

CAPÍTULO 16. MODELOS PROBABILÍSTICOS ESTRUCTURADOS PARA EL APRENDIZAJE PROFUNDO

por lo general, no es muy fácil de interpretar para un ser humano después del hecho, aunque las técnicas de visualización pueden permitir una caracterización aproximada de lo que representan. Cuando las variables latentes se utilizan en el contexto de los modelos gráficos tradicionales, a menudo se diseñan teniendo en cuenta una semántica específica: el tema de un documento, la inteligencia de un estudiante, la enfermedad que causa los síntomas de un paciente, etc. más interpretables por los profesionales humanos y, a menudo, tienen más garantías teóricas, pero son menos capaces de escalar a problemas complejos y no son reutilizables en tantos contextos diferentes como los modelos profundos.

Otra diferencia obvia es el tipo de conectividad que normalmente se usa en el enfoque de aprendizaje profundo. Los modelos gráficos profundos suelen tener grandes grupos de unidades que están todas conectadas a otros grupos de unidades, de modo que las interacciones entre dos grupos pueden describirse mediante una sola matriz. Los modelos gráficos tradicionales tienen muy pocas conexiones y la elección de las conexiones para cada variable puede diseñarse individualmente. El diseño de la estructura del modelo está estrechamente relacionado con la elección del algoritmo de inferencia. Los enfoques tradicionales de los modelos gráficos suelen tener como objetivo mantener la manejabilidad de la inferencia exacta. Cuando esta restricción es demasiado limitante, un algoritmo de inferencia aproximada popular es un algoritmo llamado **propagación de creencias locas**. Ambos enfoques a menudo funcionan bien con gráficos muy escasamente conectados. En comparación, los modelos utilizados en el aprendizaje profundo tienden a conectar cada unidad visible vía muchas unidades ocultas h_i , de modo que puede proporcionar una representación distribuida de v (y probablemente varias otras variables observadas también). Las representaciones distribuidas tienen muchas ventajas, pero desde el punto de vista de los modelos gráficos y la complejidad computacional, las representaciones distribuidas tienen la desventaja de que generalmente producen gráficos que no son lo suficientemente dispersos para que las técnicas tradicionales de inferencia exacta y propagación de creencias locas sean relevantes. Como consecuencia, una de las diferencias más llamativas entre la comunidad de modelos gráficos más grandes y la comunidad de modelos gráficos profundos es que la propagación de creencias descabelladas casi nunca se usa para el aprendizaje profundo. En cambio, la mayoría de los modelos profundos están diseñados para hacer que el muestreo de Gibbs o los algoritmos de inferencia variacional sean eficientes. Otra consideración es que los modelos de aprendizaje profundo contienen una gran cantidad de variables latentes, lo que hace que el código numérico eficiente sea esencial. Esto proporciona una motivación adicional, además de la elección del algoritmo de inferencia de alto nivel, para agrupar las unidades en capas con una matriz que describe la interacción entre dos capas. Esto permite que los pasos individuales del algoritmo se implementen con operaciones eficientes de productos de matrices o generalizaciones escasamente conectadas, como productos de matrices diagonales en bloque o convoluciones.

Finalmente, el enfoque de aprendizaje profundo para el modelado gráfico se caracteriza por una marcada tolerancia a lo desconocido. En lugar de simplificar el modelo hasta que se puedan calcular exactamente todas las cantidades que queríamos, aumentamos la potencia de

el modelo hasta que apenas sea posible entrenarlo o usarlo. A menudo usamos modelos cuyas distribuciones marginales no se pueden calcular y nos conformamos simplemente con extraer muestras aproximadas de estos modelos. A menudo entrenamos modelos con una función objetivo intratable que ni siquiera podemos aproximar en un período de tiempo razonable, pero aún podemos entrenar aproximadamente el modelo si podemos obtener de manera eficiente una estimación del gradiente de dicha función. El enfoque de aprendizaje profundo a menudo consiste en averiguar cuál es la cantidad mínima de información que necesitamos absolutamente y luego descubrir cómo obtener una aproximación razonable de esa información lo más rápido posible.

16.7.1 Ejemplo: La máquina de Boltzmann restringida

El **máquina de Boltzmann restringida**(GRB) ([Smolensky, 1986](#)) o **armonio** es el ejemplo por excelencia de cómo se utilizan los modelos gráficos para el aprendizaje profundo. El RBM no es en sí mismo un modelo profundo. En cambio, tiene una sola capa de variables latentes que se pueden usar para aprender una representación de la entrada. en el [capítulo 20](#), veremos cómo se pueden usar los RBM para construir muchos modelos más profundos. Aquí, mostramos cómo el RBM ejemplifica muchas de las prácticas utilizadas en una amplia variedad de modelos gráficos profundos: sus unidades están organizadas en grandes grupos llamados capas, la conectividad entre capas se describe mediante una matriz, la conectividad es relativamente densa, el modelo está diseñado para permitir un muestreo eficiente de Gibbs, y el énfasis del diseño del modelo está en liberar el algoritmo de entrenamiento para aprender variables latentes cuya semántica no fue especificada por el diseñador. Posteriormente, en la [sección 20.2](#), revisaremos la RBM con más detalle.

El RBM canónico es un modelo basado en energía con unidades binarias visibles y ocultas. Su función energética es

$$m(v, h) = -b \cdot v - c \cdot h - v \cdot h \quad (16.10)$$

dónde b , C , y W son parámetros aprendibles, de valor real y sin restricciones. Podemos ver que el modelo se divide en dos grupos de unidades: v y h , y la interacción entre ellos se describe mediante una matriz W . El modelo se representa gráficamente en la figura [16.14](#). Como deja en claro esta figura, un aspecto importante de este modelo es que no hay interacciones directas entre dos unidades visibles o entre dos unidades ocultas (por lo tanto, la "restringida", una máquina general de Boltzmann puede tener conexiones arbitrarias).

Las restricciones en la estructura RBM producen buenas propiedades

$$p_{\text{ag}}(h/v) = \prod_i p_{\text{ag}}(h_i/v) \quad (16.11)$$

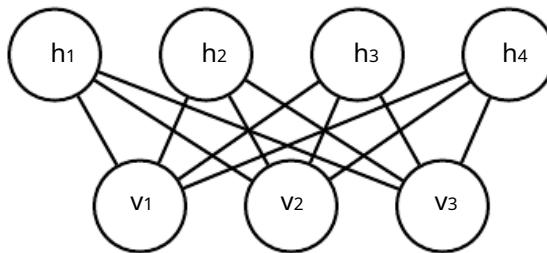


Figura 16.14: Una RBM dibujada como una red de Markov.

y

$$pag(v/h) = \prod_i pag(v_i/h). \quad (16.12)$$

Los condicionales individuales también son fáciles de calcular. Para el RBM binario obtenemos:

$$PAG(h=1/v) = \sigma v \cdot W_{:,i} + b_i, \quad (16.13)$$

$$PAG(h=0/v) = 1 - \sigma v \cdot W_{:,i} + b_i. \quad (16.14)$$

Juntas, estas propiedades permiten una eficiente **bloque gibbs** muestreo, que alterna entre el muestreo de todos los *h*s simultáneamente y muestreando todos los *v*s simultáneamente. Las muestras generadas por el muestreo de Gibbs a partir de un modelo RBM se muestran en la figura 16.15.

Dado que la función de energía en sí misma es solo una función lineal de los parámetros, es fácil obtener sus derivadas. Por ejemplo,

$$\frac{\partial}{\partial W_{yo,j}} m(v,h) = -vh_j. \quad (16.15)$$

Estas dos propiedades, el muestreo eficiente de Gibbs y las derivadas eficientes, hacen que el entrenamiento sea conveniente. En el capítulo 18, veremos que los modelos no dirigidos pueden entrenarse calculando dichas derivadas aplicadas a muestras del modelo.

El entrenamiento del modelo induce una representación *h*de los datos *v*. A menudo podemos usar $m_{ih} \sim pag(h/v)[h]$ como un conjunto de características para describir *v*.

En general, el RBM demuestra el enfoque típico de aprendizaje profundo para modelos gráficos: aprendizaje de representación logrado a través de capas de variables latentes, combinado con interacciones eficientes entre capas parametrizadas por matrices.

El lenguaje de los modelos gráficos proporciona un lenguaje elegante, flexible y claro para describir modelos probabilísticos. En los capítulos siguientes, usamos este lenguaje, entre otras perspectivas, para describir una amplia variedad de modelos probabilísticos profundos.



Figura 16.15: Muestras de un RBM entrenado y sus pesos. Imagen reproducida con permiso de [LISA \(2008\)](#). (Izquierda) Muestras de un modelo entrenado en MNIST, extraídas mediante el muestreo de Gibbs. Cada columna es un proceso de muestreo de Gibbs separado. Cada fila representa la salida de otros 1000 pasos de muestreo de Gibbs. Las muestras sucesivas están altamente correlacionadas entre sí. (Derecha) Los vectores de peso correspondientes. Compare esto con las muestras y los pesos de un modelo de factor lineal, que se muestra en la figura 13.2. Las muestras aquí son mucho mejores porque el RBM anterior $p_{\theta}(h)$ no está obligado a ser factorial. El RBM puede aprender qué características deben aparecer juntas al muestrear. Por otro lado, el RBM posterior $p_{\theta}(h | v)$ es factorial, mientras que la escasa codificación posterior $p_{\theta}(h | v)$ no lo es, por lo que el modelo de codificación dispersa puede ser mejor para la extracción de características. Otros modelos pueden tener un factor no factorial $p_{\theta}(h)$ y no factorial $p_{\theta}(h | v)$.

capítulo 17

Métodos de Montecarlo

Los algoritmos aleatorios se dividen en dos categorías aproximadas: algoritmos de Las Vegas y algoritmos de Monte Carlo. Los algoritmos de Las Vegas siempre devuelven con precisión la respuesta correcta (o informan que fallaron). Estos algoritmos consumen una cantidad aleatoria de recursos, generalmente memoria o tiempo. Por el contrario, los algoritmos de Monte Carlo devuelven respuestas con una cantidad aleatoria de error. La cantidad de error normalmente se puede reducir gastando más recursos (generalmente tiempo de ejecución y memoria). Para cualquier presupuesto computacional fijo, un algoritmo de Monte Carlo puede proporcionar una respuesta aproximada.

Muchos problemas en el aprendizaje automático son tan difíciles que nunca podemos esperar obtener respuestas precisas para ellos. Esto excluye los algoritmos deterministas precisos y los algoritmos de Las Vegas. En su lugar, debemos usar algoritmos aproximados deterministas o aproximaciones de Monte Carlo. Ambos enfoques son omnipresentes en el aprendizaje automático. En este capítulo, nos centramos en los métodos de Monte Carlo.

17.1 Métodos de muestreo y Monte Carlo

Muchas tecnologías importantes que se utilizan para lograr los objetivos de aprendizaje automático se basan en extraer muestras de alguna distribución de probabilidad y usar estas muestras para formar una estimación de Monte Carlo de alguna cantidad deseada.

17.1.1 ¿Por qué tomar muestras?

Hay muchas razones por las que podemos desear extraer muestras de una distribución de probabilidad. El muestreo proporciona una forma flexible de aproximar muchas sumas y

integrales a costo reducido. A veces usamos esto para proporcionar una aceleración significativa a una suma costosa pero manejable, como en el caso de submuestrear el costo total de capacitación con minilotes. En otros casos, nuestro algoritmo de aprendizaje requiere que nos aproximemos a una suma o integral intratable, como el gradiente de la función de partición logarítmica de un modelo no dirigido. En muchos otros casos, el muestreo es en realidad nuestro objetivo, en el sentido de que queremos entrenar un modelo que pueda tomar muestras de la distribución de entrenamiento.

17.1.2 Conceptos básicos del muestreo Monte Carlo

Cuando una suma o una integral no se pueden calcular exactamente (por ejemplo, la suma tiene un número exponencial de términos y no se conoce una simplificación exacta), a menudo es posible aproximarla utilizando el muestreo de Monte Carlo. La idea es ver la suma o integral como si fuera una expectativa bajo alguna distribución y *aproximar la expectativa por un promedio correspondiente*. Dejar

$$S = \frac{1}{n} \sum_{i=1}^n pag(X_i) F(X_i) = \text{mipag}[F(X)] \quad (17.1)$$

O

$$S = \frac{1}{n} \sum_{i=1}^n pag(X_i) F(X_i) dX_i = \text{mipag}[F(X)] \quad (17.2)$$

sea S la suma o integral a estimar, reescrita como una expectativa, con la restricción de que $pages$ una distribución de probabilidad (para la suma) o una densidad de probabilidad (para la integral) sobre una variable aleatoria X .

Podemos aproximar S dibujando n muestras $X(1), \dots, X(n)$ de $page$ y luego formando el promedio empírico

$$S_{\text{norte}} = \frac{1}{n} \sum_{i=1}^n F(X(i)). \quad (17.3)$$

Esta aproximación se justifica por algunas propiedades diferentes. La primera observación trivial es que el estimador es imparcial, ya que

$$\text{MI}[S_{\text{norte}}] = \frac{1}{n} \sum_{i=1}^n \text{MI}[F(X(i))] = \frac{1}{n} \sum_{i=1}^n S = S. \quad (17.4)$$

Pero además, el **ley de los grandes números** establece que si las muestras $X(i)$ son iid, entonces el promedio converge casi con seguridad al valor esperado:

$$\lim_{n \rightarrow \infty} S_{\text{norte}} = S, \quad (17.5)$$

siempre que la variación de los términos individuales, $\text{Var}[F(X_{(j)})]$, está ligado. Para ver esto más claramente, considere la varianza de s_{norte} como n_{norte} aumenta la varianza $\text{Var}[s_{\text{norte}}]$ decrece y converge a 0, siempre que $\text{Var}[F(X_{(j)})] < \infty$:

$$\text{Var}[s_{\text{norte}}] = \frac{1}{n_{\text{norte}}} - \frac{\text{Var}[F(X)]}{n_{\text{norte}}^2} \quad (17.6)$$

$$= \frac{\text{Var}[F(X)]}{n_{\text{norte}}} \quad (17.7)$$

Este conveniente resultado también nos dice cómo estimar la incertidumbre en un promedio de Monte Carlo o, de manera equivalente, la cantidad de error esperado de la aproximación de Monte Carlo. Calculamos tanto el promedio empírico de los $F(X_{(j)})$ y su varianza empírica,¹ y luego dividir la varianza estimada por el número de muestras n_{norte} obtener un estimador de $\text{Var}[s_{\text{norte}}]$. El **teorema del límite central** nos dice que la distribución de la media, s_{norte} , converge a una distribución normal con media s_{norte} y varianza $\text{Var}[F(X)]$. Esto nos permite estimar intervalos de confianza alrededor de la estimación s_{norte} , utilizando la distribución acumulativa de la densidad normal.

Sin embargo, todo esto depende de nuestra capacidad para muestrear fácilmente desde la distribución base, $p_{\text{ag}}(X)$, pero hacerlo no siempre es posible. Cuando no es factible tomar muestras de p_{ag} , una alternativa es utilizar el muestreo por importancia, presentado en la sección 17.2. Un enfoque más general es formar una secuencia de estimadores que converjan hacia la distribución de interés. Ese es el enfoque de las cadenas de Monte Carlo Markov (sección 17.3).

17.2 Muestreo de importancia

Un paso importante en la descomposición del integrando (o sumando) utilizado por el método de Monte Carlo en la ecuación 17.2 es decidir qué parte del integrando debe desempeñar el papel de la probabilidad $p_{\text{ag}}(X)$ y qué parte del integrando debe desempeñar el papel de la cantidad $F(X)$ cuyo valor esperado (bajo esa distribución de probabilidad) se va a estimar. No hay descomposición única porque $p_{\text{ag}}(X)F(X)$ siempre puede ser reescrito como

$$p_{\text{ag}}(X)F(X) = q(X) \frac{p_{\text{ag}}(X)F(X)}{q(X)}, \quad (17.8)$$

de donde ahora muestreamos q y promedio p_{ag}/q . En muchos casos, deseamos calcular una expectativa para un dado p_{ag} y F , y el hecho de que el problema se especifica

¹A menudo se prefiere el estimador insesgado de la varianza, en el que la suma de las diferencias al cuadrado se divide por $n_{\text{norte}} - 1$ en lugar de n_{norte} .

desde el principio como una expectativa sugiere que este \hat{p}_{avg} sería una elección natural de descomposición. Sin embargo, la especificación original del problema puede no ser la elección óptima en términos del número de muestras requeridas para obtener un nivel de precisión dado. Afortunadamente, la forma de la elección óptima q^* se puede derivar fácilmente. lo óptimo q^* corresponde a lo que se denomina muestreo de importancia óptima.

Debido a la identidad mostrada en la ecuación 17.8, cualquier estimador Monte Carlo

$$\hat{p}_{\text{avg}} = \frac{1}{n} \sum_{i=1, X(i) \sim p}^{n} f(X(i)) \quad (17.9)$$

se puede transformar en un estimador de muestreo de importancia

$$\hat{s}_q = \frac{1}{n} \sum_{i=1, X(i) \sim q}^{n} \frac{p(X(i)) f(X(i))}{q(X(i))} \quad (17.10)$$

Vemos fácilmente que el valor esperado del estimador no depende de q :

$$\text{mi}_q[\hat{s}_q] = \text{mi}_q[\hat{p}_{\text{avg}}] = s. \quad (17.11)$$

Sin embargo, la varianza de un estimador de muestreo de importancia puede ser muy sensible a la elección de q . La varianza está dada por

$$\text{Var}[\hat{s}_q] = \text{Var}\left[\frac{p(X) f(X)}{q(X)} \right] \quad (17.12)$$

La varianza mínima ocurre cuando q es

$$q^*(X) = \frac{p(X) / f(X)}{Z}, \quad (17.13)$$

dónde Z es la constante de normalización, elegida de modo que $q^*(X)$ sumas o integra a 1 según sea apropiado. Las distribuciones de muestreo de mejor importancia ponen más peso donde el integrando es más grande. De hecho, cuando $f(X)$ no cambia de signo, $\text{Var}[\hat{s}_q] = 0$, significa que *una sola muestra es suficiente* cuando se utiliza la distribución óptima. Por supuesto, esto es sólo porque el cálculo de q^* esencialmente ha resuelto el problema original, por lo que generalmente no es práctico utilizar este enfoque de extraer una sola muestra de la distribución óptima.

Cualquier elección de distribución de muestreo q es válido (en el sentido de producir el valor esperado correcto) y q^* es el óptimo (en el sentido de producir una varianza mínima). Muestreo de q^* suele ser inviable, pero otras opciones de q puede ser factible y al mismo tiempo reducir un poco la varianza.

Otro enfoque es utilizar **muestreo de importancia sesgada**, que tiene la ventaja de no requerir normalizado p_{avg} . En el caso de variables discretas, el estimador de muestreo de importancia sesgada viene dado por

$$\hat{s}^{\text{BIS}} = \frac{-\sum_{i=1}^n \frac{p_{\text{avg}}(X(i)) f(X(i)) q(i)}{q(X(i))}}{\sum_{i=1}^n \frac{\text{notario público}(X(i))}{q(X(i))}} \quad (17.14)$$

$$= \frac{-\sum_{i=1}^n \frac{p_{\text{avg}}(X(i)) f(X(i)) \tilde{q}(i)}{\tilde{q}(X(i))}}{\sum_{i=1}^n \frac{\text{notario público}(X(i))}{\tilde{q}(X(i))}} \quad (17.15)$$

$$= \frac{-\sum_{i=1}^n \frac{p_{\text{avg}}(X(i)) f(X(i))}{\tilde{q}(X(i))}}{\sum_{i=1}^n \frac{n \tilde{p}(X(i))}{\tilde{q}(X(i))}} \quad (17.16)$$

dónde p_{avg} y \tilde{q} son las formas no normalizadas de p_{avg} y q el $X(i)$ son las muestras de q . Este estimador está sesgado porque $\text{MI}[\hat{s}_{\text{BIS}}] = -s$, excepto asintóticamente cuando $norte \rightarrow \infty$ y el denominador de la ecuación 17.14 converge a 1. Por lo tanto, este estimador se llama asintóticamente insesgado.

Aunque es una buena elección de q puede mejorar en gran medida la eficiencia de la estimación de Monte Carlo, una mala elección de q puede hacer que la eficiencia sea mucho peor. Volviendo a la ecuación 17.12, vemos que si hay muestras de q para las cuales $p_{\text{avg}}(X)/f(X)$ es grande, entonces la varianza del estimador puede ser muy grande. Esto puede suceder cuando $q(X)$ es diminuto mientras que ninguno $p_{\text{avg}}(X)$ ni $f(X)$ son lo suficientemente pequeños como para cancelarlo. El q La distribución generalmente se elige para que sea una distribución muy simple para que sea fácil de muestrear. Cuando X es de alta dimensión, esta simplicidad en q hace que coincida p_{avg}/f mal. Cuando $q(X(i)) - p_{\text{avg}}(X(i))/f(X(i))$, el muestreo de importancia recopila muestras inútiles (suma de números pequeños o ceros). Por otro lado, cuando $q(X(i)) - p_{\text{avg}}(X(i))/f(X(i))$, lo que ocurrirá con menos frecuencia, la proporción puede ser enorme. Debido a que estos últimos eventos son raros, es posible que no aparezcan en una muestra típica, lo que produce una subestimación típica des, compensado rara vez por una gran sobreestimación. Tales números muy grandes o muy pequeños son típicos cuando X es de alta dimensión, porque en alta dimensión el rango dinámico de probabilidades conjuntas puede ser muy grande.

A pesar de este peligro, el muestreo de importancia y sus variantes se han encontrado muy útiles en muchos algoritmos de aprendizaje automático, incluidos los algoritmos de aprendizaje profundo. Por ejemplo, vea el uso del muestreo de importancia para acelerar el entrenamiento en modelos de lenguaje neuronal con un vocabulario amplio (sección 12.4.3.3) u otras redes neuronales con un gran número de salidas. Vea también cómo se ha utilizado el muestreo de importancia para estimar una función de partición (la constante de normalización de una probabilidad

distribución) en la sección 18.7, y para estimar la verosimilitud logarítmica en modelos dirigidos profundos como el autocodificador variacional, en la sección 20.10.3. El muestreo de importancia también se puede utilizar para mejorar la estimación del gradiente de la función de costo utilizada para entrenar los parámetros del modelo con descenso de gradiente estocástico, particularmente para modelos como los clasificadores donde la mayor parte del valor total de la función de costo proviene de un pequeño número de errores clasificados. ejemplos El muestreo de ejemplos más difíciles con mayor frecuencia puede reducir la varianza del gradiente en tales casos (Hinton, 2006).

