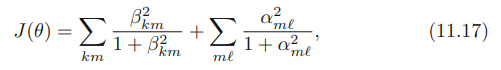
**11.5.2 Sobreajuste**

A menudo, las redes neuronales tienen demasiados pesos y sobreajustarán los datos en el mínimo global de R. En los primeros desarrollos de las redes neuronales, ya sea por diseño o por accidente, se utilizó una regla de detención temprana para evitar el sobreajuste. Aquí entrenamos el modelo solo por un tiempo y nos detenemos mucho antes de acercarnos al mínimo global. Dado que las ponderaciones comienzan en una solución altamente regularizada (lineal), esto tiene el efecto de reducir el modelo final hacia un modelo lineal. Un conjunto de datos de validación es útil para determinar cuándo detenerse, ya que esperamos que el error de validación comience a aumentar.

Un método más explícito para la regularización es la disminución del peso, que es análogo a la regresión de crestas que se usa para los modelos lineales (Sección 3.4.1). Agregamos una penalización a la función de error R (θ) + λJ (θ), donde:



y λ ≥ 0 es un parámetro de ajuste. Los valores más grandes de λ tenderán a reducir los pesos hacia cero: normalmente se utiliza la validación cruzada para estimar λ. El efecto de la penalización es simplemente agregar los términos 2βkm y 2αm a las respectivas expresiones de gradiente (11.13). Se han propuesto otras formas de sanción, por ejemplo,

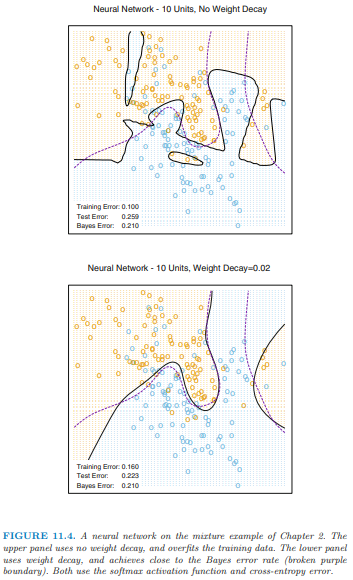


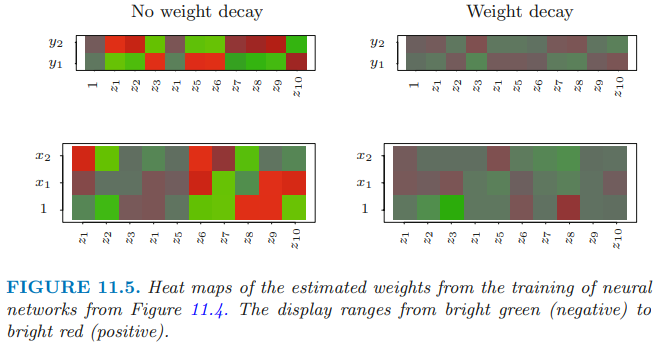
conocido como la penalización por eliminación de peso. Esto tiene el efecto de encoger pesos más pequeños más que (11.16).

La figura 11.4 muestra el resultado de entrenar una red neuronal con diez unidades ocultas, sin disminución de peso (panel superior) y con disminución de peso (panel inferior), al ejemplo de mezcla del Capítulo 2. La disminución de peso ha mejorado claramente la predicción. La figura 11.5 muestra mapas de calor de los pesos estimados del entrenamiento (las versiones en escala de grises de estos se denominan diagramas de Hinton). Vemos que la disminución del peso ha amortiguado los pesos en ambas capas: los pesos resultantes se distribuyen de manera bastante uniforme entre las diez unidades ocultas.

**11.5.3 Escalado de las entradas**

Dado que la escala de las entradas determina la escala efectiva de los pesos en la capa inferior, puede tener un gran efecto en la calidad de la solución final. Al principio, es mejor estandarizar todas las entradas para que tengan una media de cero y una desviación estándar de uno. Esto garantiza que todas las entradas se traten por igual en el proceso de regularización y permite elegir un rango significativo para los pesos iniciales aleatorios. Con entradas estandarizadas, es típico tomar pesos uniformes aleatorios en el rango [−0,7, +0,7].





**11.5.4 Número de capas y unidades ocultas**

En general, es mejor tener demasiadas unidades ocultas que muy pocas. Con muy pocas unidades ocultas, es posible que el modelo no tenga suficiente flexibilidad para capturar las no linealidades en los datos; con demasiadas unidades ocultas, los pesos adicionales se pueden reducir a cero si se utiliza la regularización adecuada. Por lo general, el número de unidades ocultas se encuentra en el rango de 5 a 100, y el número aumenta con el número de entradas y el número de casos de entrenamiento. Es más común colocar un número razonablemente grande de unidades y entrenarlas con regularización. Algunos investigadores usan la validación cruzada para estimar el número óptimo, pero esto parece innecesario si se usa la validación cruzada para estimar el parámetro de regularización. La elección del número de capas ocultas se basa en el conocimiento previo y la experimentación. Cada capa extrae características de la entrada para regresión o clasificación. El uso de múltiples capas ocultas permite la construcción de características jerárquicas en diferentes niveles de resolución. En la Sección 11.6 se ofrece un ejemplo del uso eficaz de varias capas.

**11.5.5 Mínimos múltiples**

La función de error R (θ) no es convexa y posee muchos mínimos locales. Como resultado, la solución final obtenida depende en gran medida de la elección de los pesos iniciales. Uno debe probar al menos una serie de configuraciones iniciales aleatorias y elegir la solución que dé el error más bajo (penalizado). Probablemente un mejor enfoque es usar las predicciones promedio sobre la colección de redes como predicción final (Ripley, 1996). Esto es preferible a promediar los pesos, ya que la no linealidad del modelo implica que esta solución promediada podría ser bastante pobre. Otro enfoque es el ensacado, que promedia las predicciones del entrenamiento de redes a partir de versiones perturbadas aleatoriamente de los datos de entrenamiento. Esto se describe en la Sección 8.7.