17.3 Métodos Monte Carlo de la cadena de Markov

En muchos casos, deseamos utilizar una técnica de Monte Carlo, pero no existe un método manejable para extraer muestras exactas de la distribución $p_{\text{modelo}}(X)$ o de una buena distribución de muestreo de importancia (baja varianza) $q(X)$. En el contexto del aprendizaje profundo, esto sucede con mayor frecuencia cuando $p_{\text{modelo}}(X)$ está representado por un modelo no dirigido. En estos casos, introducimos una herramienta matemática llamada **cadena de Markov** a una muestra aproximada de $p_{\text{modelo}}(X)$. La familia de algoritmos que utilizan cadenas de Markov para realizar estimaciones de Monte Carlo se llama **Métodos de Monte Carlo de la cadena de Markov** (MCMC). Los métodos de cadena de Markov Monte Carlo para el aprendizaje automático se describen con más detalle en Koller y Friedman (2009). Las garantías genéricas más estándar para las técnicas MCMC solo son aplicables cuando el modelo no asigna probabilidad cero a ningún estado. Por lo tanto, es más conveniente presentar estas técnicas como muestreo de un modelo basado en energía (EBM) $p_{\text{ag}}(X) \propto \text{Exp}(-E(X))$ como se describe en la sección 16.2.4. En la formulación de EBM, se garantiza que cada estado tiene una probabilidad distinta de cero. De hecho, los métodos MCMC tienen una aplicación más amplia y se pueden usar con muchas distribuciones de probabilidad que contienen estados de probabilidad cero. Sin embargo, las garantías teóricas sobre el comportamiento de los métodos MCMC deben probarse caso por caso para diferentes familias de tales distribuciones. En el contexto del aprendizaje profundo, lo más común es confiar en las garantías teóricas más generales que se aplican naturalmente a todos los modelos basados en energía.

Para entender por qué es difícil extraer muestras de un modelo basado en energía, considere un EBM sobre solo dos variables, definiendo una distribución $p_{\text{ag}}(a,b)$. Para muestrear a , debemos extraer a de $p_{\text{ag}}(a/b)$, y para muestrear b , debemos extraerlo de $p_{\text{ag}}(b/a)$. Parece ser un problema intratable del huevo y la gallina. Los modelos dirigidos evitan esto porque su gráfico es dirigido y acíclico. Actuar **muestreo ancestral** uno simplemente muestrea cada una de las variables en orden topológico, condicionando a los padres de cada variable, que se garantiza que ya han sido muestreados (sección 16.3). El muestreo ancestral define un método eficiente de un solo paso

de obtener una muestra.

En un EBM, podemos evitar este problema del huevo y la gallina muestreando usando una cadena de Markov. La idea central de una cadena de Markov es tener un estado X que comienza como un valor arbitrario. Con el tiempo, actualizamos aleatoriamente X repetidamente. Eventualmente X se convierte (casi) en una buena muestra de $\text{pag}(X)$. Formalmente, una cadena de Markov se define por un estado aleatorio X y una distribución de transición $\pi(X_{-}/X)$ especificando la probabilidad de que una actualización aleatoria pase al estado X si comienza en el estado X . Ejecutar la cadena de Markov significa actualizar repetidamente el estado X a un valor X -muestreado de $\pi(X_{-}/X)$.

Para obtener una comprensión teórica de cómo funcionan los métodos MCMC, es útil repararmetrizar el problema. Primero, restringimos nuestra atención al caso donde la variable aleatoria X tiene muchos estados contables. Entonces podemos representar el estado como un entero positivo X . Diferentes valores enteros de X mapan de nuevo a diferentes estados X en el problema original.

Considere lo que sucede cuando ejecutamos infinitas cadenas de Markov en paralelo. Todos los estados de las diferentes cadenas de Markov se extraen de alguna distribución $q(t)(X)$, donde t indica el número de pasos de tiempo que han transcurrido. Al principio, $q(0)$ es una distribución que usamos para inicializar arbitrariamente X para cada cadena de Markov. Más tarde, $q(t)$ está influenciado por todos los pasos de la cadena de Markov que se han ejecutado hasta ahora. Nuestro objetivo es para $q(t)(X)$ converger a $\text{pag}(X)$.

Porque hemos reparametrizado el problema en términos de entero positivo X , podemos describir la distribución de probabilidad q usando un vector v , con

$$q(x=i) = v_i. \quad (17.17)$$

Considere lo que sucede cuando actualizamos el estado de una sola cadena de Markov X a un nuevo estado X . La probabilidad de que un solo estado aterrice en el estado X es dado por

$$q(t+1)(X) = \sum_X q(t)(X) \pi(X_{-}/X). \quad (17.18)$$

Usando nuestra parametrización de enteros, podemos representar el efecto del operador de transición π utilizando una matriz A . Definimos A de modo que

$$A_{yo,j} = \pi(X=yo/x=j). \quad (17.19)$$

Usando esta definición, ahora podemos reescribir la ecuación 17.18. En lugar de escribirlo en términos de q y π para entender cómo se actualiza un solo estado, ahora podemos usar v y A para describir cómo cambia la distribución completa sobre todas las diferentes cadenas de Markov (que se ejecutan en paralelo) a medida que aplicamos una actualización:

$$v(t) = A v(t-1). \quad (17.20)$$

Aplicar la actualización de la cadena de Markov repetidamente corresponde a multiplicar por la matriz A repetidamente. En otras palabras, podemos pensar en el proceso como una exponenciación de la matriz A :

$$\nu_t = A^t \nu_0. \quad (17.21)$$

La matriz A tiene una estructura especial porque cada una de sus columnas representa una distribución de probabilidad. Tales matrices se llaman **matrices estocásticas**. Si hay una probabilidad distinta de cero de pasar de cualquier estado X a cualquier otro estado X' por algo de t , entonces el teorema de Perron-Frobenius ([Escalinata, 1907; Frobenius, 1908](#)) garantiza que el mayor valor propio es real e igual a 1. Con el tiempo, podemos ver que todos los valores propios están exponenciados:

$$\nu_t = V \text{diag}(\lambda) V^{-1} \nu_0 = V \text{diag}(\lambda)^t V^{-1} \nu_0. \quad (17.22)$$

Este proceso hace que todos los valores propios que no son iguales a 1 decaden a cero. Bajo algunas condiciones suaves adicionales, A garantiza que tiene solo un vector propio con valor propio 1. El proceso converge así a una **distribución estacionaria**, a veces también llamado **distribución de equilibrio**. En la convergencia,

$$\nu = A\nu = \nu, \quad (17.23)$$

y esta misma condición se cumple para cada paso adicional. Esta es una ecuación de vector propio. Para ser un punto estacionario, ν debe ser un vector propio con valor propio correspondiente 1. Esta condición garantiza que una vez que hayamos alcanzado la distribución estacionaria, las aplicaciones repetidas del procedimiento de muestreo de transición no cambien la distribución sobre los estados de todas las diversas cadenas de Markov (aunque el operador de transición cambia cada estado individual, por supuesto).

Si hemos elegido T correctamente, entonces la distribución estacionaria π será igual a la distribución π que queremos muestrear. Describiremos cómo elegir T en breve, en la sección [17.4](#).

La mayoría de las propiedades de las cadenas de Markov con estados contables se pueden generalizar a variables continuas. En esta situación, algunos autores llaman a la Cadena de Markov una **cadena harris** pero usamos el término Cadena de Markov para describir ambas condiciones. En general, una cadena de Markov con operador de transición T convergerá, en condiciones suaves, a un punto fijo descrito por la ecuación

$$q(X-) = \sum_{X'} \pi(X-/X), \quad (17.24)$$

que en el caso discreto es simplemente reescribir la ecuación [17.23](#). Cuando X es discreta, la expectativa corresponde a una suma, y cuando X es continua, la expectativa corresponde a una integral.

Independientemente de si el estado es continuo o discreto, todos los métodos de la cadena de Markov consisten en aplicar repetidamente actualizaciones estocásticas hasta que finalmente el estado comienza a producir muestras de la distribución de equilibrio. Correr la cadena de Markov hasta que alcanza su distribución de equilibrio se llama “**ardiendo en**” la cadena de Markov. Una vez que la cadena ha alcanzado el equilibrio, se puede extraer una secuencia de infinitas muestras de la distribución de equilibrio. Están distribuidas de manera idéntica, pero dos muestras sucesivas cualesquiera estarán altamente correlacionadas entre sí. Por lo tanto, una secuencia finita de muestras puede no ser muy representativa de la distribución de equilibrio. Una forma de mitigar este problema es devolver solo cada *n* de las muestras sucesivas, de modo que nuestra estimación de las estadísticas de la distribución de equilibrio no esté tan sesgada por la correlación entre una muestra de MCMC y las siguientes muestras. Las cadenas de Markov son, por lo tanto, costosas de usar debido al tiempo requerido para quemar la distribución de equilibrio y el tiempo requerido para pasar de una muestra a otra muestra razonablemente decorrelacionada después de alcanzar el equilibrio. Si uno desea muestras verdaderamente independientes, puede ejecutar varias cadenas de Markov en paralelo. Este enfoque utiliza computación paralela adicional para eliminar la latencia. La estrategia de usar solo una cadena de Markov para generar todas las muestras y la estrategia de usar una cadena de Markov para cada muestra deseada son dos extremos; los profesionales del aprendizaje profundo suelen utilizar una cantidad de cadenas similar a la cantidad de ejemplos en un minilote y luego extraen tantas muestras como sea necesario de este conjunto fijo de cadenas de Markov. Un número comúnmente usado de cadenas de Markov es 100.

Otra dificultad es que no sabemos de antemano cuántos pasos debe recorrer la cadena de Markov antes de alcanzar su distribución de equilibrio. Este lapso de tiempo se denomina **tiempo de mezcla**. También es muy difícil probar si una cadena de Markov ha alcanzado el equilibrio. No tenemos una teoría lo suficientemente precisa para guiarnos en la respuesta a esta pregunta. La teoría nos dice que la cadena convergerá, pero no mucho más. Si analizamos la cadena de Markov desde el punto de vista de una matriz A actuando sobre un vector de probabilidades v , entonces sabemos que la cadena se mezcla cuando A^t ha perdido efectivamente todos los valores propios de A además del valor propio único de 1. Esto significa que la magnitud del segundo valor propio más grande determinará el tiempo de mezclado. Sin embargo, en la práctica, no podemos representar nuestra cadena de Markov en términos de una matriz. El número de estados que puede visitar nuestro modelo probabilístico es exponencialmente grande en el número de variables, por lo que es inviable representar v, A , o los valores propios de A . Debido a estos y otros obstáculos, normalmente no sabemos si una cadena de Markov se ha mezclado. En su lugar, simplemente ejecutamos la cadena de Markov durante un período de tiempo que estimamos aproximadamente suficiente y usamos métodos heurísticos para determinar si la cadena se ha mezclado. Estos métodos heurísticos incluyen la inspección manual de muestras o la medición de correlaciones entre

muestras sucesivas.

17.4 Muestreo de Gibbs

Hasta ahora hemos descrito cómo extraer muestras de una distribución $q(X)$ actualizando repetidamente $X \leftarrow X \sim \pi(X/X)$. Sin embargo, no hemos descrito cómo asegurar que $q(X)$ es una distribución útil. En este libro se consideran dos enfoques básicos. La primera es derivar π de un dado aprendido $p_{\text{model}}(X)$, se describe a continuación con el caso del muestreo de EBM. La segunda es parametrizar directamente π y aprenderlo, de modo que su distribución estacionaria defina implícitamente el $p_{\text{model}}(X)$ de interés. Los ejemplos de este segundo enfoque se analizan en las secciones [20.12](#) y [20.13](#).

En el contexto del aprendizaje profundo, comúnmente usamos cadenas de Markov para extraer muestras de un modelo basado en energía que define una distribución $p_{\text{model}}(X)$. En este caso, queremos que el $q(X)$ para que la cadena de Markov sea $p_{\text{model}}(X)$. Para obtener lo deseado $q(X)$, debemos elegir un adecuado $\pi(X/X)$.

Un enfoque conceptualmente simple y efectivo para construir una cadena de Markov que toma muestras de $p_{\text{model}}(X)$ es usar **Muestreo de Gibbs**, en el que el muestreo de $\pi(X/X)$ se logra seleccionando una variable x y tomar muestras de p_{model} condicionado a sus vecinos en el grafo no dirigido G_{RAM} definiendo la estructura del modelo basado en la energía. También es posible muestrear varias variables al mismo tiempo, siempre que sean condicionalmente independientes dadas todas sus vecinas. Como se muestra en el ejemplo de RBM en la sección [16.7.1](#), todas las unidades ocultas de un RBM se pueden muestrear simultáneamente porque son condicionalmente independientes entre sí dadas todas las unidades visibles. Asimismo, todas las unidades visibles pueden muestrearse simultáneamente porque son condicionalmente independientes entre sí dadas todas las unidades ocultas. Los enfoques de muestreo de Gibbs que actualizan muchas variables simultáneamente de esta manera se denominan **bloque de muestreo de Gibbs**.

Enfoques alternativos para el diseño de cadenas de Markov para tomar muestras de p_{model} es posible. Por ejemplo, el algoritmo Metropolis-Hastings se usa mucho en otras disciplinas. En el contexto del enfoque de aprendizaje profundo para el modelado no dirigido, es raro utilizar cualquier enfoque que no sea el muestreo de Gibbs. Las técnicas de muestreo mejoradas son una posible frontera de investigación.

17.5 El desafío de mezclar entre modos separados

La principal dificultad involucrada con los métodos MCMC es que tienen una tendencia a **mezcla** mal. Idealmente, muestras sucesivas de una cadena de Markov diseñada para muestrear

de $\text{pag}(X)$ serían completamente independientes entre sí y visitarían muchas regiones diferentes en el espacio proporcional a su probabilidad. En cambio, especialmente en casos de alta dimensión, las muestras de MCMC se vuelven muy correlacionadas. Nos referimos a este comportamiento como mezcla lenta o incluso falta de mezcla. Se puede considerar que los métodos MCMC con mezcla lenta realizan inadvertidamente algo parecido a un descenso de gradiente ruidoso en la función de energía, o una subida de colinas ruidosa equivalente en la probabilidad, con respecto al estado de la cadena (las variables aleatorias que se muestrean). La cadena tiende a dar pequeños pasos (en el espacio del estado de la cadena de Markov), a partir de una configuración $X_{(t-1)}$ a una configuración $X_{(t)}$, con la energía $m(X_{(t)})$ generalmente menor o aproximadamente igual a la energía $m(X_{(t-1)})$, con preferencia por movimientos que producen configuraciones de menor energía. Al partir de una configuración bastante improbable (mayor energía que las típicas $\text{depag}(X)$), la cadena tiende a reducir gradualmente la energía del estado y solo ocasionalmente pasa a otro modo. Una vez que la cadena ha encontrado una región de baja energía (por ejemplo, si las variables son píxeles en una imagen, una región de baja energía podría ser una variedad conectada de imágenes del mismo objeto), que llamamos modo, la cadena tienden a caminar alrededor de ese modo (siguiendo una especie de caminata aleatoria). De vez en cuando saldrá de ese modo y generalmente volverá a él o (si encuentra una ruta de escape) se moverá hacia otro modo. El problema es que las rutas de escape exitosas son raras para muchas distribuciones interesantes, por lo que la cadena de Markov continuará probando el mismo modo por más tiempo del que debería.

Esto es muy claro cuando consideramos el algoritmo de muestreo de Gibbs (sección 17.4). En este contexto, considere la probabilidad de pasar de un modo a un modo cercano dentro de un número dado de pasos. Lo que determinará esa probabilidad es la forma de la “barrera de energía” entre estos modos. Las transiciones entre dos modos que están separados por una barrera de alta energía (una región de baja probabilidad) son exponencialmente menos probables (en términos de la altura de la barrera de energía). Esto se ilustra en la figura 17.1. El problema surge cuando hay múltiples modas con alta probabilidad que están separadas por regiones de baja probabilidad, especialmente cuando cada paso de muestreo de Gibbs debe actualizar solo un pequeño subconjunto de variables cuyos valores están determinados en gran medida por las otras variables.

Como ejemplo simple, considere un modelo basado en energía sobre dos variables a y b, que son ambos binarios con signo, tomando valores -1 y 1. Si $m(a, b) = -wab$ para algún número positivo grande w , entonces el modelo expresa una fuerte creencia de que a y b tienen el mismo signo. Considere actualizar b usando un paso de muestreo de Gibbs con $a = 1$. La distribución condicional sobre b está dada por $PAG(b = 1 / a = 1) = \sigma(w)$. Si w es grande, el sigmoide se satura, y la probabilidad de asignar también a b 1 está cerca de 1. Asimismo, si $a = -1$, la probabilidad de asignar b a ser -1 está cerca de 1. Según $PAG_{\text{modelo}}(a, b)$, ambos signos de ambas variables son igualmente probables.

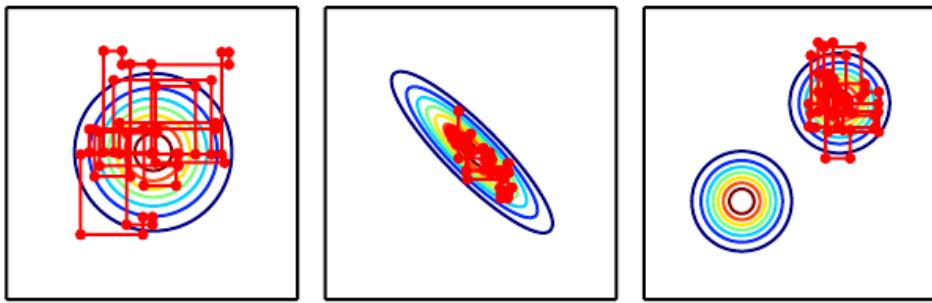


Figura 17.1: Caminos seguidos por el muestreo de Gibbs para tres distribuciones, con la cadena de Markov inicializada en el modo en ambos casos. (Izquierda) Una distribución normal multivariante con dos variables independientes. El muestreo de Gibbs se mezcla bien porque las variables son independientes. (Centro) Una distribución normal multivariante con variables altamente correlacionadas. La correlación entre variables dificulta que la cadena de Markov se mezcle. Debido a que la actualización de cada variable debe estar condicionada a la otra variable, la correlación reduce la velocidad a la que la cadena de Markov puede alejarse del punto de partida. (Derecha) Una mezcla de gaussianas con modos ampliamente separados que no están alineados con el eje. El muestreo de Gibbs se mezcla muy lentamente porque es difícil cambiar de modo mientras se modifica solo una variable a la vez.

De acuerdo a $PAG_{\text{modelo}}(a/b)$, ambas variables deben tener el mismo signo. Esto significa que el muestreo de Gibbs rara vez cambiará los signos de estas variables.

En escenarios más prácticos, el desafío es aún mayor porque no solo nos preocupamos por hacer transiciones entre dos modos, sino más en general entre todos los modos que puede contener un modelo real. Si varias de estas transiciones son difíciles debido a la dificultad de mezclar entre modos, resulta muy costoso obtener un conjunto confiable de muestras que cubra la mayoría de los modos, y la convergencia de la cadena a su distribución estacionaria es muy lenta.

A veces, este problema se puede resolver encontrando grupos de unidades altamente dependientes y actualizándolos todos simultáneamente en un bloque. Desafortunadamente, cuando las dependencias son complicadas, puede ser difícil computacionalmente extraer una muestra del grupo. Después de todo, el problema para el que se introdujo originalmente la cadena de Markov es este problema de muestreo de un gran grupo de variables.

En el contexto de modelos con variables latentes, que definen una distribución conjunta $pag_{\text{modelo}}(x, h)$, a menudo extraemos muestras de X alternando entre el muestreo de $pag_{\text{modelo}}(x / h)$ y muestreo de $pag_{\text{modelo}}(h / X)$. Desde el punto de vista de la mezcla.

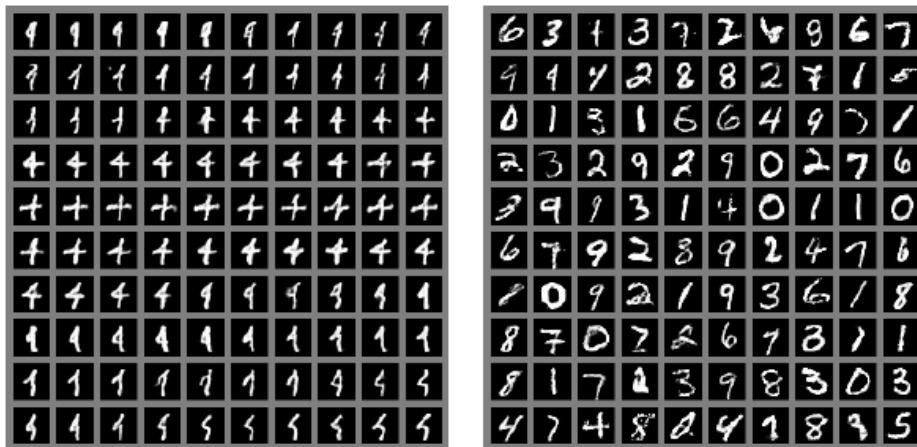


Figura 17.2: Una ilustración del problema de mezcla lenta en modelos probabilísticos profundos. Cada panel debe leerse de izquierda a derecha y de arriba a abajo. (Izquierda) Muestras consecutivas del muestreo de Gibbs aplicadas a una máquina Boltzmann profunda entrenada en el conjunto de datos MNIST. Las muestras consecutivas son similares entre sí. Debido a que el muestreo de Gibbs se realiza en un modelo gráfico profundo, esta similitud se basa más en características semánticas que visuales en bruto, pero todavía es difícil para la cadena de Gibbs hacer la transición de un modo de distribución a otro, por ejemplo, cambiando el identidad de dígitos. (Derecha) Muestras ancestrales consecutivas de una red adversarial generativa. Debido a que el muestreo ancestral genera cada muestra independientemente de las demás, no hay problema de mezcla.

rápidamente, nos gustaría *que* $p_{\text{modelo}}(h \mid X)$ tener una entropía muy alta. Sin embargo, desde el punto de vista del aprendizaje de una representación útil de h , nos gustaría h para codificar suficiente información sobre X reconstruirlo bien, lo que implica que h y X deben tener una información mutua muy alta. Estos dos objetivos están en desacuerdo entre sí. A menudo aprendemos modelos generativos que codifican con mucha precisión X en h pero no se pueden mezclar muy bien. Esta situación surge con frecuencia con las máquinas de Boltzmann: cuanto más nítida es la distribución que aprende una máquina de Boltzmann, más difícil es que se mezcle bien una muestra de cadena de Markov de la distribución modelo. Este problema se ilustra en la figura 17.2.

Todo esto podría hacer que los métodos MCMC sean menos útiles cuando la distribución de interés tiene una estructura múltiple con una variedad separada para cada clase: la distribución se concentra en muchos modos y estos modos están separados por vastas regiones de alta energía. Este tipo de distribución es lo que esperamos en muchos problemas de clasificación y haría que los métodos MCMC convergieran muy lentamente debido a la mala mezcla entre modos.

17.5.1 Templado para mezclar entre modos

Cuando una distribución tiene picos pronunciados de alta probabilidad rodeados por regiones de baja probabilidad, es difícil mezclar los diferentes modos de la distribución. Varias técnicas para una mezcla más rápida se basan en la construcción de versiones alternativas de la distribución objetivo en las que los picos no son tan altos y los valles circundantes no son tan bajos. Los modelos basados en la energía proporcionan una forma particularmente sencilla de hacerlo. Hasta ahora, hemos descrito un modelo basado en energía que define una distribución de probabilidad

$$p_{\alpha}(X) \propto \text{Exp}(-E(X)). \quad (17.25)$$

Los modelos basados en energía se pueden aumentar con un parámetro adicional β controlando cuán bruscamente pico es la distribución:

$$p_{\alpha\beta}(X) \propto \text{Exp}(-\beta E(X)). \quad (17.26)$$

El β parámetro se describe a menudo como el recíproco de la **temperatura**, que refleja el origen de los modelos basados en energía en la física estadística. Cuando la temperatura desciende a cero y β se eleva al infinito, el modelo basado en la energía se vuelve determinista. Cuando la temperatura sube al infinito y β cae a cero, la distribución (para discretos X) se vuelve uniforme.

Por lo general, un modelo se entrena para ser evaluado en $\beta=1$. Sin embargo, podemos hacer uso de otras temperaturas, particularmente aquellas donde $\beta < 1$. **templado**es una estrategia general de mezcla entre modos de p_{α} rápidamente extrayendo muestras con $\beta < 1$.

Cadenas de Markov basadas en **transiciones moderadas**(Neal, 1994) tome muestras temporalmente de distribuciones de temperatura más alta para mezclar en diferentes modos, luego reanude el muestreo de la distribución de temperatura de la unidad. Estas técnicas se han aplicado a modelos como los RBM (Salakhutdinov, 2010). Otro enfoque es utilizar **templado paralelo**(Iba, 2001), en el que la cadena de Markov simula muchos estados diferentes en paralelo, a diferentes temperaturas. Los estados de temperatura más alta se mezclan lentamente, mientras que los estados de temperatura más baja, a temperatura 1, proporcionar muestras precisas del modelo. El operador de transición incluye el intercambio estocástico de estados entre dos niveles de temperatura diferentes, de modo que una muestra de probabilidad suficientemente alta de una ranura de temperatura alta pueda saltar a una ranura de temperatura más baja. Este enfoque también se ha aplicado a los RBM (Desjardins et al., 2010; Cho et al., 2010). Aunque el templado es un enfoque prometedor, en este momento no ha permitido a los investigadores hacer un gran avance para resolver el desafío de tomar muestras de EBM complejos. Una posible razón es que hay **temperaturas críticas**alrededor del cual la transición de temperatura debe ser muy lenta (ya que la temperatura se reduce gradualmente) para que el revenido sea efectivo.

17.5.2 La profundidad puede ayudar a mezclar

Al extraer muestras de un modelo de variable latente $p_{\text{ag}}(h, x)$, hemos visto que si $p_{\text{ag}}(h / X)$ codifica X demasiado bien, entonces el muestreo de $p_{\text{ag}}(x / h)$ no cambiará X mucho y la mezcla será pobre. Una forma de resolver este problema es hacer / ser una representación profunda, que codifica X en h de tal manera que una cadena de Markov en el espacio de h puede mezclar más fácilmente. Muchos algoritmos de aprendizaje de representación, como codificadores automáticos y RBM, tienden a producir una distribución marginal sobre h que es más uniforme y más unimodal que la distribución de datos original sobre X . Se puede argumentar que esto surge al tratar de minimizar el error de reconstrucción mientras se usa todo el espacio de representación disponible, porque la minimización del error de reconstrucción sobre los ejemplos de entrenamiento se logrará mejor cuando los diferentes ejemplos de entrenamiento se distingan fácilmente entre sí en h -espacio, y por lo tanto bien separados. [bengio et al.](#) (2013a) observaron que pilas más profundas de codificadores automáticos regularizados o RBM producen distribuciones marginales en el nivel superior h -espacio que apareció más disperso y más uniforme, con menos espacio entre las regiones correspondientes a diferentes modos (categorías, en los experimentos). Entrenar un RBM en ese espacio de nivel superior permitió que el muestreo de Gibbs se mezclara más rápido entre los modos. Sin embargo, no está claro cómo explotar esta observación para ayudar a entrenar y muestrear mejor a partir de modelos generativos profundos.

A pesar de la dificultad de mezclarlas, las técnicas de Monte Carlo son útiles y, a menudo, son la mejor herramienta disponible. De hecho, son la herramienta principal utilizada para enfrentar la función de partición intratable de los modelos no dirigidos, que se analiza a continuación.

capítulo 18

Confrontando la función de partición

En la sección 16.2.2 vimos que muchos modelos probabilísticos (comúnmente conocidos como modelos gráficos no dirigidos) están definidos por una distribución de probabilidad no normalizada $pag(X; \theta)$. Debemos normalizar pag dividiendo por una función de partición $Z(\theta)$ para obtener una distribución de probabilidad válida:

$$pag(X; \theta) = \frac{1}{Z(\theta)} pag(X; \theta). \quad (18.1)$$

La función de partición es una integral (para variables continuas) o una suma (para variables discretas) sobre la probabilidad no normalizada de todos los estados:

$$\int pag(X) dX \quad (18.2)$$

$$\sum_X pag(X). \quad (18.3)$$

Esta operación es intratable para muchos modelos interesantes.

Como veremos en el capítulo 20, varios modelos de aprendizaje profundo están diseñados para tener una constante de normalización manejable, o están diseñados para usarse de formas que no involucran computación $pag(X)$ en absoluto. Sin embargo, otros modelos enfrentan directamente el desafío de las funciones de partición intratables. En este capítulo, describimos técnicas utilizadas para entrenar y evaluar modelos que tienen funciones de partición intratables.

18.1 El gradiente logarítmico de verosimilitud

Lo que hace particularmente difícil el aprendizaje de modelos no dirigidos por máxima verosimilitud es que la función de partición depende de los parámetros. El gradiente del log-verosimilitud con respecto a los parámetros tiene un término correspondiente al gradiente de la función de partición:

$$\nabla_{\theta} \text{registro}_{\text{pag}}(X; \theta) = \nabla_{\theta} \text{registro}_{\text{pag}}(X; \theta) - \nabla_{\theta} \text{registro}_Z(\theta). \quad (18.4)$$

Esta es una descomposición bien conocida en el**fase positiva**y**fase negativa** de aprendizaje.

Para la mayoría de los modelos de interés no dirigidos, la fase negativa es difícil. Los modelos sin variables latentes o con pocas interacciones entre variables latentes suelen tener una fase positiva tratable. El ejemplo por excelencia de un modelo con una fase positiva sencilla y una fase negativa difícil es el RBM, que tiene unidades ocultas que son condicionalmente independientes entre sí dadas las unidades visibles. El caso donde la fase positiva es difícil, con interacciones complicadas entre variables latentes, se trata principalmente en el capítulo19. Este capítulo se centra en las dificultades de la fase negativa.

Veamos más de cerca el gradiente deregistro Z :

$$\nabla_{\theta} \text{registro}_Z \quad (18.5)$$

$$= \frac{\nabla_{\theta} Z}{Z} \quad (18.6)$$

$$= \frac{\nabla_{\theta} \sum_x p_{\text{pag}}(x)}{Z} \quad (18.7)$$

$$= \frac{\sum_x x \nabla_{\theta} p_{\text{pag}}(x)}{Z}. \quad (18.8)$$

Para modelos que garantizan $p_{\text{pag}}(X) > 0$ para todos X , podemos sustituir $\exp(\text{registro}_{\text{pag}}(X))$ para $p_{\text{pag}}(X)$:

$$= \frac{x \nabla_{\theta} \exp(\text{registro}_{\text{pag}}(X))}{Z} \quad (18.9)$$

$$= \frac{x \exp(\text{registro}_{\text{pag}}(X)) \nabla_{\theta} \text{registro}_{\text{pag}}(X)}{Z} \quad (18.10)$$

$$= \frac{x (p_{\text{pag}}(X) \nabla_{\theta} \text{registro}_{\text{pag}}(X))}{Z} \quad (18.11)$$

$$= \frac{x p_{\text{pag}}(X) \nabla_{\theta} \text{registro}_{\text{pag}}(X)}{Z} \quad (18.12)$$

$$= \text{mix-} \text{pag}(X) \nabla_{\theta} \text{registro} \text{pag}(X). \quad (18.13)$$

Esta derivación hizo uso de la suma sobre discretos X , pero se aplica un resultado similar usando integración sobre continuo X . En la versión continua de la derivación, usamos la regla de Leibniz para diferenciación bajo el signo integral para obtener la identidad

$$\nabla_{\theta} \text{pag}(X) dX = \nabla_{\theta} \text{pag}(X) dX. \quad (18.14)$$

Esta identidad es aplicable sólo bajo ciertas condiciones de regularidad en pag y $\nabla_{\theta} \text{pag}(X)$. En términos de teoría de la medida, las condiciones son: (i) La distribución no normalizada pag debe ser una función integrable de Lebesgue de X por cada valor de θ ; (ii) El gradiente $\nabla_{\theta} \text{pag}(X)$ debe existir para todos θ y casi todos X ; (iii) Debe existir un integrable función $R(X)$ que limita $\nabla_{\theta} \text{pag}(X)$ en el sentido de que $\max_{\theta} |\partial_{\theta} \text{pag}(X)| \leq R(X)$ para todos θ y casi todos X . Afortunadamente, la mayoría de los modelos de aprendizaje automático de interés tienen estas propiedades.

esta identidad

$$\nabla_{\theta} \text{registro} Z = \text{mix-} \text{pag}(X) \nabla_{\theta} \text{registro} \text{pag}(X) \quad (18.15)$$

es la base de una variedad de métodos de Monte Carlo para maximizar aproximadamente la probabilidad de modelos con funciones de partición intratables.

El enfoque de Monte Carlo para aprender modelos no dirigidos proporciona un marco intuitivo en el que podemos pensar tanto en la fase positiva como en la fase negativa. En la fase positiva, aumentamos $\text{registro} \text{pag}(X)$ para X extraído de los datos. En la fase negativa, disminuimos la función de partición al disminuir $\text{registro} \text{pag}(X)$ extraído de la distribución del modelo.

En la literatura de aprendizaje profundo, es común parametrizar $\text{registro} \text{pag}$ en términos de una función de energía (ecuación 16.7). En este caso, podemos interpretar la fase positiva como empujando hacia abajo la energía de los ejemplos de entrenamiento y la fase negativa como empujando hacia arriba la energía de las muestras extraídas del modelo, como se ilustra en la figura 18.1.

18.2 MÁXIMA VEROSIMILITUD ESTOCÁSTICA Y DIVERGENCIA CONTRASTIVA

La forma ingenua de implementar la ecuación 18.15 es calcularlo quemando un conjunto de cadenas de Markov a partir de una inicialización aleatoria cada vez que se necesita el gradiente. Cuando el aprendizaje se realiza mediante el descenso de gradiente estocástico, esto significa que las cadenas deben grabarse una vez por paso de gradiente. Este enfoque conduce a la

procedimiento de entrenamiento presentado en el algoritmo 18.1. El alto costo de quemar las cadenas de Markov en el bucle interno hace que este procedimiento no sea factible computacionalmente, pero este procedimiento es el punto de partida que otros algoritmos más prácticos pretenden aproximar.

Algoritmo 18.1 Un algoritmo MCMC ingenuo para maximizar la probabilidad logarítmica con una función de partición intratable que utiliza el ascenso de gradiente.

Colocar γ , el tamaño del paso, a un pequeño número positivo.

Colocar k , el número de pasos de Gibbs, lo suficientemente alto como para permitir grabar. Quizás 100 para entrenar un RBM en un pequeño parche de imagen.

mientras no convergentehacer

Muestra un mini lote de γ de metro ejemplos $\{X(1), \dots, X(\text{metro})\}$ del conjunto de entrenamiento.

gramo $\leftarrow \frac{\text{gramo}}{\text{metro}}$ $\gamma = 1$ $\nabla \theta \text{registro}_{\text{pag}}(X(\gamma); \theta)$.

Inicializar un conjunto de metro muestras $\{X(1), \dots, X(\text{metro})\}$ a valores aleatorios (por ejemplo, de una distribución uniforme o normal, o posiblemente una distribución con marginales coincidentes con los marginales del modelo).

para $\gamma = 1$ a k **hacer**

para $j = 1$ a metro **hacer**

$X(j) \leftarrow \text{gibbs_update}(\tilde{X}(j))$. **fin**

para

fin para

gramo $\leftarrow \frac{\text{gramo}}{\text{metro}}$ $\gamma = 1$ $\nabla \theta \text{registro}_{\text{pag}}(X(\gamma); \theta)$.

$\theta \leftarrow \theta + \text{gramo}$.

terminar mientras

Podemos ver el enfoque MCMC de máxima verosimilitud como un intento de lograr el equilibrio entre dos fuerzas, una empujando hacia arriba la distribución del modelo donde ocurren los datos y otra empujando hacia abajo la distribución del modelo donde ocurren las muestras del modelo. Cifra 18.1 ilustra este proceso. Las dos fuerzas corresponden a maximizar $\text{registro}_{\text{pag}}$ minimizando $\text{registro}_{\text{Z}}$. Varias aproximaciones a la fase negativa son posibles. Se puede entender que cada una de estas aproximaciones hace que la fase negativa sea computacionalmente más barata, pero también la empuja hacia abajo en las ubicaciones incorrectas.

Debido a que la fase negativa implica extraer muestras de la distribución del modelo, podemos considerarla como encontrar puntos en los que el modelo cree firmemente.

Debido a que la fase negativa actúa para reducir la probabilidad de esos puntos, generalmente se considera que representan las creencias incorrectas del modelo sobre el mundo. En la literatura se las denomina con frecuencia "alucinaciones" o "partículas de fantasía". De hecho, la fase negativa se ha propuesto como una posible explicación

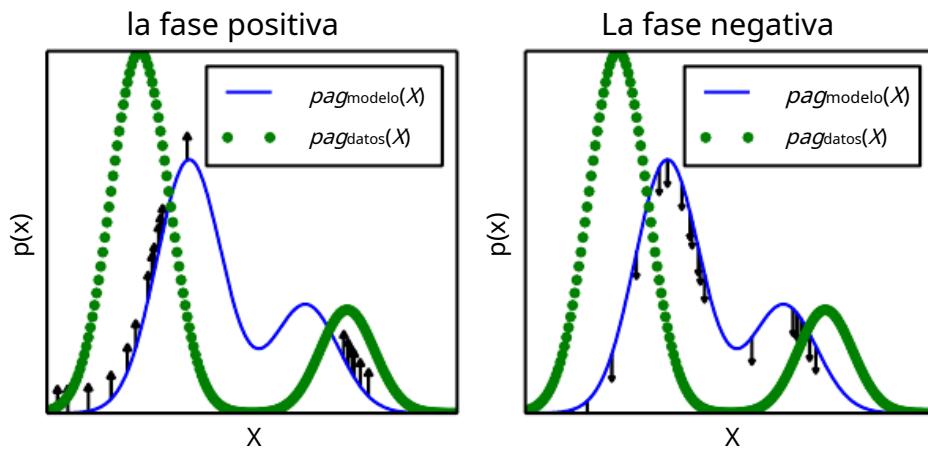


Figura 18.1: La vista del algoritmo 18.1 como teniendo una “fase positiva” y una “fase negativa”. (Izquierda) En la fase positiva, muestreamos puntos de la distribución de datos y elevamos su probabilidad no normalizada. Esto significa que los puntos que probablemente aparecen en los datos aumentan más. (Bien) En la fase negativa, muestreamos puntos de la distribución del modelo y empujamos hacia abajo su probabilidad no normalizada. Esto contrarresta la tendencia de la fase positiva de simplemente agregar una gran constante a la probabilidad no normalizada en todas partes. Cuando la distribución de datos y la distribución del modelo son iguales, la fase positiva tiene la misma posibilidad de empujar hacia arriba en un punto que la fase negativa tiene que empujar hacia abajo. Cuando esto ocurre, ya no hay ningún gradiente (en expectativa) y el entrenamiento debe terminar.

por soñar en humanos y otros animales (Crick y Mitchison, 1983), la idea es que el cerebro mantiene un modelo probabilístico del mundo y sigue el gradiente deregistro p_{ag} mientras experimenta eventos reales mientras está despierto y sigue el gradiente negativo deregistro $p_{\text{ag}} \text{minimizarregistro}$ mientras duerme y experimenta eventos muestrados del modelo actual. Este punto de vista explica gran parte del lenguaje utilizado para describir algoritmos con una fase positiva y negativa, pero no se ha demostrado que sea correcto con experimentos neurocientíficos. En los modelos de aprendizaje automático, generalmente es necesario usar la fase positiva y negativa simultáneamente, en lugar de períodos separados de vigilia y sueño REM. Como veremos en la sección 19.5, otros algoritmos de aprendizaje automático extraen muestras de la distribución del modelo para otros fines y dichos algoritmos también podrían proporcionar una explicación de la función del sueño soñado.

Dada esta comprensión del papel de la fase positiva y negativa del aprendizaje, podemos intentar diseñar una alternativa menos costosa al algoritmo 18.1. El costo principal del algoritmo MCMC ingenuo es el costo de quemar las cadenas de Markov a partir de una inicialización aleatoria en cada paso. Una solución natural es inicializar las cadenas de Markov a partir de una distribución muy cercana a la distribución del modelo,

para que la operación de quemado no tome tantos pasos.

El **divergencia contrastiva**(CD, o CD- k para indicar CD con k pasos de Gibbs) inicializa la cadena de Markov en cada paso con muestras de la distribución de datos ([Hinton,2000,2010](#)). Este enfoque se presenta como algoritmo.[18.2](#). La obtención de muestras de la distribución de datos es gratuita, porque ya están disponibles en el conjunto de datos. Inicialmente, la distribución de datos no está cerca de la distribución del modelo, por lo que la fase negativa no es muy precisa. Afortunadamente, la fase positiva aún puede aumentar con precisión la probabilidad de los datos del modelo. Después de que la fase positiva ha tenido algún tiempo para actuar, la distribución del modelo se acerca más a la distribución de datos y la fase negativa comienza a ser precisa.

Algoritmo 18.2 El algoritmo de divergencia contrastiva, utilizando gradiente ascendente como procedimiento de optimización.

Colocar- el tamaño del paso, a un pequeño número positivo.

Colocar k , el número de pasos de Gibbs, lo suficientemente alto como para permitir un muestreo en cadena de Markov de $\text{p}_{\theta}(X; \theta)$ para mezclar cuando se inicializa desde $\text{p}_{\theta}^{\text{datos}}$. Quizás 1-20 para entrenar un RBM en un pequeño parche de imagen.

mientrasno convergentehacer

Muestrar un mini lote de $metro$ ejemplos $\{X_{(1)}, \dots, X_{(metro)}\}$ del conjunto de entrenamiento.

gramo $\leftarrow \frac{1}{metro} \sum_{i=1}^{metro} \nabla_{\theta} \text{registro} \text{p}_{\theta}(X_{(i)}; \theta)$.

para $i=1$ a $metro$ **hacer**

$X_{(i)} \leftarrow X_{(i)}$.

fin para

para $j=1$ a k **hacer**

para $j=1$ a $metro$ **hacer**

$X_{(j)} \leftarrow \text{gibbs_update}(\tilde{X}_{(j)})$. **fin**

para

fin para

gramo $\leftarrow \frac{1}{metro} \sum_{i=1}^{metro} \nabla_{\theta} \text{registro} \text{p}_{\theta}(X_{(i)}; \theta)$.

$\theta \leftarrow \theta + \text{gramo}$.

terminar mientras

Por supuesto, CD sigue siendo una aproximación a la fase negativa correcta. La forma principal en que CD falla cualitativamente al implementar la fase negativa correcta es que no suprime regiones de alta probabilidad que están lejos de los ejemplos de entrenamiento reales. Estas regiones que tienen alta probabilidad bajo el modelo pero baja probabilidad bajo la distribución de generación de datos se denominan**modos espurios**. Cifra[18.2](#)ilustra por qué sucede esto. Esencialmente, se debe a que los modos en la distribución del modelo que están lejos de la distribución de datos no serán visitados por

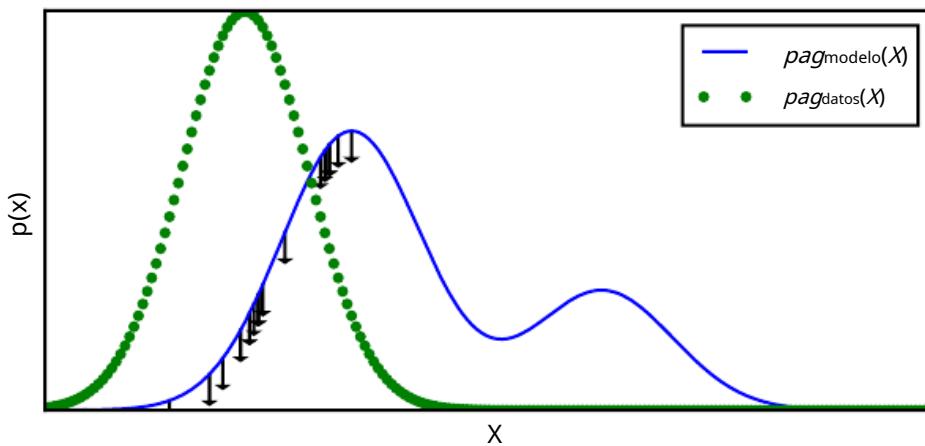


Figura 18.2: Una ilustración de cómo la fase negativa de la divergencia contrastiva (algoritmo 18.2) puede fallar en suprimir los modos espurios. Un modo espurio es un modo que está presente en la distribución del modelo pero ausente en la distribución de datos. Debido a que la divergencia contrastiva inicializa sus cadenas de Markov a partir de puntos de datos y ejecuta la cadena de Markov solo durante unos pocos pasos, es poco probable que visite modos en el modelo que están lejos de los puntos de datos. Esto significa que al tomar muestras del modelo, a veces obtendremos muestras que no se parecen a los datos. También significa que debido a que desperdicia parte de su masa de probabilidad en estos modos, el modelo tendrá dificultades para colocar una masa de alta probabilidad en los modos correctos. A los efectos de la visualización, esta figura utiliza un concepto algo simplificado de distancia: el modo espurio está lejos del modo correcto a lo largo de la recta numérica en

rEsto corresponde a una cadena de Markov basada en hacer movimientos locales con un solo X variable en Para la mayoría de los modelos probabilísticos profundos, las cadenas de Markov se basan en el muestreo de Gibbs y pueden realizar movimientos no locales de variables individuales, pero no pueden mover todas las variables simultáneamente. Para estos problemas, normalmente es mejor considerar la distancia de edición entre modos, en lugar de la distancia euclíadiana. Sin embargo, la distancia de edición en un espacio dimensional alto es difícil de representar en un gráfico 2D.

Cadenas de Markov inicializadas en puntos de entrenamiento, a menos que ϵ sea muy grande.

[Carreira-Perpiñán y Hinton\(2005\)](#) mostró experimentalmente que el estimador de CD está sesgado para RBM y máquinas de Boltzmann completamente visibles, ya que converge en puntos diferentes que el estimador de máxima verosimilitud. Argumentan que debido a que el sesgo es pequeño, el CD podría usarse como una forma económica de inicializar un modelo que luego podría ajustarse a través de métodos MCMC más costosos. [Bengio y Delalleau\(2009\)](#) mostró que CD puede interpretarse como descartando los términos más pequeños del gradiente de actualización de MCMC correcto, lo que explica el sesgo.

CD es útil para entrenar modelos poco profundos como RBM. Estos, a su vez, se pueden apilar para inicializar modelos más profundos como DBN o DBM. Sin embargo, CD no proporciona mucha ayuda para entrenar modelos más profundos directamente. Esto se debe a que es difícil

obtener muestras de las unidades ocultas dadas muestras de las unidades visibles. Dado que las unidades ocultas no están incluidas en los datos, la inicialización desde los puntos de entrenamiento no puede resolver el problema. Incluso si inicializamos las unidades visibles de los datos, aún necesitaremos grabar en una cadena de Markov el muestreo de la distribución sobre las unidades ocultas condicionadas en esas muestras visibles.

Se puede pensar que el algoritmo CD penaliza al modelo por tener una cadena de Markov que cambia la entrada rápidamente cuando la entrada proviene de los datos. Esto significa que el entrenamiento con CD se parece un poco al entrenamiento con codificador automático. Aunque CD tiene más sesgo que algunos de los otros métodos de entrenamiento, puede ser útil para el entrenamiento previo de modelos poco profundos que luego se apilarán. Esto se debe a que se alienta a los primeros modelos de la pila a copiar más información hasta sus variables latentes, poniéndola así a disposición de los modelos posteriores. Esto se debe considerar más como un efecto secundario a menudo aprovechable del entrenamiento de CD en lugar de una ventaja de diseño basada en principios.

[Sutskever y Tieleman\(2010\)](#) mostró que la dirección de actualización del CD no es el gradiente de ninguna función. Esto permite situaciones en las que el CD podría circular para siempre, pero en la práctica esto no es un problema grave.

Una estrategia diferente que resuelve muchos de los problemas con CD es inicializar las cadenas de Markov en cada paso de gradiente con sus estados del paso de gradiente anterior. Este enfoque fue descubierto por primera vez bajo el nombre **máxima verosimilitud estocástica**(SML) en la comunidad de matemática aplicada y estadística ([younes,1998](#)) y más tarde redescubierta de forma independiente bajo el nombre **divergencia contrastiva persistente**(PCD, o PCD- k para indicar el uso de k Pasos de Gibbs por actualización) en la comunidad de aprendizaje profundo ([Tieleman,2008](#)). Ver algoritmo 18.3. La idea básica de este enfoque es que, siempre que los pasos tomados por el algoritmo de gradiente estocástico sean pequeños, el modelo del paso anterior será similar al modelo del paso actual. De ello se deduce que las muestras de la distribución del modelo anterior estarán muy cerca de ser muestras justas de la distribución del modelo actual, por lo que una cadena de Markov inicializada con estas muestras no requerirá mucho tiempo para mezclar.

Debido a que cada cadena de Markov se actualiza continuamente a lo largo del proceso de aprendizaje, en lugar de reiniciarse en cada paso de gradiente, las cadenas son libres de moverse lo suficiente como para encontrar todos los modos del modelo. SML es, por lo tanto, considerablemente más resistente a la formación de modelos con modos espurios que CD. Además, debido a que es posible almacenar el estado de todas las variables muestreadas, ya sean visibles o latentes, SML proporciona un punto de inicialización tanto para las unidades ocultas como para las visibles. CD solo puede proporcionar una inicialización para las unidades visibles y, por lo tanto, requiere grabación para modelos profundos. SML puede entrenar modelos profundos de manera eficiente.

Aguja *et al.* (2010) compararon SML con muchos de los otros criterios presentados en este capítulo. Descubrieron que SML da como resultado la mejor probabilidad de registro del conjunto de prueba para un RBM, y que si las unidades ocultas de RBM se usan como características para un clasificador SVM, SML da como resultado la mejor precisión de clasificación.

SML es vulnerable a volverse inexacto si el algoritmo de gradiente estocástico puede mover el modelo más rápido de lo que la cadena de Markov puede mezclar entre pasos. Esto puede suceder si η es demasiado pequeño o θ es demasiado largo. Desafortunadamente, el rango de valores permisible depende en gran medida del problema. No existe una forma conocida de probar formalmente si la cadena se está mezclando con éxito entre los pasos. Subjetivamente, si la tasa de aprendizaje es demasiado alta para el número de pasos de Gibbs, el operador humano podrá observar que hay mucha más variación en las muestras de fase negativa en los pasos de gradiente que en las diferentes cadenas de Markov. Por ejemplo, un modelo entrenado en MNIST podría muestrear exclusivamente 7 en un paso. El proceso de aprendizaje presionará fuertemente hacia abajo en el modo correspondiente a 7s, y el modelo podría muestrear exclusivamente 9s en el siguiente paso.

Algoritmo 18.3 El algoritmo estocástico de máxima verosimilitud/divergencia contrastiva persistente que utiliza el ascenso de gradiente como procedimiento de optimización.

Colocar η , el tamaño del paso, a un pequeño número positivo.

Colocar k , el número de pasos de Gibbs, lo suficientemente alto como para permitir un muestreo en cadena de Markov de $p_{\text{ag}}(X; \theta + \eta \text{gramo})$ para quemar, a partir de muestras $p_{\text{ag}}(X; \theta)$. Quizás 1 para RBM en un pequeño parche de imagen, o 5-50 para un modelo más complicado como un DBM. Inicializar un conjunto de m muestras $\{X^{(1)}, \dots, X^{(m)}\}$ a valores aleatorios (por ejemplo, de una distribución uniforme o normal, o posiblemente una distribución con marginales coincidentes con los marginales del modelo).

mientras no convergente **hacer**

 Muestra un mini lote de m ejemplos $\{X^{(1)}, \dots, X^{(m)}\}$ del conjunto de entrenamiento.

 gramo \leftarrow $\frac{1}{m} \sum_{i=1}^m \nabla_{\theta} \log p_{\text{ag}}(X^{(i)}; \theta)$.

para $i = 1$ **a** k **hacer**

para $j = 1$ **a** m **hacer**

$X^{(j)} \leftarrow \text{gibbs_update}(\tilde{X}^{(j)})$.

fin

para

fin para

 gramo \leftarrow $\frac{1}{m} \sum_{i=1}^m \nabla_{\theta} \log p_{\text{ag}}(X^{(i)}; \theta)$.

$\theta \leftarrow \theta + \eta \text{gramo}$.

terminar mientras

Se debe tener cuidado al evaluar las muestras de un modelo entrenado con SML. Es necesario extraer las muestras a partir de una nueva cadena de Markov

inicializado desde un punto de inicio aleatorio después de que el modelo haya terminado de entrenar. Las muestras presentes en las cadenas negativas persistentes utilizadas para el entrenamiento se han visto influenciadas por varias versiones recientes del modelo y, por lo tanto, pueden hacer que el modelo parezca tener una mayor capacidad de la que realmente tiene.

Berglund y Raiko(2013) realizaron experimentos para examinar el sesgo y la varianza en la estimación del gradiente proporcionada por CD y SML. CD demuestra tener una varianza más baja que el estimador basado en un muestreo exacto. SML tiene una varianza más alta. La causa de la baja varianza de CD es el uso de los mismos puntos de entrenamiento tanto en la fase positiva como en la negativa. Si la fase negativa se inicializa desde diferentes puntos de entrenamiento, la varianza se eleva por encima de la del estimador basado en un muestreo exacto.

Todos estos métodos basados en el uso de MCMC para extraer muestras del modelo pueden, en principio, usarse con casi cualquier variante de MCMC. Esto significa que las técnicas como SML se pueden mejorar utilizando cualquiera de las técnicas mejoradas de MCMC descritas en el capítulo17, como el templado en paralelo (Desjardins et al., 2010; Cho et al., 2010).

Un enfoque para acelerar la mezcla durante el aprendizaje no se basa en cambiar la tecnología de muestreo de Monte Carlo, sino en cambiar la parametrización del modelo y la función de costo.**PCD rápido**o FPCD (Tieleman y Hinton, 2009) implica reemplazar los parámetros θ de un modelo tradicional con expresión

$$\theta = \theta_{\text{lento}} + \theta_{\text{rápido}}. \quad (18.16)$$

Ahora hay el doble de parámetros que antes, y se suman por elementos para proporcionar los parámetros utilizados por la definición del modelo original. La copia rápida de los parámetros se entrena con una tasa de aprendizaje mucho mayor, lo que le permite adaptarse rápidamente en respuesta a la fase negativa de aprendizaje y empujar la cadena de Markov a un nuevo territorio. Esto obliga a la cadena de Markov a mezclarse rápidamente, aunque este efecto solo ocurre durante el aprendizaje, mientras que los pesos rápidos pueden cambiar libremente. Por lo general, también se aplica una disminución de peso significativa a los pesos rápidos, alentándolos a converger a valores pequeños, después de tomar valores grandes solo transitoriamente durante el tiempo suficiente para alentar a la cadena de Markov a cambiar de modo.

Un beneficio clave de los métodos basados en MCMC descritos en esta sección es que proporcionan una estimación del gradiente deregistro Z y por lo tanto podemos esencialmente descomponer el problema en elregistro pag contribución y laregistro Z contribución. Entonces podemos usar cualquier otro método para abordarregistro $pag(X)$,y simplemente agregue nuestro gradiente de fase negativa al gradiente del otro método. En particular, esto significa que nuestra fase positiva puede hacer uso de métodos que proporcionan solo un límite inferior en pag . La mayoría de los otros métodos de tratar conregistro Z presentadas en este capítulo son

incompatible con los métodos de fase positiva basados en ligaduras.

18.3 Pseudoverosimilitud

Las aproximaciones de Monte Carlo a la función de partición y su gradiente confrontan directamente la función de partición. Otros enfoques eluden el problema al entrenar el modelo sin calcular la función de partición. La mayoría de estos enfoques se basan en la observación de que es fácil calcular razones de probabilidades en un modelo probabilístico no dirigido. Esto se debe a que la función de partición aparece tanto en el numerador como en el denominador de la razón y cancela:

$$\frac{pag(X)}{pag(y)} = \frac{\tilde{z}pag(X)}{\tilde{z}pag(y)} = \frac{pag(X)}{pag^*(y)}. \quad (18.17)$$

La pseudoverosimilitud se basa en la observación de que las probabilidades condicionales toman esta forma basada en la relación y , por lo tanto, se pueden calcular sin conocer la función de partición. Supongamos que particionamos X en a , b y C , donde a contiene las variables sobre las que queremos encontrar la distribución condicional, b contiene las variables que queremos condicionar, y C contiene las variables que no forman parte de nuestra consulta.

$$pag(a/b) = \frac{pag(a,b)}{pag(b)} = \frac{pag(a,b)}{\sum_{a,c} pag(a,b,C)} = \frac{pag(a,b)}{\sum_{a,c} pag(a,b,C)}. \quad (18.18)$$

Esta cantidad requiere marginar a , que puede ser una operación muy eficiente siempre que a y C no contienen muchas variables. En el caso extremo, a puede ser una sola variable y C puede estar vacío, por lo que esta operación requiere solo tantas evaluaciones de pag a que hay valores de una sola variable aleatoria.

Desafortunadamente, para calcular el logaritmo de la verosimilitud, necesitamos marginar grandes conjuntos de variables. Si hay n total de variables, debemos marginar un conjunto de tamaño $n - 1$. Por la regla de la cadena de probabilidad,

$$\text{registro}pag(x) = \text{registro}pag(X_1) + \text{registro}pag(X_2/X_1) + \dots + \text{registro}pag(X_{n-1}/X_{1:n-1}). \quad (18.19)$$

En este caso, hemos hecho a más o menos pequeño, pero C puede ser tan grande como $X_{2:n}$. ¿Qué pasa si simplemente nos movemos C en b para reducir el costo computacional? Esto produce el **pseudoprobabiliad** (Besag, 1975) función objetivo, basada en la predicción del valor de la característica X_i dadas todas las otras características X_{-i} :

$$\begin{aligned} & \text{registro}pag(X_i/X_{-i}) \\ & \text{registro}pag(X_i/X_{-i}). \end{aligned} \quad (18.20)$$

$\neq 1$

Si cada variable aleatoria tiene k diferentes valores, esto requiere sólo $k \times n$ evaluaciones de *pag* calcular, a diferencia de k^n evaluaciones necesarias para calcular la función de partición.

Esto puede parecer un truco sin principios, pero se puede probar que la estimación mediante la maximización de la pseudoverosimilitud es asintóticamente consistente (Masé, 1995). Por supuesto, en el caso de conjuntos de datos que no se acerquen al límite de muestra grande, la pseudoverosimilitud puede mostrar un comportamiento diferente del estimador de máxima verosimilitud.

Es posible intercambiar complejidad computacional por desviación del comportamiento de máxima verosimilitud usando el **pseudoprobabilidad generalizada** estimador (Huang y Ogata, 2002). El estimador de pseudoverosimilitud generalizada utiliza m diferentes conjuntos $S_{(i)}, i=1, \dots, m$ de índices de variables que aparecen juntas en el lado izquierdo de la barra de condicionamiento. En el caso extremo de $m = 1$ y $S_{(1)} = 1, \dots, n$ la pseudoverosimilitud generalizada recupera la log-verosimilitud. En el caso extremo de $m = n$ y $S_{(i)} = \{i\}$, la pseudoverosimilitud generalizada recupera la pseudoverosimilitud. La función objetiva de pseudoverosimilitud generalizada viene dada por

$$\text{registro}_{\text{pag}}(X_{S_{(i)}} / X_{-S_{(i)}}) \quad (18.21)$$

$i=1$

El rendimiento de los enfoques basados en pseudoverosimilitudes depende en gran medida de cómo se utilice el modelo. La pseudoprobabilidad tiende a desempeñarse mal en tareas que requieren un buen modelo de la articulación completa $\text{pag}(X)$, como la estimación de la densidad y el muestreo. Sin embargo, puede funcionar mejor que la máxima probabilidad para tareas que requieren solo las distribuciones condicionales utilizadas durante el entrenamiento, como completar pequeñas cantidades de valores faltantes. Las técnicas de pseudoverosimilitud generalizada son especialmente poderosas si los datos tienen una estructura regular que permite laSlos conjuntos de índices deben diseñarse para capturar las correlaciones más importantes y dejar fuera grupos de variables que solo tienen una correlación insignificante. Por ejemplo, en imágenes naturales, los píxeles que están muy separados en el espacio también tienen una correlación débil, por lo que la pseudoverosimilitud generalizada se puede aplicar con cada uno. Sel conjunto es una pequeña ventana espacialmente localizada.

Una debilidad del estimador de pseudoverosimilitud es que no se puede usar con otras aproximaciones que proporcionan solo un límite inferior en $\text{pag}(X)$, como la inferencia variacional, que se tratará en el capítulo 19. Esto es porque pag aparece en el denominador. Un límite inferior en el denominador proporciona solo un límite superior en la expresión como un todo, y no hay ningún beneficio en maximizar un límite superior. Esto dificulta la aplicación de enfoques de pseudoverosimilitud a modelos profundos como las máquinas profundas de Boltzmann, ya que los métodos variacionales son uno de los enfoques dominantes para marginar aproximadamente las muchas capas de variables ocultas.

que interactúan entre sí. Sin embargo, la pseudoverosimilitud sigue siendo útil para el aprendizaje profundo, porque se puede usar para entrenar modelos de una sola capa o modelos profundos que usan métodos de inferencia aproximada que no se basan en límites inferiores.

La pseudoverosimilitud tiene un costo mucho mayor por paso de gradiente que SML, debido a su cálculo explícito de todos los condicionales. Sin embargo, la pseudoverosimilitud generalizada y criterios similares aún pueden funcionar bien si solo se calcula un condicional seleccionado al azar por ejemplo ([Buen compañero et al., 2013b](#)), lo que reduce el costo computacional para que coincida con el de SML.

Aunque el estimador de pseudoverosimilitud no minimiza explícitamente registro_Z , todavía se puede pensar que tiene algo parecido a una fase negativa. Los denominadores de cada distribución condicional dan como resultado que el algoritmo de aprendizaje suprima la probabilidad de todos los estados que tienen solo una variable que difiere de un ejemplo de entrenamiento.

Ver [Marlin y de Freitas\(2011\)](#) para un análisis teórico de la eficiencia asintótica de la pseudoverosimilitud.

18.4 Coincidencia de puntajes y coincidencia de proporciones

Coincidencia de puntuación ([Hyvärinen, 2005](#)) proporciona otro medio consistente de entrenar un modelo sin estimar Z o sus derivados. El nombre coincidencia de puntuación proviene de la terminología en la que las derivadas de una densidad logarítmica con respecto a su argumento, $\nabla \text{registro}_{\text{pag}}(X)$, se llaman **supuntaje**. La estrategia utilizada por la coincidencia de puntuación es minimizar la diferencia cuadrática esperada entre las derivadas de la densidad logarítmica del modelo con respecto a la entrada y las derivadas de la densidad logarítmica de los datos con respecto a la entrada:

$$L(x, \theta) = \frac{1}{2} \| \nabla_{\theta} \text{registro}_{\text{pagmodelo}}(X; \theta) - \nabla_{\theta} \text{registro}_{\text{pagdatos}}(X) \|_2^2 \quad (18.22)$$

$$\mathcal{J}(\theta) = \frac{1}{2} \text{mi}_{\text{pagdatos}(X)} L(x, \theta) \quad (18.23)$$

$$\theta^* = \min_{\theta} \mathcal{J}(\theta) \quad (18.24)$$

Esta función objetivo evita las dificultades asociadas con diferenciar la función de partición Z porque Z no es una función de X por lo tanto $\nabla X Z = 0$. Inicialmente, la coincidencia de puntuaciones parece tener una nueva dificultad: calcular la puntuación de la distribución de datos requiere el conocimiento de la verdadera distribución que genera los datos de entrenamiento. pagdatos . Afortunadamente, minimizar el valor esperado de $L(x, \theta)$ es

equivalente a minimizar el valor esperado de

$$L(x, \theta) = - \sum_{j=1}^{\text{norte}} \frac{\partial}{\partial x_j} \text{registro}_{\text{pag}}(\text{modelo}(x; \theta)) + \frac{1}{2} \sum_{j=1}^{\text{norte}} \frac{\partial^2}{\partial x_j^2} \text{registro}_{\text{pag}}(\text{modelo}(x; \theta))^{-2} \quad (18.25)$$

dónde norte es la dimensionalidad de X .

Debido a que el emparejamiento de puntajes requiere tomar derivadas con respecto a X , no es aplicable a modelos de datos discretos. Sin embargo, las variables latentes en el modelo pueden ser discretas.

Al igual que la pseudoverosimilitud, la coincidencia de puntajes solo funciona cuando podemos evaluar $\text{registro}_{\text{pag}}(X)$ y sus derivados directamente. No es compatible con métodos que solo proporcionan un límite inferior en $\text{registro}_{\text{pag}}(X)$, porque la coincidencia de puntuación requiere las derivadas y las segundas derivadas de $\text{registro}_{\text{pag}}(X)$ y un límite inferior no transmite información sobre sus derivados. Esto significa que la coincidencia de puntaje no se puede aplicar a modelos de estimación con interacciones complicadas entre las unidades ocultas, como modelos de codificación dispersa o máquinas de Boltzmann profundas. Si bien la coincidencia de puntajes se puede usar para entrenar previamente la primera capa oculta de un modelo más grande, no se ha aplicado como una estrategia de entrenamiento previo para las capas más profundas de un modelo más grande. Esto probablemente se deba a que las capas ocultas de tales modelos suelen contener algunas variables discretas.

Si bien la coincidencia de puntajes no tiene explícitamente una fase negativa, puede verse como una versión de divergencia contrastiva que utiliza un tipo específico de cadena de Markov (Hyvärinen, 2007a). La cadena de Markov en este caso no es un muestreo de Gibbs, sino un enfoque diferente que realiza movimientos locales guiados por el gradiente. La coincidencia de puntuación es equivalente a CD con este tipo de cadena de Markov cuando el tamaño de los movimientos locales se aproxima a cero.

lyu(2009) coincidencia de puntuación generalizada con el caso discreto (pero cometió un error en su derivación que fue corregido por Aguaja et al.(2010)). Aguaja et al. (2010) encontrado que **coincidencia de puntuación generalizada**(GSM) no funciona en espacios discretos de alta dimensión donde la probabilidad observada de muchos eventos es 0.

Un enfoque más exitoso para extender las ideas básicas de coincidencia de puntajes a datos discretos **escorrespondencia de proporciones**(Hyvärinen, 2007b). La coincidencia de proporciones se aplica específicamente a los datos binarios. El emparejamiento de razones consiste en minimizar el promedio sobre ejemplos de la siguiente función objetivo:

$$L(\text{RM})(x, \theta) = - \sum_{j=1}^{\text{norte}} \frac{1}{1 + \frac{\text{pag}_{\text{modelo}}(x; \theta)}{\text{pag}_{\text{modelo}}(F(x, j); \theta)}}^{-2}, \quad (18.26)$$

dónde $R(X_j)$ devoluciones X con la broca en posición j Florida debido a la coincidencia de razones evita la función de partición usando el mismo truco que el estimador de pseudoverosimilitud: en una razón de dos probabilidades, la función de partición se cancela. [Aguja et al. \(2010\)](#) encontraron que la coincidencia de proporciones supera a SML, pseudoverosimilitud y GSM en términos de la capacidad de los modelos entrenados con coincidencia de proporciones para eliminar el ruido de las imágenes del conjunto de prueba.

Al igual que el estimador de pseudoverosimilitud, el emparejamiento de razones requiere *n* evaluaciones de p_{ag} por punto de datos, lo que hace que su costo computacional por actualización sea aproximadamente *n* veces mayor que la de SML.

Al igual que con el estimador de pseudoverosimilitud, se puede considerar que la coincidencia de proporciones empuja hacia abajo todos los estados de fantasía que tienen solo una variable diferente de un ejemplo de entrenamiento. Dado que la coincidencia de proporciones se aplica específicamente a los datos binarios, esto significa que actúa en todos los estados de fantasía dentro de la distancia 1 de Hamming de los datos.

La coincidencia de proporciones también puede ser útil como base para tratar con datos dispersos de alta dimensión, como los vectores de conteo de palabras. Este tipo de datos plantea un desafío para los métodos basados en MCMC porque los datos son extremadamente costosos de representar en formato denso, sin embargo, el muestreador MCMC no produce valores escasos hasta que el modelo ha aprendido a representar la escasez en la distribución de datos. [Delfín y Bengio \(2013\)](#) superó este problema mediante el diseño de una aproximación estocástica no sesgada para la coincidencia de proporciones. La aproximación evalúa solo un subconjunto seleccionado aleatoriamente de los términos del objetivo y no requiere que el modelo genere muestras completas de fantasía.

Ver [Marlin y de Freitas \(2011\)](#) para un análisis teórico de la eficiencia asintótica del emparejamiento de proporciones.

18.5 Coincidencia de puntuación de eliminación de ruido

En algunos casos, es posible que deseemos regularizar el emparejamiento de puntajes, ajustando una distribución

$$p_{\text{ag}} \text{alisado}(X) = p_{\text{ag}} \text{datos}(y) q(x / y) dy \quad (18.27)$$

en lugar de la verdadera $p_{\text{ag}} \text{datos}$. La distribución $q(x / y)$ es un proceso de corrupción, generalmente uno que forma X añadiendo una pequeña cantidad de ruido y .

La coincidencia de puntajes de eliminación de ruido es especialmente útil porque, en la práctica, generalmente no tenemos acceso a la verdadera $p_{\text{ag}} \text{datos}$ sino más bien solo una distribución empírica definida por muestras de ella. Cualquier estimador consistente, dada la capacidad suficiente, hará $p_{\text{ag}} \text{modelo}$ en un conjunto de distribuciones de Dirac centradas en los puntos de entrenamiento. Suavizado por q ayuda a reducir este problema, en la pérdida de la propiedad de consistencia asintótica.

descrito en la sección 5.4.5. Kingma y Le Cun (2010) introdujeron un procedimiento para realizar el cotejo de puntaje regularizado con la distribución de suavizado q siendo ruido normalmente distribuido.

Recuperar de la sección 14.5.1 que varios algoritmos de entrenamiento de autocodificador son equivalentes a la coincidencia de puntaje o la coincidencia de puntaje de eliminación de ruido. Estos algoritmos de entrenamiento de autocodificador son, por lo tanto, una forma de superar el problema de la función de partición.

18.6 Estimación de contraste de ruido

La mayoría de las técnicas para estimar modelos con funciones de partición intratables no proporcionan una estimación de la función de partición. SML y CD estiman solo el gradiente de la función de partición de registro, en lugar de la función de partición en sí. La coincidencia de puntuación y la pseudoverosimilitud evitan por completo calcular cantidades relacionadas con la función de partición.

Estimación de contraste de ruido (NCE) (Gutmann y Hyvarinen, 2010) adopta una estrategia diferente. En este enfoque, la distribución de probabilidad estimada por el modelo se representa explícitamente como

$$\text{registro} \hat{p}_{\text{modelo}}(x) = \text{registro} p_{\text{modelo}}(X; \theta) + C, \quad (18.28)$$

dónde C se introduce explícitamente como una aproximación de $-\text{registro} Z(\theta)$. En lugar de estimar sólo θ , el procedimiento de estimación contrastiva de ruido trata C como un parámetro más y estimaciones θ y C simultáneamente, usando el mismo algoritmo para ambos. La resultante $\text{registro} \hat{p}_{\text{modelo}}(X)$ por lo tanto, puede no corresponder exactamente a una distribución de probabilidad válida, pero se acercará cada vez más a ser válida como la estimación de C mejora¹.

Tal enfoque no sería posible utilizando la máxima verosimilitud como criterio para el estimador. El criterio de máxima verosimilitud optaría por establecer C arbitrariamente alto, en lugar de establecer C para crear una distribución de probabilidad válida.

NCE funciona al reducir el problema de aprendizaje no supervisado de estimar $p_{\text{ag}}(X)$ al de aprender un clasificador binario probabilístico en el que una de las categorías corresponde a los datos generados por el modelo. Este problema de aprendizaje supervisado está construido de tal manera que la estimación de máxima verosimilitud en este aprendizaje supervisado

¹NCE también es aplicable a problemas con una función de partición manejable, donde no es necesario introducir el parámetro adicional C . Sin embargo, ha generado el mayor interés como medio para estimar modelos con funciones de partición difíciles.

problema de aprendizaje define un estimador asintóticamente consistente del problema original.

Específicamente, introducimos una segunda distribución, la **distribución de ruido** $pag_{\text{ruido}}(X)$. La distribución del ruido debe ser manejable para evaluar y tomar muestras. Ahora podemos construir un modelo sobre ambos y una nueva variable de clase binaria y. En el nuevo modelo conjunto, especificamos que

$$pag_{\text{articulación}}(y=1) = \frac{1}{2} \quad (18.29)$$

$$pag_{\text{articulación}}(X / y=1) = pag_{\text{modelo}}(X), \quad (18.30)$$

y

$$pag_{\text{articulación}}(X / y=0) = pag_{\text{ruido}}(X). \quad (18.31)$$

En otras palabras, y es una variable de cambio que determina si generaremos X del modelo o de la distribución del ruido.

Podemos construir un modelo conjunto similar de datos de entrenamiento. En este caso, la variable switch determina si dibujamos X desde el **datos** o del ruido distribución. Formalmente, $pag_{\text{tren}}(y=1) = 1/2$, $pag_{\text{tren}}(X / y=1) = pag_{\text{datos}}(X)$, y $pag_{\text{tren}}(X / y=0) = pag_{\text{ruido}}(X)$.

Ahora podemos usar el aprendizaje estándar de máxima verosimilitud en el **supervisado** problema de aprendizaje de ajuste $pag_{\text{articulación}}$ a pag_{tren} :

$$\theta, c = \underset{\theta, c}{\text{argmax}}_{mix, y=pag_{\text{tren}}} \text{registro } pag_{\text{articulación}}(y / X). \quad (18.32)$$

La distribución $pag_{\text{articulación}}$ esencialmente un modelo de regresión logística aplicado a la diferencia en las probabilidades logarítmicas del modelo y la distribución del ruido:

$$pag_{\text{articulación}}(y=1 / x) = \frac{pag_{\text{modelo}}(X)}{pag_{\text{modelo}}(X) + pag_{\text{ruido}}(X)} \quad (18.33)$$

$$= \frac{1}{1 + \frac{pag_{\text{ruido}}(X)}{pag_{\text{modelo}}(X)}} \quad (18.34)$$

$$= \frac{1}{1 + \frac{\text{experiencia}}{\frac{\text{registro } pag_{\text{ruido}}(X)}{pag_{\text{modelo}}(X)}}} - \quad (18.35)$$

$$= \sigma - \frac{pag_{\text{ruido}}(X)}{pag_{\text{modelo}}(X)} \quad (18.36)$$

$$= \sigma(\text{registro } pag_{\text{modelo}}(X) - \text{registro } pag_{\text{ruido}}(X)). \quad (18.37)$$

Por lo tanto, NCE es fácil de aplicar siempre que $\text{registro}_{\text{pag}}^{\text{modelo}}$ es fácil de propagar hacia atrás y, como se especificó anteriormente, $\text{pag}_{\text{ruido}}$ es fácil de evaluar (para evaluar pag articulación) y sample from (para generar los datos de entrenamiento).

NCE tiene más éxito cuando se aplica a problemas con pocas variables aleatorias, pero puede funcionar bien incluso si esas variables aleatorias pueden tomar una gran cantidad de valores. Por ejemplo, se ha aplicado con éxito para modelar la distribución condicional sobre una palabra dado el contexto de la palabra (Mnih y Kavukcuoglu, 2013). Aunque la palabra puede extraerse de un amplio vocabulario, solo hay una palabra.

Cuando NCE se aplica a problemas con muchas variables aleatorias, se vuelve menos eficiente. El clasificador de regresión logística puede rechazar una muestra de ruido identificando cualquier variable cuyo valor sea poco probable. Esto significa que el aprendizaje se ralentiza mucho después de $\text{pag}_{\text{modelo}}$ ha aprendido las estadísticas marginales básicas. Imagine aprender un modelo de imágenes de caras, utilizando ruido gaussiano no estructurado como $\text{pag}_{\text{ruido}}$. Si $\text{pag}_{\text{modelo}}$ aprende sobre los ojos, puede rechazar casi todas las muestras de ruido no estructuradas sin haber aprendido nada sobre otras características faciales, como la boca.

La restricción que $\text{pag}_{\text{ruido}}$ debe ser fácil de evaluar y fácil de muestrear puede ser demasiado restrictivo. Cuando $\text{pag}_{\text{ruido}}$ es simple, es probable que la mayoría de las muestras sean demasiado distintas de los datos para forzar $\text{pag}_{\text{modelo}}$ para mejorar notablemente.

Al igual que la coincidencia de puntajes y la pseudoverosimilitud, NCE no funciona si solo se establece un límite inferior en pagestá disponible. Este límite inferior podría usarse para construir un límite inferior en $\text{pag}_{\text{articulación}}(y=1/X)$, pero solo se puede usar para construir un límite superior en $\text{pag}_{\text{articulación}}(y=0/X)$, que aparece en la mitad de los términos del objetivo NCE. Asimismo, un límite inferior en $\text{pag}_{\text{ruido}}$ no es útil, porque proporciona sólo un límite superior en $\text{pag}_{\text{articulación}}(y=1/X)$.

Cuando la distribución del modelo se copia para definir una nueva distribución de ruido antes de cada paso de gradiente, NCE define un procedimiento llamado **estimación autocontrastiva**, cuyo gradiente esperado es equivalente al gradiente esperado de máxima verosimilitud (Buen compañero, 2014). El caso especial de NCE donde las muestras de ruido son las generadas por el modelo sugiere que la máxima verosimilitud puede interpretarse como un procedimiento que obliga a un modelo a aprender constantemente a distinguir la realidad de sus propias creencias en evolución, mientras que la estimación contrastiva de ruido logra un costo computacional reducido, forzando únicamente al modelo a distinguir la realidad de una línea de base fija (el modelo de ruido).

El uso de la tarea supervisada de clasificar entre muestras de entrenamiento y muestras generadas (con la función de energía del modelo utilizada para definir el clasificador) para proporcionar un gradiente en el modelo se introdujo anteriormente en varias formas (brotando et al., 2003b; bengio, 2009).

La estimación contrastiva de ruido se basa en la idea de que un buen modelo generativo debería ser capaz de distinguir los datos del ruido. Una idea estrechamente relacionada es que un buen modelo generativo debería ser capaz de generar muestras que ningún clasificador pueda distinguir de los datos. Esta idea produce redes antagónicas generativas (sección 20.10.4).

18.7 Estimación de la función de partición

Si bien gran parte de este capítulo está dedicado a describir métodos que evitan la necesidad de calcular la función de partición intratable $Z(\theta)$ asociado con un modelo gráfico no dirigido, en esta sección discutimos varios métodos para estimar directamente la función de partición.

Estimar la función de partición puede ser importante porque la necesitamos si deseamos calcular la probabilidad normalizada de los datos. Esto es a menudo importante en *evaluando* el modelo, monitoreando el rendimiento del entrenamiento y comparando modelos entre sí.

Por ejemplo, imagina que tenemos dos modelos: modelo $METRO_A$ definiendo una probabilidad distribución de la ciudad $p_{\text{page}}(X; \theta_A) = \frac{1}{Z(\theta_A)} p_{\text{page}}(X; \theta_A)$ y modelo $METRO_B$ definiendo una probabilidad distribución $p_{\text{page}}(X; \theta_B) = \frac{1}{Z(\theta_B)} p_{\text{page}}(X; \theta_B)$. Una forma común de comparar los modelos es evaluar y comparar la probabilidad que ambos modelos asignan a un iid conjunto de datos de prueba. Supongamos que el conjunto de prueba consta de m ejemplos $\{X^{(1)}, \dots, X^{(m)}\}$. Si $i p_{\text{page}}(X^{(i)}; \theta_A) > i p_{\text{page}}(X^{(i)}; \theta_B)$ o equivalentemente si

$$\sum_i \text{registro}_{\text{page}}^A(X^{(i)}; \theta_A) - \sum_i \text{registro}_{\text{page}}^B(X^{(i)}; \theta_B) > 0, \quad (18.38)$$

entonces decimos que $METRO_A$ es mejor modelo que $METRO_B$ (o, al menos, es un mejor modelo del conjunto de prueba), en el sentido de que tiene una mejor probabilidad de registro de prueba. Desafortunadamente, probar si esta condición se cumple requiere conocimiento de la función de partición. Desafortunadamente, la ecuación 18.38 parece requerir evaluar la probabilidad logarítmica que el modelo asigna a cada punto, lo que a su vez requiere evaluar la función de partición. Podemos simplificar un poco la situación reorganizando la ecuación 18.38 en una forma en la que solo necesitamos saber la relación de las funciones de partición de los dos modelos:

$$\sum_i \text{registro}_{\text{page}}^A(X^{(i)}; \theta_A) - \sum_i \text{registro}_{\text{page}}^B(X^{(i)}; \theta_B) = \sum_i \frac{\text{page}^A(X^{(i)}; \theta_A)}{\text{page}^B(X^{(i)}; \theta_B)} - m \sum_i \frac{Z(\theta_A)}{Z(\theta_B)} \quad (18.39)$$

Así podemos determinar si $METRO_A$ es mejor modelo que $METRO_B$ sin conocer la función de partición de ninguno de los modelos sino solo su relación. Como veremos en breve, podemos estimar esta razón utilizando un muestreo de importancia, siempre que los dos modelos sean similares.

Sin embargo, si quisieramos calcular la probabilidad real de los datos de prueba bajo cualquiera $METRO_A$ o $METRO_B$, necesitaríamos calcular el valor real de las funciones de partición. Dicho esto, si conocieramos la razón de dos funciones de partición, $r = \frac{Z(\theta_B)}{Z(\theta_A)}$, y sabíamos el valor real de solo uno de los dos, digamos $Z(\theta_A)$, podríamos calcular el valor del otro:

$$Z(\theta_B) = rZ(\theta_A) = \frac{Z(\theta_B)}{Z(\theta_A)} Z(\theta_A) \quad (18.40)$$

Una forma sencilla de estimar la función de partición es utilizar un método Monte Carlo como el muestreo de importancia simple. Presentamos el enfoque en términos de variables continuas usando integrales, pero se puede aplicar fácilmente a variables discretas reemplazando las integrales con sumatoria. Usamos una distribución propuesta $p_{\text{prop}}(x) = \frac{p_{\text{real}}(x)}{\int p_{\text{real}}(X) dX}$ que admite el muestreo manejable y la evaluación manejable de tanto la función de partición Z y la distribución no normalizada $p_{\text{real}}(X)$.

$$Z_1 = \int p_{\text{real}}(X) dX \quad (18.41)$$

$$= \int \frac{p_{\text{real}}(X)}{\int p_{\text{real}}(X) dX} p_{\text{prop}}(X) dX \quad (18.42)$$

$$= Z_0 \int p_{\text{real}}(X) \frac{p_{\text{prop}}(X)}{\int p_{\text{real}}(X) dX} dX \quad (18.43)$$

$$\hat{Z}_1 = \frac{Z_0}{k} \sum_{k=1}^K \frac{p_{\text{real}}(X_{(k)})}{p_{\text{prop}}(X_{(k)})} \quad \text{s.t.: } X_{(k)} \sim p_{\text{real}} \quad (18.44)$$

En la última línea, hacemos un estimador de Monte Carlo, \hat{Z}_1 , de la integral utilizando muestras extraídas de $p_{\text{real}}(X)$ y luego ponderar cada muestra con la proporción de los no normalizados p_{real} y la propuesta p_{prop} .

Vemos también que este enfoque nos permite estimar la relación entre las funciones de partición como

$$\frac{1}{k} \sum_{k=1}^K \frac{p_{\text{real}}(X_{(k)})}{p_{\text{prop}}(X_{(k)})} \quad \text{s.t.: } X_{(k)} \sim p_{\text{real}}. \quad (18.45)$$

Este valor se puede usar directamente para comparar dos modelos como se describe en la ecuación 18.39.

Si la distribución $pago$ está cerca de pag_1 , ecuación 18.44 puede ser una forma efectiva de estimar la función de partición (Minka, 2005). Desafortunadamente, la mayor parte del tiempo pag es a la vez complicado (generalmente multimodal) y definido en un espacio dimensional alto. Es difícil encontrar un tratable $pago$ que es lo suficientemente simple para evaluar sin dejar de estar lo suficientemente cerca de pag_1 para dar como resultado una aproximación de alta calidad. Si pag y pag_1 están cerca, la mayoría de las muestras de pag tendrá baja probabilidad bajo pag_1 y por lo tanto hacen una contribución (relativamente) insignificante a la suma en la ecuación 18.44.

Tener pocas muestras con pesos significativos en esta suma dará como resultado un estimador de mala calidad debido a la alta varianza. Esto se puede entender cuantitativamente a través de una estimación de la varianza de nuestra estimación \hat{Z}_1 :

$$\text{Var} \hat{Z}_1 = \frac{\sum_{k=1}^n \frac{Z_0^{pag}}{k^2} - \frac{1(X^{(k)})}{pago(X^{(k)})} - \hat{Z}_1}{\overline{pago(X^{(k)})}}^{-2}. \quad (18.46)$$

Esta cantidad es mayor cuando hay una desviación significativa en los valores de los pesos de importancia $pago(X^{(k)})$.

Pasamos ahora a dos estrategias relacionadas desarrolladas para hacer frente a la desafiante tarea de estimar funciones de partición para distribuciones complejas sobre espacios de alta dimensión: muestreo de importancia recocido y muestreo puente. Ambos comienzan con la estrategia de muestreo de importancia simple presentada anteriormente y ambos intentan superar el problema de la propuesta $pago$ estar demasiado lejos de pag_1 introduciendo distribuciones intermedias que intentan *cubrir la brecha* entre $pago$ y pag_1 .

18.7.1 Muestreo de importancia recocido

En situaciones donde $D_{KL}(pago-pag_1)$ es grande (es decir, donde hay poca superposición entre $pago$ y pag_1), una estrategia llamada **muestreo de importancia recocido** (AIS) intenta cerrar la brecha mediante la introducción de distribuciones intermedias (Jarzynski, 1997; Neal, 2001). Considere una secuencia de distribuciones $pago_0, \dots, pago_{norte}$, con $0 = \eta_0 < \eta_1 < \dots < \eta_{n-1} < \eta_{norte} = 1$ de modo que la primera y la última distribución de la secuencia son $pago$ y pag_1 respectivamente.

Este enfoque nos permite estimar la función de partición de una distribución multimodal definida en un espacio de alta dimensión (como la distribución definida por un RBM entrenado). Comenzamos con un modelo más simple con una función de partición conocida (como un RBM con ceros para los pesos) y estimamos la relación entre las funciones de partición de los dos modelos. La estimación de esta relación se basa en la estimación de las relaciones de una secuencia de muchas distribuciones similares, como la secuencia de RBM con pesos interpolados entre cero y los pesos aprendidos.

Ahora podemos escribir la razón $\frac{Z_1}{Z_0}$ como

$$\frac{Z_1}{Z_0} = \frac{Z_1}{Z_0} \frac{Z_{\eta_1}}{Z_{\eta_1}} \cdots \frac{Z_{\eta_{n-1}}}{Z_{\eta_{n-1}}} \quad (18.47)$$

$$= \frac{Z_{\eta_1}}{Z_0 Z_{\eta_1}} \frac{Z_{\eta_2}}{Z_{\eta_2} Z_{\eta_1}} \cdots \frac{Z_{\eta_{n-1}}}{Z_{\eta_{n-2}} Z_{\eta_{n-1}}} \frac{Z_1}{Z_{\eta_{n-1}}} \quad (18.48)$$

$$= \prod_{j=0}^{D_f} \frac{Z_{\eta_{j+1}}}{Z_{\eta_j}} \quad (18.49)$$

siempre que las distribuciones p_{η_j} y $p_{\eta_{j+1}}$, para todos $0 \leq j \leq n-1$, son lo suficientemente close, podemos estimar confiablemente cada uno de los factores $\frac{Z_{\eta_{j+1}}}{Z_{\eta_j}}$ usando un muestreo de importancia simple y luego utilícelos para obtener una estimación de $\frac{Z_1}{Z_0}$.

¿De dónde vienen estas distribuciones intermedias? Así como la distribución de la propuesta original p_{η_0} es una elección de diseño, también lo es la secuencia de distribuciones $p_{\eta_1}, \dots, p_{\eta_{n-1}}$. Es decir, puede construirse específicamente para adaptarse al dominio del problema. Una opción popular y de propósito general para las distribuciones intermedias es usar el promedio geométrico ponderado de la distribución objetivo p_{η_1} y la distribución propuesta inicial (para la cual se conoce la función de partición) p_{η_0} :

$$p_{\eta_1} = p_{\eta_0} p_{\eta_1} / p_{\eta_0} \quad (18.50)$$

Para tomar muestras de estas distribuciones intermedias, definimos una serie de funciones de transición de cadena de Markov $T_{\eta}(X' | X)$ que definen la distribución de probabilidad condicional de la transición a X' dado que actualmente estamos en X . El operador de transición $T_{\eta}(X' | X)$ se define para salir $p_{\eta_1}(X')$ invariante:

$$p_{\eta_1}(X') = \int p_{\eta_1}(X) T_{\eta}(X' | X) dX \quad (18.51)$$

Estas transiciones se pueden construir como cualquier método Monte Carlo de cadena de Markov (por ejemplo, Metropolis-Hastings, Gibbs), incluidos métodos que involucran múltiples pasos a través de todas las variables aleatorias u otros tipos de iteraciones.

La estrategia de muestreo de AIS es entonces generar muestras a partir de p_{η_0} y luego use los operadores de transición para generar secuencialmente muestras de las distribuciones intermedias hasta que lleguemos a las muestras de la distribución objetivo p_{η_1} :

- para $k=1 \dots k$
 - Muestra $X_k \sim p_{\eta_k}(X)$

- Muestra $X_{\eta_2} \sim T_{\eta_1}(X_{\eta_1})_{\eta_2}^k / X_{\eta_1}^{(k)}$
- ...
- Muestra $X_{\eta_{n-1}} \sim T_{\eta_{n-2}}(X_{\eta_{n-1}})_{\eta_{n-1}}^{(k)} / X_{\eta_{n-2}}^{(k)}$
- Muestra $X_{\eta_{norte}} \sim T_{\eta_{n-1}}(X_{\eta_{norte}})_{\eta_{n-1}}^{(k)} / X_{\eta_{n-1}}^{(k)}$
- fin

para muestra k , podemos derivar el peso de importancia encadenando los pesos de importancia para los saltos entre las distribuciones intermedias dadas en ecuación 18.49:

$$w(k) = \frac{pag(X_{\eta_1})}{pag(X_{\eta_0})} \frac{pag_{\eta_2}(X_{\eta_1})_{\eta_2}^{(k)}}{pag_{\eta_1}(X_{\eta_2})^{(k)}} \dots \frac{pag(X_{\eta_n})}{pag(X_{\eta_{norte}})}. \quad (18.52)$$

Para evitar problemas numéricos como el desbordamiento, probablemente sea mejor calcular registro $w(k)$ sumando y restando probabilidades logarítmicas, en lugar de calcular $w(k)$ multiplicando y dividiendo probabilidades.

Con el procedimiento de muestreo así definido y los pesos de importancia dados en la ecuación 18.52, la estimación de la razón de las funciones de partición viene dada por:

$$\frac{Z_1}{Z_0} \approx \frac{1}{k} \sum_{k=1}^K w(k) \quad (18.53)$$

Para verificar que este procedimiento define un esquema de muestreo de importancia válido, podemos mostrar (Neal, 2001) que el procedimiento AIS corresponde a un muestreo de importancia simple en un espacio de estado extendido con puntos muestreados en el espacio del producto $[X_{\eta_1}, \dots, X_{\eta_{n-1}}, X_1]$. Para ello, definimos la distribución sobre el espacio extendido como:

$$pag(X_{\eta_1}, \dots, X_{\eta_{n-1}}, X_1) \quad (18.54)$$

$$= pag_1(X_1) T_{\eta_{n-1}}(X_{\eta_{n-1}} / X_1) T_{\eta_{n-2}}(X_{\eta_{n-2}} / X_{\eta_{n-1}}) \dots T_{\eta_1}(X_{\eta_1} / X_{\eta_2}), \quad (18.55)$$

dónde T_a es el reverso del operador de transición definido por T_a (mediante una aplicación de la regla de Bayes):

$$T_a(X_{-} / X) = \frac{pag(X)}{pag_a(X)} T_a(X / X) = \frac{\tilde{pag}(X)}{pag_a(X)} T_a(X / X). \quad (18.56)$$

Reemplazando lo anterior en la expresión de la distribución conjunta en el espacio de estado extendido dado en la ecuación 18.55, obtenemos:

$$pag(X_{\eta_1}, \dots, X_{\eta_{n-1}}, X_1) \quad (18.57)$$

$$= pag_1(X_1) \frac{pag_{\eta_{n-1}}(X_{\eta_{n-1}})}{pag_{\eta_{n-1}}(X_1)} T_{\eta_{n-1}}(X_1 / X_{\eta_{n-1}}) \frac{\sum_{j=1}^{\eta_n} pag_j(X_{\eta_j})}{\sum_{j=1}^{\eta_n} pag_j(X_{\eta_j})} T_{\eta}(X_{\eta+1} / X_{\eta}) \quad (18.58)$$

$$= \frac{pag_1(X_1)}{pag_{\eta_{n-1}}(X_1)} T_{\eta_{n-1}}(X_1 / X_{\eta_{n-1}}) pag_{\eta}(X_{\eta}) \frac{\sum_{j=1}^{\eta_n} pag_{\eta+1}(X_{\eta+1})}{\sum_{j=1}^{\eta_n} pag_{\eta}(X_{\eta+1})} T_{\eta}(X_{\eta+1} / X_{\eta}). \quad (18.59)$$

Ahora tenemos medios para generar muestras a partir de la distribución de propuestas conjuntas. *q*sobre la muestra extendida a través de un esquema de muestreo dado anteriormente, con la distribución conjunta dada por:

$$q(X_{\eta_1}, \dots, X_{\eta_{n-1}}, X_1) = pag_0(X_{\eta_1}) T_{\eta_1}(X_{\eta_2} / X_{\eta_1}) \dots T_{\eta_{n-1}}(X_1 / X_{\eta_{n-1}}). \quad (18.60)$$

Tenemos una distribución conjunta en el espacio extendido dada por la ecuación 18.59. Tomando $q(X_{\eta_1}, \dots, X_{\eta_{n-1}}, X_1)$ como la propuesta de distribución sobre el espacio de estados extendido del que tomaremos muestras, queda por determinar los pesos de importancia:

$$w(k) = \frac{pag(X_{\eta_1}, \dots, X_{\eta_{n-1}}, X_1)}{q(X_{\eta_1}, \dots, X_{\eta_{n-1}}, X_1)} = \frac{pag(X_{\eta_1}^{(k)})}{pag(X_{\eta_{n-1}}^{(k)})} \dots \frac{pag(X_{\eta_n}^{(k)})}{pag(X_{\eta_1}^{(k)})} \frac{pag_{\eta}(X_{\eta}^{(k)})}{pag_{\eta}(X_{\eta_1}^{(k)})} \quad (18.61)$$

Estos pesos son los mismos que los propuestos para AIS. Por lo tanto, podemos interpretar AIS como un muestreo de importancia simple aplicado a un estado extendido y su validez se deriva inmediatamente de la validez del muestreo de importancia.

El muestreo de importancia recocido (AIS) fue descubierto por primera vez por [Jarzynski \(1997\)](#) y luego otra vez, independientemente, por [Neal \(2001\)](#). Actualmente es la forma más común de estimar la función de partición para modelos probabilísticos no dirigidos. Las razones de esto pueden tener más que ver con la publicación de un artículo influyente ([Salakhutdinov y Murray, 2008](#)) que describe su aplicación para estimar la función de partición de máquinas de Boltzmann restringidas y redes de creencias profundas que con cualquier ventaja inherente que el método tenga sobre el otro método descrito a continuación.

Se puede encontrar una discusión de las propiedades del estimador AIS (por ejemplo, su varianza y eficiencia) en [Neal \(2001\)](#).

18.7.2 Muestreo de puente

Muestreo de puente [bennett \(1976\)](#) es otro método que, como AIS, aborda las deficiencias del muestreo de importancia. En lugar de encadenar una serie de

distribuciones intermedias, el muestreo puente se basa en una única distribución pag^* , conocido como el puente, para interpolar entre una distribución con función de partición conocida, pag_0 , y una distribución pag^1 para la cual estamos tratando de estimar la función de partición Z_1 .

El muestreo de puente estima la relación Z_1/Z_0 como el cociente de los pesos de importancia esperados entre pag_0 y pag^* y entre pag^1 y pag^* :

$$\frac{Z_1}{Z_0} \approx \frac{\int k \text{pag}_0(X(k))}{\int k \text{pag}^1(X(k))} = \frac{\int k \text{pag}^*(X(k))}{\int k \text{pag}^1(X(k))} \quad (18.62)$$

Si la distribución del puente pag^* se elige cuidadosamente para tener una gran superposición de soporte con ambas pag_0 y pag^1 , entonces el muestreo puente puede permitir la distancia entre dos distribuciones (o más formalmente, $D_{KL}(\text{pag}_0, \text{pag}^1)$) sea mucho mayor que con el muestreo de importancia estándar.

Se puede demostrar que la distribución puente óptima está dada por $\text{pag}^{(\text{optimal})}(X) \propto \frac{\text{pag}_0(X)\text{pag}_1(X)}{r\text{pag}_0(X)+\text{pag}_1(X)}$ dónde $r = Z/Z_1 > 0$. Al principio, esto parece ser una solución inviable, como parecería requerir la misma cantidad que estamos tratando de estimar, Z_1/Z_0 . Sin embargo, es posible comenzar con una estimación aproximada de r y usamos la distribución puente resultante para refinar nuestra estimación iterativamente ([Neal, 2005](#)). Es decir, reestimamos iterativamente la razón y usamos cada iteración para actualizar el valor de r .

Muestreo de importancia ligada Tanto el muestreo AIS como el puente tienen sus ventajas. Si $D_{KL}(\text{pag}_0, \text{pag}^1)$ no es demasiado grande (porque pag_0 y pag^1 están lo suficientemente cerca) el muestreo puente puede ser un medio más eficaz para estimar la relación de las funciones de partición que el AIS. Sin embargo, si las dos distribuciones están demasiado separadas para una sola distribución pag^* para cerrar la brecha, al menos se puede usar AIS con potencialmente muchas distribuciones intermedias para abarcar la distancia entre pag_0 y pag^1 . [Neal \(2005\)](#) mostró cómo su método de muestreo de importancia vinculada aprovechó el poder de la estrategia de muestreo puente para unir las distribuciones intermedias utilizadas en AIS para mejorar significativamente las estimaciones generales de la función de partición.

Estimación de la función de partición durante el entrenamiento Si bien AIS se ha aceptado como el método estándar para estimar la función de partición para muchos modelos no dirigidos, es lo suficientemente intensivo desde el punto de vista computacional que sigue siendo inviable de usar durante el entrenamiento. Sin embargo, las estrategias alternativas que se han explorado para mantener una estimación de la función de partición a lo largo del entrenamiento

Usando una combinación de muestreo de puente, AIS de cadena corta y templado paralelo, [Desjardin et al. \(2011\)](#) ideó un esquema para rastrear la función de partición de un

RBM durante todo el proceso de formación. La estrategia se basa en el mantenimiento de estimaciones independientes de las funciones de partición de la RBM a cada temperatura que opera en el esquema de templado paralelo. Los autores combinaron estimaciones de muestreo puente de las proporciones de las funciones de partición de las cadenas vecinas (es decir, del templado en paralelo) con estimaciones de AIS a lo largo del tiempo para obtener una estimación de baja varianza de las funciones de partición en cada iteración de aprendizaje.

Las herramientas descritas en este capítulo proporcionan muchas formas diferentes de superar el problema de las funciones de partición intratables, pero puede haber varias otras dificultades involucradas en el entrenamiento y el uso de modelos generativos. El principal de ellos es el problema de la inferencia intratable, que confrontamos a continuación.

capítulo 19

Inferencia aproximada

Muchos modelos probabilísticos son difíciles de entrenar porque es difícil realizar inferencias en ellos. En el contexto del aprendizaje profundo, generalmente tenemos un conjunto de variables visibles v y un conjunto de variables latentes h . El desafío de la inferencia generalmente se refiere al difícil problema de computar $\text{pag}(h \mid v)$ o tomando expectativas con respecto a ella. Estas operaciones suelen ser necesarias para tareas como el aprendizaje de máxima probabilidad.

Muchos modelos gráficos simples con una sola capa oculta, como las máquinas de Boltzmann restringidas y el PCA probabilístico, se definen de una manera que hace que las operaciones de inferencia como la informática $\text{pag}(h \mid v)$, o tomando expectativas con respecto a ella, simple. Desafortunadamente, la mayoría de los modelos gráficos con múltiples capas de variables ocultas tienen distribuciones posteriores intratables. La inferencia exacta requiere una cantidad exponencial de tiempo en estos modelos. Incluso algunos modelos con una sola capa, como la codificación escasa, tienen este problema.

En este capítulo, presentamos varias de las técnicas para enfrentar estos problemas de inferencia intratables. Posteriormente, en el capítulo 20, describiremos cómo usar estas técnicas para entrenar modelos probabilísticos que de otro modo serían intratables, como las redes de creencias profundas y las máquinas profundas de Boltzmann.

Los problemas de inferencia intratables en el aprendizaje profundo generalmente surgen de interacciones entre variables latentes en un modelo gráfico estructurado. Ver figura 19.1 para algunos ejemplos. Estas interacciones pueden deberse a interacciones directas en modelos no dirigidos o interacciones de "explicación" entre ancestros mutuos de la misma unidad visible en modelos dirigidos.

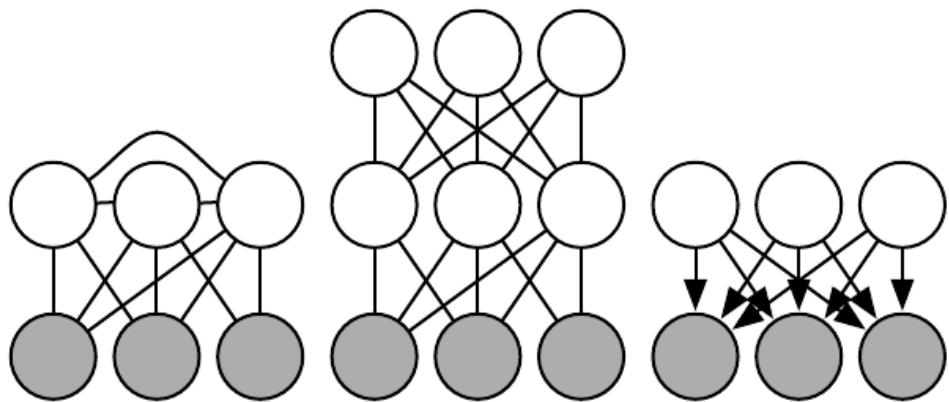


Figura 19.1: Los problemas de inferencia intratables en el aprendizaje profundo suelen ser el resultado de interacciones entre variables latentes en un modelo gráfico estructurado. Estos pueden deberse a los bordes que conectan directamente una variable latente con otra, o debido a caminos más largos que se activan cuando se observa el hijo de una estructura en V.(Izquierda) **Máquina semi-restringida de Boltzmann**(Osindero y Hinton,2008) con conexiones entre unidades ocultas. Estas conexiones directas entre las variables latentes hacen que la distribución posterior sea intratable debido a las grandes camarillas de variables latentes.(Centro)Una máquina profunda de Boltzmann, organizada en capas de variables sin conexiones entre capas,(Derecha)Este modelo dirigido tiene interacciones entre las variables latentes cuando se observan las variables visibles, porque cada dos variables latentes son copadres. Algunos modelos probabilísticos pueden proporcionar una inferencia manejable sobre las variables latentes a pesar de tener una de las estructuras gráficas que se muestran arriba. Esto es posible si las distribuciones de probabilidad condicional se eligen para introducir independencias adicionales más allá de las descritas en el gráfico. Por ejemplo, el PCA probabilístico tiene la estructura gráfica que se muestra a la derecha, pero todavía tiene una inferencia simple debido a las propiedades especiales de las distribuciones condicionales específicas que utiliza (condicionales lineales de Gauss con vectores de base ortogonales entre sí).

19.1 Inferencia como optimización

Muchos enfoques para enfrentar el problema de la inferencia difícil hacen uso de la observación de que la inferencia exacta puede describirse como un problema de optimización. Entonces se pueden derivar algoritmos de inferencia aproximados approximando el problema de optimización subyacente.

Para construir el problema de optimización, supongamos que tenemos un modelo probabilístico que consta de variables observadas v y variables latentes h . Nos gustaría calcular la probabilidad logarítmica de los datos observados, $\text{registro} \text{pag}(v; \theta)$. A veces es muy difícil calcular $\text{registro} \text{pag}(v; \theta)$ si es costoso marginar h . En su lugar, podemos calcular un límite inferior $L(v, \theta, q)$ en $\text{registro} \text{pag}(v; \theta)$. Este límite se llama **límite inferior de evidencia** (ELBO). Otro nombre comúnmente usado para este límite inferior es el negativo **energía libre variacional**. Específicamente, el límite inferior de la evidencia se define como

$$L(v, \theta, q) = \text{registro} \text{pag}(v; \theta) - D_{\text{KL}}(q(h / v) \text{-} \text{pag}(h / v; \theta)) \quad (19.1)$$

dónde q es una distribución de probabilidad arbitraria sobre h .

Porque la diferencia entre $\text{registro} \text{pag}(v)$ y $L(v, \theta, q)$ está dada por la divergencia KL y debido a que la divergencia KL siempre es no negativa, podemos ver que L siempre tiene como máximo el mismo valor que la probabilidad logarítmica deseada. Los dos son iguales si y solo si q es la misma distribución que $\text{pag}(h / v)$.

Asombrosamente, L puede ser considerablemente más fácil de calcular para algunas distribuciones q . El álgebra simple muestra que podemos reorganizar L en una forma mucho más conveniente:

$$L(v, \theta, q) = \text{registro} \text{pag}(v; \theta) - D_{\text{KL}}(q(h / v) \text{-} \text{pag}(h / v; \theta)) \quad (19.2)$$

$$= \text{registro} \text{pag}(v; \theta) - \text{mi}_{h \sim q} \text{registro} \frac{q(h / v)}{\text{pag}(h / v)} \quad (19.3)$$

$$= \text{registro} \text{pag}(v; \theta) - \text{mi}_{h \sim q} \text{registro} \frac{q(h / v)}{\frac{\text{pag}(h, v; \theta)}{\text{pag}(v; \theta)}} \quad (19.4)$$

$$= \text{registro} \text{pag}(v; \theta) - \text{mi}_{h \sim q} [\text{registro} q(h / v) - \text{registro} \text{pag}(h, v; \theta) + \text{registro} \text{pag}(v; \theta)] \quad (19.5)$$

$$= -\text{mi}_{h \sim q} [\text{registro} q(h / v) - \text{registro} \text{pag}(h, v; \theta)]. \quad (19.6)$$

Esto produce la definición más canónica del límite inferior de evidencia,

$$L(v, \theta, q) = \text{mi}_{h \sim q} [\text{registro} \text{pag}(h, v)] + H(q). \quad (19.7)$$

Para una adecuada elección de q , L es manejable para calcular. Para cualquier elección de q , L proporciona un límite inferior de la probabilidad. Para $q(h / v)$ que son mejores

aproximaciones de $q(h / v)$, el límite inferior L será más estrecho, es decir, más próximo a $\text{registro}q(v)$. Cuando $q(h / v) = \text{pag}(h / v)$, la aproximación es perfecta, y $L(v, \theta, q) = \text{registro}q(v, \theta)$.

Por lo tanto, podemos pensar en la inferencia como el procedimiento para encontrar la que maximiza L . La inferencia exacta maximiza L perfectamente buscando en una familia de funciones q que incluye $\text{pag}(h / v)$. A lo largo de este capítulo, mostraremos cómo derivar diferentes formas de inferencia aproximada usando optimización aproximada para encontrar q . Podemos hacer que el procedimiento de optimización sea menos costoso pero aproximado restringiendo la familia de distribuciones que la optimización puede buscar sobre o mediante un procedimiento de optimización imperfecto que puede no maximizar completamente L sino simplemente aumentarlo en una cantidad significativa.

No importa qué elección de q usamos, L es un límite inferior. Podemos obtener límites más estrechos o más flexibles que son más baratos o más caros de calcular dependiendo de cómo decidamos abordar este problema de optimización. Podemos obtener un emparejamiento pobre pero reduzca el costo computacional usando un procedimiento de optimización imperfecto, o usando un procedimiento de optimización perfecto sobre una familia restringida de q distribuciones.

19.2 Maximización de expectativas

El primer algoritmo que presentamos basado en maximizar un límite inferior L es el **maximización de expectativas** (EM), un algoritmo de entrenamiento popular para modelos con variables latentes. Describimos aquí una vista sobre el algoritmo EM desarrollado por [Neal y Hinton \(1999\)](#). A diferencia de la mayoría de los otros algoritmos que describimos en este capítulo, EM no es un enfoque para la inferencia aproximada, sino un enfoque para el aprendizaje con un posterior aproximado.

El algoritmo EM consiste en alternar entre dos pasos hasta la convergencia:

- **El E-paso** (Paso de expectativa): Let $\theta(0)$ denote el valor de los parámetros al comienzo del paso. Colocar $q(h_i / v) = \text{pag}(h_i / v; \theta(0))$ para todos los índices i de los ejemplos de entrenamiento v_i en el que queremos entrenar (tanto las variantes de lotes como las de minibatches son válidas). Con esto queremos decir que se define en términos de *actual* valor del parámetro de $\theta(0)$; si variamos θ entonces $\text{pag}(h / v; \theta)$ cambiará pero $q(h / v)$ permanecerá igual a $\text{pag}(h / v; \theta(0))$.
- **El paso M** (Paso de maximización): maximizar total o parcialmente

$$\frac{1}{n} \sum_i L(v_i, \theta, q) \quad (19.8)$$

con respecto a θ utilizando el algoritmo de optimización de su elección.

Esto se puede ver como un algoritmo de ascenso de coordenadas para maximizar L . En un paso, maximizamos L con respecto a q , y por otro, maximizamos L con respecto a θ .

El ascenso de gradiente estocástico en modelos de variables latentes puede verse como un caso especial del algoritmo EM donde el paso M consiste en tomar un solo paso de gradiente. Otras variantes del algoritmo EM pueden dar pasos mucho más grandes. Para algunas familias de modelos, el paso M puede incluso realizarse analíticamente, saltando hasta la solución óptima para θ dada la corriente q .

Aunque el paso E implica una inferencia exacta, podemos pensar que el algoritmo EM utiliza una inferencia aproximada en algún sentido. Específicamente, el paso M asume que el mismo valor de q puede usarse para todos los valores de θ . Esto introducirá una brecha entre L y el verdadero registro $p_{\theta}(v)$ medida que el paso M se aleja más y más del valor θ_0 utilizado en el paso E. Afortunadamente, el paso E reduce la brecha a cero nuevamente cuando ingresamos al ciclo para la próxima vez.

El algoritmo EM contiene algunas ideas diferentes. Primero, está la estructura básica del proceso de aprendizaje, en el que actualizamos los parámetros del modelo para mejorar la probabilidad de un conjunto de datos completo, donde todas las variables que faltan tienen sus valores proporcionados por una estimación de la distribución posterior. Esta percepción particular no es exclusiva del algoritmo EM. Por ejemplo, usar el descenso de gradiente para maximizar la probabilidad logarítmica también tiene esta misma propiedad; los cálculos del gradiente de verosimilitud logarítmica requieren tomar expectativas con respecto a la distribución posterior sobre las unidades ocultas. Otra idea clave en el algoritmo EM es que podemos continuar usando un valor de q incluso después de haber pasado a un valor diferente de θ . Este conocimiento particular se utiliza en todo el aprendizaje automático clásico para derivar grandes actualizaciones de M-step. En el contexto del aprendizaje profundo, la mayoría de los modelos son demasiado complejos para admitir una solución manejable para una actualización óptima de pasos M grandes, por lo que esta segunda idea, que es más exclusiva del algoritmo EM, rara vez se usa.

19.3 Inferencia MAP y codificación dispersa

Usualmente usamos el término inferencia para referirnos a calcular la distribución de probabilidad sobre un conjunto de variables dado otro. Cuando entrenamos modelos probabilísticos con variables latentes, generalmente nos interesa calcular $p_{\theta}(h / v)$. Una forma alternativa de inferencia es calcular el único valor más probable de las variables que faltan, en lugar de inferir la distribución completa sobre sus valores posibles. En el contexto

de modelos de variables latentes, esto significa computar

$$h = \underset{h}{\text{argmax}} \text{registro } pag(h / v). \quad (19.9)$$

Esto se conoce como **máximo a posteriori** inferencia, inferencia MAP abreviada.

La inferencia MAP generalmente no se considera una inferencia aproximada: calcula el valor exacto más probable de h^* . Sin embargo, si deseamos desarrollar un proceso de aprendizaje basado en maximizar $L(v, h, q)$, entonces es útil pensar en la inferencia MAP como un procedimiento que proporciona un valor de q . En este sentido, podemos pensar en la inferencia MAP como una inferencia aproximada, ya que no proporciona la inferencia óptima. q .

Recuperar de la sección 19.1 que la inferencia exacta consiste en maximizar

$$L(v, \theta, q) = \text{mi}_{h \sim q} [\text{registro } pag(h, v)] + H(q) \quad (19.10)$$

con respecto a q sobre una familia no restringida de distribuciones de probabilidad, usando un algoritmo de optimización exacto. Podemos derivar la inferencia MAP como una forma de inferencia aproximada restringiendo la familia de distribuciones q que puede extraerse de. Específicamente, requerimos q para asumir una distribución de Dirac:

$$q(h / v) = d(h - \mu). \quad (19.11)$$

Esto significa que ahora podemos controlar q completamente a través de μ . Eliminación de los términos de L que no varían con μ , nos quedamos con el problema de optimización

$$\mu = \underset{\mu}{\text{registro máximo de argumento}} pag(h=\mu, v), \quad (19.12)$$

que es equivalente al problema de inferencia MAP

$$h = \underset{h}{\text{argmax}} \text{registro } pag(h / v). \quad (19.13)$$

Por lo tanto, podemos justificar un procedimiento de aprendizaje similar a EM, en el que alternamos entre realizar la inferencia MAP para inferir h^* y luego actualizar θ aumentar registro $pag(h^*, v)$. Al igual que con EM, esta es una forma de ascenso coordinado en L , donde alternamos entre usar la inferencia para optimizar L con respecto a q y el uso de actualizaciones de parámetros para optimizar L con respecto a θ . El procedimiento en su conjunto puede justificarse por el hecho de que L es un límite inferior en registro $pag(v)$. En el caso de la inferencia MAP, esta justificación es bastante vacía, porque el límite es infinitamente flexible, debido a la entropía diferencial de infinito negativo de la distribución de Dirac. Sin embargo, agregar ruido a μ haría que el límite volviera a tener sentido.

La inferencia MAP se usa comúnmente en el aprendizaje profundo como un extractor de características y un mecanismo de aprendizaje. Se utiliza principalmente para modelos de codificación dispersa.

Recuperar de la sección 13.4 que la codificación escasa es un modelo de factor lineal que impone un previo inductor de escasez en sus unidades ocultas. Una elección común es un factorial de Laplace anterior, con

$$pag(h) = \frac{\lambda}{2} m_i - \lambda |h_i|. \quad (19.14)$$

Luego, las unidades visibles se generan realizando una transformación lineal y agregando ruido:

$$pag(x / h) = norte(v, \{Qué? + b, \beta\}). \quad (19.15)$$

Computar o incluso representar $pag(h / v)$ es difícil. Cada par de variables h_i y h_j ambos son padres de v . Esto significa que cuando se observa, el modelo gráfico contiene un camino activo que conecta h_i y h_j . Todas las unidades ocultas participan así en una camarilla masiva en $pag(h / v)$. Si el modelo fuera gaussiano, estas interacciones se podrían modelar de manera eficiente a través de la matriz de covarianza, pero la escasez previa hace que estas interacciones no sean gaussianas.

Porque $pag(h / v)$ es intratable, al igual que el cálculo de la verosimilitud logarítmica y su gradiente. Por lo tanto, no podemos utilizar el aprendizaje exacto de máxima verosimilitud. En cambio, usamos la inferencia MAP y aprendemos los parámetros maximizando el ELBO definido por la distribución de Dirac alrededor de la estimación MAP de h .

Si concatenamos todos los h vectores en el conjunto de entrenamiento en una matriz H y concatenar todos los v vectores en una matriz V , entonces el proceso de aprendizaje de codificación dispersa consiste en minimizar

$$\mathcal{J}(\text{alto, ancho}) = \sum_{yo,j} \left(\frac{|H_{yo,j}|^2}{2} + \frac{(V - HW)_{yo,j}^2}{2} \right). \quad (19.16)$$

La mayoría de las aplicaciones de codificación dispersa también implican una disminución del peso o una restricción en las normas de las columnas de W , con el fin de prevenir la solución patológica con extremadamente pequeño H y largo W .

Podemos minimizar \mathcal{J} alternando entre minimización con respecto a H y minimización con respecto a W . Ambos subproblemas son convexos. De hecho, la minimización con respecto a W es solo un problema de regresión lineal. Sin embargo, la minimización de J con respecto a ambos argumentos no suele ser un problema convexo.

Minimización con respecto a H requiere algoritmos especializados como el algoritmo de búsqueda de signos de características (Sotavento et al., 2007).

19.4 Inferencia variacional y aprendizaje

Hemos visto cómo el límite inferior de la evidencia $L(v, \theta, q)$ es un límite inferior en $\text{registro}_{\text{pag}}(v; \theta)$, cómo se puede considerar que la inferencia maximiza L con respecto a q , y cómo se puede considerar que el aprendizaje maximiza L con respecto a θ . Hemos visto que el algoritmo EM nos permite dar grandes pasos de aprendizaje con un q que los algoritmos de aprendizaje basados en la inferencia MAP nos permiten aprender utilizando una estimación puntual de $\text{registro}_{\text{pag}}(h / v)$ en lugar de inferir toda la distribución. Ahora desarrollamos el enfoque más general del aprendizaje variacional.

La idea central detrás del aprendizaje variacional es que podemos maximizar L sobre una familia restringida de distribuciones q . Esta familia debe elegirse de modo que sea fácil de calcular $\text{registro}_{\text{pag}}(h, v)$. Una forma típica de hacer esto es introducir suposiciones acerca de cómo q factoriza.

Un enfoque común para el aprendizaje variacional es imponer la restricción de que q es una distribución factorial:

$$q(h / v) = \prod_i q(h_i / v). \quad (19.17)$$

Esto se llama el **campo medio** acercarse. Más generalmente, podemos imponer cualquier estructura de modelo gráfico que elijamos en q , para determinar de manera flexible cuántas interacciones queremos que capture nuestra aproximación. Este enfoque de modelo gráfico completamente general se llama **inferencia variacional estructurada** (Saúl y Jordán, 1996).

La belleza del enfoque variacional es que no necesitamos especificar una forma paramétrica específica para q . Específicamente, podemos decir cómo debe factorizarse, pero luego el problema de optimización determina la distribución de probabilidad óptima dentro de esas restricciones de factorización. Para variables latentes discretas, esto solo significa que usamos técnicas de optimización tradicionales para optimizar un número finito de variables que describen el q distribución. Para las variables latentes continuas, esto significa que usamos una rama de las matemáticas llamada cálculo de variaciones para realizar la optimización sobre un espacio de funciones y, de hecho, determinar qué función se debe usar para representar q . El cálculo de variaciones es el origen de los nombres "aprendizaje variacional" e "inferencia variacional", aunque estos nombres se aplican incluso cuando las variables latentes son discretas y no se necesita el cálculo de variaciones. En el caso de variables latentes continuas, el cálculo de variaciones es una técnica poderosa que elimina gran parte de la responsabilidad del diseñador humano del modelo, quien ahora debe especificar solo cómo q factoriza, en lugar de tener que adivinar cómo diseñar un q que puede aproximar con precisión la parte posterior.

Porque $L(v, \theta, q)$ se define como $\text{registro}_{\text{pag}}(v; \theta) - D_{KL}(q(h / v) \| \text{registro}_{\text{pag}}(h / v; \theta))$, podemos pensar en maximizar L con respecto a q como minimizar $D_{KL}(q(h / v) \| \text{registro}_{\text{pag}}(h / v))$.

En este sentido, nos ajustamos q a p . Sin embargo, lo estamos haciendo con la dirección opuesta de la divergencia KL a la que estamos acostumbrados a usar para ajustar una aproximación. Cuando usamos el aprendizaje de máxima probabilidad para ajustar un modelo a los datos, minimizamos $D_{KL}(p \text{datos} \| p \text{modelo})$. Como se ilustra en la figura 3.6, esto significa que la máxima verosimilitud fomenta que el modelo tenga una alta probabilidad en todos los lugares donde los datos tienen una alta probabilidad, mientras que nuestro procedimiento de inferencia basado en la optimización fomenta tener baja probabilidad en todas partes el verdadero posterior tiene baja probabilidad. Ambas direcciones de la divergencia KL pueden tener propiedades deseables e indeseables. La elección de cuál usar depende de qué propiedades son las de mayor prioridad para cada aplicación. En el caso del problema de optimización de inferencia, optamos por utilizar $D_{KL}(q(h / \nu) \| p(h / \nu))$ por razones computacionales. En concreto, la computación $D_{KL}(q(h / \nu) \| p(h / \nu))$ consiste en evaluar las expectativas con respecto a q , por lo que al diseñar q para ser simple, podemos simplificar las expectativas requeridas. La dirección opuesta de la divergencia KL requeriría calcular las expectativas con respecto al verdadero posterior. Debido a que la forma del verdadero posterior está determinada por la elección del modelo, no podemos diseñar un enfoque de cálculo de costo reducido $D_{KL}(p(h / \nu) \| q(h / \nu))$ exactamente.

19.4.1 Variables latentes discretas

La inferencia variacional con variables latentes discretas es relativamente sencilla. Definimos una distribución q , típicamente uno donde cada factor de q simplemente se define mediante una tabla de búsqueda sobre estados discretos. En el caso más simple, h es binario y hacemos la suposición de campo medio de que q factoriza sobre cada individuo h_i . En este caso podemos parametrizar q con un vector \hat{h} cuyas entradas son probabilidades. Entonces $q(h = 1 / \nu) = \hat{h}_i$.

Después de determinar cómo representar q , simplemente optimizamos sus parámetros. En el caso de variables latentes discretas, este es solo un problema de optimización estándar. En principio, la selección de q podría hacerse con cualquier algoritmo de optimización, como el descenso de gradiente.

Debido a que esta optimización debe ocurrir en el ciclo interno de un algoritmo de aprendizaje, debe ser muy rápida. Para lograr esta velocidad, normalmente usamos algoritmos de optimización especiales que están diseñados para resolver problemas comparativamente pequeños y simples en muy pocas iteraciones. Una opción popular es iterar ecuaciones de punto fijo, en otras palabras, resolver

$$\frac{\partial}{\partial \hat{h}_i} L = 0 \quad (19.18)$$

para \hat{h}_i . Actualizamos repetidamente diferentes elementos de \hat{h} hasta que satisfagamos una convergencia

criterio.

Para hacer esto más concreto, mostramos cómo aplicar la inferencia variacional a la **codificación binaria escasa** (presentamos aquí el modelo desarrollado por Henniges *et al.* (2010) pero demuestran el campo medio genérico tradicional aplicado al modelo, mientras que introducen un algoritmo especializado). Esta derivación entra en detalles matemáticos considerables y está destinada al lector que desea resolver por completo cualquier ambigüedad en la descripción conceptual de alto nivel de la inferencia variacional y el aprendizaje que hemos presentado hasta ahora. Los lectores que no planeen derivar o implementar algoritmos de aprendizaje variacional pueden pasar con seguridad a la siguiente sección sin perder ningún concepto nuevo de alto nivel. Se recomienda a los lectores que continúen con el ejemplo de codificación binaria escasa que revisen la lista de propiedades útiles de las funciones que comúnmente surgen en los modelos probabilísticos en la sección 3.10. Usamos estas propiedades generosamente a lo largo de las siguientes derivaciones sin resaltar exactamente dónde usamos cada una.

En el modelo de codificación binaria escasa, la entrada $v \in \mathbb{R}^n$ se genera a partir del modelo añadiendo ruido gaussiano a la suma de m diferentes componentes que pueden estar presentes o ausentes. Cada componente es encendido o apagado por la unidad oculta correspondiente en $h \in \{0,1\}^m$:

$$pag(h=1) = \sigma(b_i) \quad (19.19)$$

$$pag(v / h) = norte(v; \beta Qué? - 1) \quad (19.20)$$

dónde β es un conjunto de sesgos aprendibles, W es una matriz de peso aprendible, y β es una matriz de precisión diagonal aprendible.

Entrenar este modelo con máxima verosimilitud requiere tomar la derivada con respecto a los parámetros. Considere la derivada con respecto a uno de los sesgos:

$$\frac{\partial}{\partial b_i} \underset{\text{registro}}{pag(v)} \quad (19.21)$$

$$= \frac{\partial}{\partial b_i} pag(v) \quad (19.22)$$

$$= \frac{\partial}{\partial b_i} \frac{h pag(h, v)}{pag(v)} \quad (19.23)$$

$$= \frac{\partial}{\partial b_i} \frac{h pag(h) pag(v / h)}{pag(v)} \quad (19.24)$$

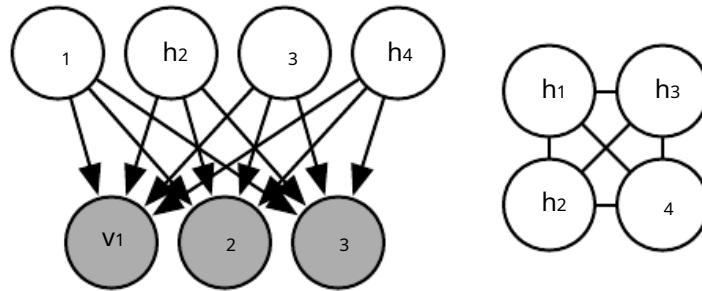


Figura 19.2: La estructura gráfica de un modelo de codificación binaria dispersa con cuatro unidades ocultas. (Izquierda) La estructura gráfica de $\text{pag}(h, v)$. Tenga en cuenta que los bordes están dirigidos y que cada dos unidades ocultas son co-padres de cada unidad visible. (Derecha) La estructura gráfica de $\text{pag}(h / v)$. Para tener en cuenta las rutas activas entre co-padres, la distribución posterior necesita un borde entre todas las unidades ocultas.

$$= \frac{\text{hpag}(v / h) \partial \text{pag}(h)}{\text{pag}(v)} \quad (19.25)$$

$$= \frac{\partial \text{pag}(h)}{h} \frac{\text{pag}(h / v)}{\text{pag}(h)} \quad (19.26)$$

$$= \mathbb{E}_{h \sim \text{pag}(h/v)} \frac{\partial}{\partial h} \text{registro}_{\text{pag}(h)}. \quad (19.27)$$

Esto requiere calcular las expectativas con respecto $\text{apag}(h / v)$. Desafortunadamente, $\text{pag}(h / v)$ es una distribución complicada. Ver figura 19.2 para la estructura gráfica de $\text{pag}(h, v)$ y $\text{pag}(h / v)$. La distribución posterior corresponde al gráfico completo sobre las unidades ocultas, por lo que los algoritmos de eliminación de variables no nos ayudan a calcular las expectativas requeridas más rápido que la fuerza bruta.

Podemos resolver esta dificultad utilizando en su lugar la inferencia variacional y el aprendizaje variacional.

Podemos hacer una aproximación de campo medio:

$$q(h / v) = \prod_i q(h_i / v). \quad (19.28)$$

Las variables latentes del modelo de codificación binaria dispersa son binarias, por lo que para representar un factorial q simplemente necesitamos modelar *distribuciones de Bernoulli* $q(h_i / v)$. Una forma natural de representar las medias de las distribuciones de Bernoulli es con un vector \hat{h} de probabilidades, con $q(h_i = 1 / v) = \hat{h}_i$. Imponemos una restricción que \hat{h} nunca es igual a 0 ni a 1, para evitar errores al calcular, por ejemplo, *registro* \hat{h}_i .

Veremos que las ecuaciones de inferencia variacional nunca asignan 0 o 1 a \hat{h}_i

analíticamente. Sin embargo, en una implementación de software, el error de redondeo de la máquina podría resultar en 001 valores. En el software, es posible que deseemos implementar una codificación dispersa binaria utilizando un vector sin restricciones de parámetros variacionales y obtener \hat{h} a través de la relación $\hat{h} = \sigma(z)$. Por lo tanto, podemos calcular con seguridad registro \hat{h} en una computadora usando la identidad registro $\sigma(z) = -\zeta(-z)$ relacionando el sigmoide y el softplus.

Para comenzar nuestra derivación del aprendizaje variacional en el modelo de codificación binaria dispersa, mostramos que el uso de esta aproximación de campo medio hace que el aprendizaje sea manejable.

El límite inferior de evidencia está dado por

$$L(v, \theta, q) \quad (19.29)$$

$$= E_{h \sim q}[\text{registro} p_{\text{ag}}(h, v)] + H(q) \quad (19.30)$$

$$= E_{h \sim q}[\text{registro} p_{\text{ag}}(h) + \text{registro} p_{\text{ag}}(v / h) - \text{registro} q(h / v)] \quad (19.31)$$

$$= E_{h \sim q} \left[\sum_{i=1}^{\text{metro}} \text{registro} p_{\text{ag}}(h_i) + \sum_{i=1}^{\text{norte}} \text{registro} p_{\text{ag}}(v_i / h) - \sum_{i=1}^{\text{metro}} \text{registro} q(h_i / v) \right] \quad (19.32)$$

$$= \sum_{i=1}^{\text{metro}} \hat{h}(\text{registro} \sigma(b_i) - \text{registro} \hat{h}_i) + (1 - \hat{h}_i)(\text{registro} \sigma(-b_i) - \text{registro} (1 - \hat{h}_i)) \quad (19.33)$$

$$+ \sum_{i=1}^{\text{norte}} \text{registro} \frac{\beta_i}{2\pi} \text{Exp} \left[-\frac{\beta_i(v_i - W_{i:h})^2}{2} \right] \quad (19.34)$$

$$= \sum_{i=1}^{\text{metro}} \hat{h}(\text{registro} \sigma(b_i) - \text{registro} \hat{h}_i) + (1 - \hat{h}_i)(\text{registro} \sigma(-b_i) - \text{registro} (1 - \hat{h}_i)) \quad (19.35)$$

$$+ \frac{1}{2} \sum_{i=1}^{\text{norte}} \text{registro} \frac{\beta_i - \beta}{2} \frac{v_i^2 - 2v_i W_{i:h} + W_{y_o, j}^2}{2\pi} - W_{y_o, j} \hat{h}_i + \sum_{k=j}^{\text{---}} W_{y_o, k} \hat{h}_j \hat{h}_k \quad (19.36)$$

Si bien estas ecuaciones son poco atractivas estéticamente, muestran que L puede expresarse en un pequeño número de operaciones aritméticas simples. El límite inferior de la evidencia L por lo tanto, es tratable. Nosotros podemos usar L como un reemplazo para el log-verosimilitud intratable.

En principio, podríamos simplemente ejecutar el ascenso de gradiente en ambos v y h esto haría un algoritmo combinado de inferencia y entrenamiento perfectamente aceptable. Por lo general, sin embargo, no hacemos esto, por dos razones. En primer lugar, esto requeriría almacenar \hat{h} para cada v . Por lo general, preferimos algoritmos que no requieren memoria por ejemplo. Es difícil escalar los algoritmos de aprendizaje a miles de millones de ejemplos si debemos recordar un vector actualizado dinámicamente asociado con cada ejemplo.

En segundo lugar, nos gustaría poder extraer las características \hat{h} muy rápidamente, para reconocer el contenido de v . En una configuración implementada realista, necesitaríamos poder calcular \hat{h} en tiempo real.

Por estas dos razones, normalmente no usamos el descenso de gradiente para calcular los parámetros medios del campo \hat{h} . En cambio, los estimamos rápidamente con ecuaciones de punto fijo.

La idea detrás de las ecuaciones de punto fijo es que estamos buscando un máximo local con respecto a \hat{h} , dónde $\nabla_h L(v, \theta, \hat{h}) = 0$. No podemos resolver eficientemente esta ecuación con respecto a todos \hat{h} simultáneamente. Sin embargo, podemos resolver para una sola variable:

$$\frac{\partial}{\partial \hat{h}_i} L(v, \theta, \hat{h}) = 0. \quad (19.37)$$

Entonces podemos aplicar iterativamente la solución a la ecuación para $i=1, \dots, n$, y repetir el ciclo hasta que satisfagamos un criterio de convergencia. Los criterios comunes de convergencia incluyen detenerse cuando un ciclo completo de actualizaciones no mejora L por más de cierta cantidad de tolerancia, o cuando el ciclo no cambia \hat{h} por más de una cierta cantidad.

La iteración de ecuaciones de punto fijo de campo medio es una técnica general que puede proporcionar una inferencia variacional rápida en una amplia variedad de modelos. Para hacer esto más concreto, mostramos cómo derivar las actualizaciones para el modelo de codificación binaria escasa en particular.

Primero, debemos escribir una expresión para las derivadas con respecto a \hat{h}_i . Para ello, sustituimos la ecuación 19.36 en el lado izquierdo de la ecuación 19.37:

$$\frac{\partial}{\partial \hat{h}_i} L(v, \theta, \hat{h}) \quad (19.38)$$

$$= \frac{\partial}{\partial \hat{h}_i} - \sum_{j=1}^{n_{\text{metro}}} \hat{h}_j (\text{registro } \sigma(b_j) - \text{registro } \hat{h}_j) + (1 - \hat{h}_j) (\text{registro } \sigma(-b_j) - \text{registro } (1 - \hat{h}_j)) \quad (19.39)$$

$$+ \frac{1}{2} \sum_{j=1}^{n_{\text{norte}}} -\text{registro } \frac{\beta}{2\pi} v_2 - \frac{\beta}{2\pi} v W_{j:h} - W_{2jk} \hat{h}_{k+} - W_{jk} \hat{h}_{k-} \quad (19.40)$$

$$= \text{registro } \sigma(b_i) - \text{registro } \hat{h}_i - 1 + \text{registro } (1 - \hat{h}_i) + 1 - \text{registro } \sigma(-b_i) \quad (19.41)$$

$$+ \sum_{j=1}^{n_{\text{norte}}} -\beta_j v_j W_{ji} - \frac{1}{2} W_{ji} - \sum_{k=i}^n W_{jk} W_{ji} h_k \quad (19.42)$$

$$= b_i - \text{registro } \hat{h}_i + \text{registro } (1 - \hat{h}_i) + v \cdot \beta W_{:,i} - \frac{1}{2} \sum_{j \neq i} W_{:,j} \beta W_{:,i} \hat{h}_j. \quad (19.43)$$

Para aplicar la regla de inferencia de actualización de punto fijo, resolvemos para el \hat{h} que establece ecuación 19.43a a 0:

$$\hat{h} = \sigma(b_i + v \cdot \beta W_{:,i} - \frac{1}{2} \sum_{j \neq i} W_{:,j} \beta W_{:,i} \hat{h}_j). \quad (19.44)$$

En este punto, podemos ver que existe una estrecha conexión entre las redes neuronales recurrentes y la inferencia en modelos gráficos. Específicamente, las ecuaciones de punto fijo de campo medio definieron una red neuronal recurrente. La tarea de esta red es realizar inferencias. Hemos descrito cómo derivar esta red a partir de la descripción de un modelo, pero también es posible entrenar la red de inferencia directamente. Varias ideas basadas en este tema se describen en el capítulo 20.

En el caso de la codificación dispersa binaria, podemos ver que la conexión de red recurrente especificada por la ecuación 19.44 consiste en actualizar repetidamente las unidades ocultas en función de los valores cambiantes de las unidades ocultas vecinas. La entrada siempre envía un mensaje fijo de $v \cdot \beta W$ a las unidades ocultas, pero las unidades ocultas actualizan constantemente el mensaje que se envían entre sí. En concreto, dos unidades \hat{h}_i y \hat{h}_j se inhiben entre sí cuando sus vectores de peso están alineados. Esta es una forma de competencia: entre dos unidades ocultas que explican la entrada, solo la que mejor explica la entrada podrá permanecer activa. Esta competencia es el intento de la aproximación del campo medio de capturar las interacciones explicativas en la codificación binaria dispersa posterior. El efecto de explicación en realidad debería causar un posterior multimodal, de modo que si extraemos muestras del posterior, algunas muestras tendrán una unidad activa, otras muestras tendrán la otra unidad activa, pero muy pocas muestras tendrán ambas activas. Desafortunadamente, la explicación de las interacciones no puede ser modelada por el factorial. Q se usa para el campo medio, por lo que la aproximación del campo medio se ve obligada a elegir un modo para modelar. Este es un ejemplo del comportamiento ilustrado en la figura 3.6.

Podemos reescribir la ecuación 19.44 en una forma equivalente que revela algo más perspectivas:

$$\hat{h} = \sigma(b_i - v \cdot \sum_{j \neq i} W_{:,j} \hat{h}_j - \frac{1}{2} \sum_{j \neq i} W_{:,j} \beta W_{:,i} \hat{h}_j). \quad (19.45)$$

En esta reformulación, vemos que la entrada en cada paso consiste en $v \cdot \sum_{j \neq i} W_{:,j} \hat{h}_j$ en vez de v . Por lo tanto, podemos pensar en la unidad i como un intento de codificar el residuo

error enviado el código de las otras unidades. Por lo tanto, podemos pensar en la codificación dispersa como un codificador automático iterativo, que codifica y decodifica repetidamente su entrada, intentando corregir errores en la reconstrucción después de cada iteración.

En este ejemplo, hemos derivado una regla de actualización que actualiza una sola unidad a la vez. Sería ventajoso poder actualizar más unidades simultáneamente. Algunos modelos gráficos, como las máquinas profundas de Boltzmann, están estructurados de tal manera que podemos resolver muchas entradas de forma simultánea. Desafortunadamente, la codificación binaria escasa no admite tales actualizaciones de bloques. En cambio, podemos usar una técnica heurística llamada **mojadura** para realizar actualizaciones de bloques. En el enfoque de amortiguamiento, resolvemos los valores individualmente óptimos de cada elemento de forma, luego mueva todos los valores en un pequeño paso en esa dirección. Ya no se garantiza que este enfoque aumente la energía en cada paso, pero funciona bien en la práctica para muchos modelos. Ver [Koller y Friedman \(2009\)](#) para obtener más información sobre cómo elegir el grado de sincronía y las estrategias de amortiguación en los algoritmos de paso de mensajes.

19.4.2 Cálculo de Variaciones

Antes de continuar con nuestra presentación del aprendizaje variacional, debemos presentar brevemente un conjunto importante de herramientas matemáticas utilizadas en el aprendizaje variacional: **cálculo de variaciones**.

Muchas técnicas de aprendizaje automático se basan en minimizar una función $J(\theta)$ encontrando el vector de entrada $\theta \in \mathbb{R}^n$ por lo que toma su valor mínimo. Esto se puede lograr con cálculo multivariado y álgebra lineal, resolviendo los puntos críticos donde $\nabla J(\theta) = 0$. En algunos casos, en realidad queremos resolver para una función $J(X)$, como cuando queremos encontrar la función de densidad de probabilidad sobre alguna variable aleatoria. Esto es lo que nos permite hacer el cálculo de variaciones.

Una función de una función J es conocido como **unfuncional** $[J]$. Por mucho que podamos tomar derivadas parciales de una función con respecto a los elementos de su argumento de valor vectorial, podemos tomar **derivados funcionales**, también conocido como **derivados variacionales**, de un funcional $[J]$ con respecto a los valores individuales de la función $J(X)$ en cualquier valor específico de X . La derivada funcional de la funcional J con respecto al valor de la función J en el punto X se denota $\frac{d}{dJ(X)}$.

Un desarrollo formal completo de las derivadas funcionales está más allá del alcance de este libro. Para nuestros propósitos, es suficiente afirmar que para funciones diferenciables $J(X)$ y funciones diferenciables $g_{\text{romo}}(y, X)$ con derivadas continuas, que

$$\frac{d}{dJ(X)} g_{\text{romo}}(J(X), X) dX = \frac{\partial}{\partial y} g_{\text{romo}}(J(X), X). \quad (19.46)$$

Para ganar algo de intuición sobre esta identidad, uno puede pensar en $F(X)$ como un vector con innumerables elementos, indexados por un vector real X . En esta (visión algo incompleta), la identidad que proporciona las derivadas funcionales es la misma que obtendríamos para un vector $\theta \in \mathbb{R}^n$ indexado por enteros positivos:

$$\frac{\partial}{\partial \theta_i} \left[\text{gramo}(\theta_j, j) \right] = \frac{\partial}{\partial \theta_i} \text{gramo}(\theta, y_0). \quad (19.47)$$

Muchos resultados en otras publicaciones de aprendizaje automático se presentan utilizando el método más general **Ecuación de Euler-Lagrange** que permite gramo depender de las derivadas de F así como el valor de F , pero no necesitamos esta forma completamente general para los resultados presentados en este libro.

Para optimizar una función con respecto a un vector, tomamos el gradiente de la función con respecto al vector y resolvemos para el punto donde cada elemento del gradiente es igual a cero. Del mismo modo, podemos optimizar un funcional resolviendo la función donde la derivada funcional en cada punto es igual a cero.

Como ejemplo de cómo funciona este proceso, considere el problema de encontrar la función de distribución de probabilidad sobre $X \in \mathbb{R}$ que tiene máxima entropía diferencial. Recuerde que la entropía de una distribución de probabilidad $pag(X)$ se define como

$$H[pag] = -\int p(X) \ln p(X) dX. \quad (19.48)$$

Para valores continuos, la expectativa es una integral:

$$H[pag] = - \int p(X) \ln p(X) dX. \quad (19.49)$$

No podemos simplemente maximizar $H[pag]$ con respecto a la función $p(X)$, porque el resultado podría no ser una distribución de probabilidad. En cambio, necesitamos usar multiplicadores de Lagrange para agregar una restricción que $p(X)$ se integra a 1. Además, la entropía aumenta sin límite a medida que aumenta la varianza. Esto hace que la cuestión de qué distribución tiene la mayor entropía no sea interesante. En cambio, preguntamos qué distribución tiene la entropía máxima para la varianza fija σ^2 . Finalmente, el problema está subdeterminado porque la distribución puede cambiarse arbitrariamente sin cambiar la entropía. Para imponer una solución única, agregamos una restricción de que la media de la distribución sea μ . El funcional de Lagrange para este problema de optimización es

$$L[pag] = \lambda_1 \int p(X) dX - 1 + \lambda_2 (\text{MI}[X] - \mu) + \lambda_3 \text{MI}[(X - \mu)^2] - \sigma^2 + H[pag] \quad (19.50)$$

$$= \lambda_1 pag(X) + \lambda_2 pag(X)X + \lambda_3 pag(X)(x - \mu)^2 - pag(X) \text{registro} pag(X) dx - \lambda_1 - \mu\lambda_2 - \sigma^2\lambda_3. \quad (19.51)$$

Para minimizar el Lagrangiano con respecto a pag , igualamos las derivadas funcionales a 0:

$$\frac{d}{dp(X)} L = \lambda + \lambda_2 X + \lambda(x - \mu)^2 - \text{registro} \quad pag(X) = 0. \quad (19.52)$$

Esta condición ahora nos dice la forma funcional de $pag(X)$. Reordenando algebraicamente la ecuación, obtenemos

$$pag(X) = \exp. \lambda_1 + \lambda_2 X + \lambda_3 (x - \mu)^2 - 1. \quad (19.53)$$

Nunca asumimos directamente que $pag(X)$ tomaría esta forma funcional; obtuvimos la expresión en sí minimizando analíticamente un funcional. Para terminar el problema de minimización, debemos elegir los valores para asegurar que todas nuestras restricciones sean satisfechas. Somos libres de elegir cualquier valor, porque el gradiente del Lagrangiano con respecto a las variables es cero siempre que las restricciones estén satisfechas. Para satisfacer todas las restricciones, podemos establecer $\lambda_1 = 1 - \text{registro} \sigma^2 \pi$, $\lambda_2 = 0$, y $\lambda_3 = -1$ para obtener

$$pag(X) = norte(X; \mu, \sigma). \quad (19.54)$$

Esta es una razón para usar la distribución normal cuando no conocemos la verdadera distribución. Debido a que la distribución normal tiene la entropía máxima, imponemos la menor cantidad posible de estructura al hacer esta suposición.

Al examinar los puntos críticos de la funcional de Lagrange para la entropía, encontramos solo un punto crítico, correspondiente a la maximización de la entropía para la varianza fija. ¿Qué pasa con la función de distribución de probabilidad que minimiza la entropía? ¿Por qué no encontramos un segundo punto crítico correspondiente al mínimo? La razón es que no existe una función específica que logre una entropía mínima. Como las funciones colocan más densidad de probabilidad en los dos puntos $X = \mu + \sigma$ y $X = \mu - \sigma$, y colocar menos densidad de probabilidad en todos los demás valores de X , pierden entropía manteniendo la varianza deseada. Sin embargo, cualquier función que coloque masa exactamente cero en todos los puntos excepto en dos no se integra a uno y no es una distribución de probabilidad válida. Por lo tanto, no existe una única función de distribución de probabilidad de entropía mínima, al igual que no existe un único número real positivo mínimo. En cambio, podemos decir que hay una secuencia de distribuciones de probabilidad que convergen para poner masa solo en estos dos puntos. Este escenario degenerado puede ser

descrito como una mezcla de distribuciones de Dirac. Debido a que las distribuciones de Dirac no están descritas por una sola función de distribución de probabilidad, ninguna distribución de Dirac o mezcla de Dirac corresponde a un solo punto específico en el espacio de funciones. Estas distribuciones son, por lo tanto, invisibles para nuestro método de resolución de un punto específico donde las derivadas funcionales son cero. Esta es una limitación del método. Las distribuciones como la de Dirac se deben encontrar por otros métodos, como adivinar la solución y luego probar que es correcta.

19.4.3 Variables latentes continuas

Cuando nuestro modelo gráfico contiene variables latentes continuas, aún podemos realizar inferencia y aprendizaje variacional al maximizar L . Sin embargo, ahora debemos usar el cálculo de variaciones al maximizar L con respecto a $q(h / v)$.

En la mayoría de los casos, los profesionales no necesitan resolver ningún problema de cálculo de variaciones por sí mismos. En su lugar, hay una ecuación general para las actualizaciones de punto fijo de campo medio. Si hacemos la aproximación del campo medio

$$q(h / v) = \underset{i}{\text{---}} q(h_i / v), \quad (19.55)$$

y arreglar $q(h_j / v)$ para todos $j = i$, entonces el óptimo $q(h_i / v)$ puede obtenerse normalizando la distribución no normalizada

$$\tilde{q}(h_i / v) = \exp. \text{mi} h_{-yo} \sim q(h_{-yo} / v) \text{registro} pag(v, h) \quad (19.56)$$

siempre y cuando pag no asigne 0 probabilidad a cualquier configuración conjunta de variables. Llevando a cabo la expectativa dentro de la ecuación se obtendrá la forma funcional correcta de $q(h_i / v)$. Sólo es necesario derivar formas funcionales de q usando directamente el cálculo de variaciones si se desea desarrollar una nueva forma de aprendizaje variacional; ecuación 19.56 produce la aproximación del campo medio para cualquier modelo probabilístico.

Ecuación 19.56 es una ecuación de punto fijo, diseñada para ser aplicada iterativamente para cada valor de i repetidamente hasta la convergencia. Sin embargo, también nos dice más que eso. Nos dice la forma funcional que tomará la solución óptima, ya sea que lleguemos allí mediante ecuaciones de punto fijo o no. Esto significa que podemos tomar la forma funcional de esa ecuación pero considerar algunos de los valores que aparecen en ella como parámetros, que podemos optimizar con cualquier algoritmo de optimización que queramos.

Como ejemplo, considere un modelo probabilístico muy simple, con variables latentes $h \in \mathbb{R}_2$ y solo una variable visible, v . Suponer que $pag(h) = norte(h; 0, I)$ y $pag(v / h) = norte(v; w-h; 1)$. De hecho, podríamos simplificar este modelo integrando h ; el resultado es solo una distribución gaussiana sobre v . El modelo en sí no es

interesante; lo hemos construido solo para proporcionar una demostración simple de cómo se puede aplicar el cálculo de variaciones al modelado probabilístico.

El verdadero posterior viene dado, hasta una constante de normalización, por

$$pag(h / v) \quad (19.57)$$

$$\propto pag(h, v) \quad (19.58)$$

$$= pag(h_1)pag(h_2)pag(v / h) \quad (19.59)$$

$$\propto \text{Exp} \left[-\frac{1}{2} (h_1^2 + h_2^2 + (v - h_1 w_1 - h_2 w_2)^2) \right] \quad (19.60)$$

$$= \exp \left[-\frac{1}{2} (h_1^2 + h_2^2 + v^2 + h_1^2 w_1^2 + h_2^2 w_2^2 - 2vh_1 w_1 - 2vh_2 w_2 + 2h_1 w_1 h_2 w_2) \right]. \quad (19.61)$$

Debido a la presencia de los términos que se multiplican h_1 y h_2 juntos, podemos ver que el verdadero posterior no factoriza sobre h_1 y h_2 .

Aplicando ecuación 19.56, encontramos eso

$$\tilde{q}(h_1 / v) \quad (19.62)$$

$$= \exp \left[-\frac{1}{2} \min_{h_2} q(h_2 / v) \right] \quad (19.63)$$

$$= \exp \left[-\frac{1}{2} \min_{h_2} q(h_2 / v) \right] \left[h_2^2 + v^2 + h_2^2 w_1^2 + h_2^2 w_2^2 \right] \quad (19.64)$$

$$- 2vh_1 w_1 - 2vh_2 w_2 + 2h_1 w_1 h_2 w_2 \right]. \quad (19.65)$$

A partir de esto, podemos ver que efectivamente solo hay dos valores que necesitamos obtener de $q(h_2 / v)$: $\min_{h_2} q(h_2 / v)$ y $\max_{h_2} q(h_2 / v)$. Escribir estos como $-h_2 - y - h_2$, obtenemos

$$\tilde{q}(h_1 / v) = \exp \left[-\frac{1}{2} (h_1^2 + h_2^2 + v^2 + h_2^2 w_1^2 + h_2^2 w_2^2) \right] \quad (19.66)$$

$$- 2vh_1 w_1 - 2vh_2 w_2 + 2h_1 w_1 h_2 w_2 \right]. \quad (19.67)$$

A partir de esto, podemos ver que \tilde{q} tiene la forma funcional de una Gaussiana. Así podemos concluir $q(h / v) = \text{norte}(h; \mu, \beta^{-1})$ donde μ y β son parámetros variacionales que podemos optimizar usando cualquier técnica que elijamos. Es importante recordar que nunca asumimos que q sería gaussiano; su forma gaussiana se derivó automáticamente usando cálculo de variaciones para maximizar q con

respecto a L . Usar el mismo enfoque en un modelo diferente podría producir una forma funcional diferente de q .

Esto fue, por supuesto, solo un pequeño caso construido con fines de demostración. Para ver ejemplos de aplicaciones reales del aprendizaje variacional con variables continuas en el contexto del aprendizaje profundo, consulte [Buen compañero et al.](#)(2013d).

19.4.4 Interacciones entre aprendizaje e inferencia

El uso de la inferencia aproximada como parte de un algoritmo de aprendizaje afecta el proceso de aprendizaje y esto, a su vez, afecta la precisión del algoritmo de inferencia.

Específicamente, el algoritmo de entrenamiento tiende a adaptar el modelo de una manera que hace que las suposiciones de aproximación que subyacen al algoritmo de inferencia aproximada se vuelvan más verdaderas. Al entrenar los parámetros, aumenta el aprendizaje variacional

$$\text{mi}_{h \sim q} \text{registro}_{\text{pag}}(v, h). \quad (19.68)$$

para un específico v , esto aumenta $\text{pag}(h \mid v)$ para valores de h que tienen alta probabilidad bajo $q(h \mid v)$ y disminuye $\text{pag}(h \mid v)$ para valores de h que tienen baja probabilidad bajo $q(h \mid v)$.

Este comportamiento hace que nuestras suposiciones aproximadas se conviertan en profecías autocumplidas. Si entrenamos el modelo con un posterior aproximado unimodal, obtendremos un modelo con un posterior verdadero mucho más cercano al unimodal de lo que hubiéramos obtenido entrenando el modelo con inferencia exacta.

Por lo tanto, es muy difícil calcular la verdadera cantidad de daño impuesto a un modelo mediante una aproximación variacional. Existen varios métodos para estimar $\text{registro}_{\text{pag}}(v)$. A menudo estimamos $\text{registro}_{\text{pag}}(v; \theta)$ después de entrenar el modelo, y encuentre que la brecha con $L(v, \theta, q)$ es pequeño. De esto, podemos concluir que nuestra aproximación variacional es precisa para el valor específico de θ que obtuvimos del proceso de aprendizaje. No debemos concluir que nuestra aproximación variacional es precisa en general o que la aproximación variacional hizo poco daño al proceso de aprendizaje. Para medir la verdadera cantidad de daño inducido por la aproximación variacional, necesitaríamos saber $\theta^* = \text{máximo } \theta \text{ } \text{registro}_{\text{pag}}(v; \theta)$. Es posible para $L(v, \theta, q) \approx \text{registro}_{\text{pag}}(v; \theta) + \text{registro}_{\text{pag}}(v; \theta) - \text{registro}_{\text{pag}}(v; \theta^*)$ para sostener simultáneamente. Si $\text{máximo}_q L(v, \theta, q) > \text{registro}_{\text{pag}}(v; \theta^*)$, porque θ^* induce una distribución posterior demasiado complicada para nuestro q familia para capturar, entonces el proceso de aprendizaje nunca se acercará θ^* . Tal problema es muy difícil de detectar, porque solo podemos saber con certeza que sucedió si tenemos un algoritmo de aprendizaje superior que puede encontrar θ^* para comparacion.

19.5 Inferencia aproximada aprendida

Hemos visto que la inferencia se puede considerar como un procedimiento de optimización que aumenta el valor de una función L . La optimización explícita mediante procedimientos iterativos, como ecuaciones de punto fijo o la optimización basada en gradientes, suele ser muy costosa y requiere mucho tiempo. Muchos enfoques de inferencia evitan este gasto aprendiendo a realizar inferencias aproximadas. Específicamente, podemos pensar en el proceso de optimización como una función \hat{F} que asigna una entrada v a una distribución aproximada $q = \text{argmax}_q L(v, q)$. Una vez que pensamos en el proceso de optimización iterativa de varios pasos como una simple función, podemos aproximarla con una red neuronal que implementa una aproximación $\hat{F}(v, \theta)$.

19.5.1 Vigilia-Sueño

Una de las principales dificultades con el entrenamiento de un modelo para inferir h de v es que no disponemos de un conjunto de entrenamiento supervisado con el que entrenar al modelo. Dado un v , no sabemos el apropiado h . El mapeo de v a h depende de la elección de la familia modelo, y evoluciona a lo largo del proceso de aprendizaje como θ cambia. El algoritmo de despertar-dormir ([Hinton et al., 1995b](#); [frey et al., 1996](#)) resuelve este problema extrayendo muestras de ambos h y v de la distribución del modelo. Por ejemplo, en un modelo dirigido, esto se puede hacer de forma económica realizando un muestreo ancestral a partir de h terminando en v . Luego, la red de inferencia se puede entrenar para realizar el mapeo inverso: predecir qué h causó el presente v . El principal inconveniente de este enfoque es que solo podremos entrenar la red de inferencia en valores de v que tienen alta probabilidad bajo el modelo. Al principio del aprendizaje, la distribución del modelo no se parecerá a la distribución de los datos, por lo que la red de inferencia no tendrá la oportunidad de aprender sobre muestras que se parezcan a los datos.

En la sección [18.2](#) vimos que una posible explicación del papel del sueño en los seres humanos y los animales es que los sueños podrían proporcionar las muestras de fase negativa que utilizan los algoritmos de entrenamiento de Monte Carlo para aproximar el gradiente negativo de la función de partición logarítmica de los modelos no dirigidos. Otra posible explicación para los sueños biológicos es que proporciona muestras de $p_{\text{ag}}(h, v)$ que se puede usar para entrenar una red de inferencia para predecir h dado v . En algunos sentidos, esta explicación es más satisfactoria que la explicación de la función de partición. Los algoritmos de Monte Carlo generalmente no funcionan bien si se ejecutan utilizando solo la fase positiva del gradiente durante varios pasos y luego solo la fase negativa del gradiente durante varios pasos. Los seres humanos y los animales suelen estar despiertos durante varias horas consecutivas y luego dormidos durante varias horas consecutivas. Es

No es evidente cómo este programa podría apoyar el entrenamiento de Monte Carlo de un modelo no dirigido. Algoritmos de aprendizaje basados en maximizar \mathcal{L} se puede ejecutar con períodos prolongados de mejora q y períodos prolongados de mejora θ , sin embargo. Si el papel de los sueños biológicos es entrenar redes para predecir q , entonces esto explica cómo los animales pueden permanecer despiertos durante varias horas (cuanto más tiempo están despiertos, mayor es la brecha entre \mathcal{L} y $\text{registro pag}(\mathcal{V})$, pero \mathcal{L} seguirá siendo un límite inferior) y permanecer dormido durante varias horas (el modelo generativo en sí no se modifica durante el sueño) sin dañar sus modelos internos. Por supuesto, estas ideas son puramente especulativas, y no hay pruebas contundentes que sugieran que soñar logra cualquiera de estos objetivos. Soñar también puede servir para el aprendizaje por refuerzo en lugar del modelado probabilístico, al tomar muestras de experiencias sintéticas del modelo de transición del animal, sobre las cuales entrenar la política del animal. O el sueño puede tener algún otro propósito aún no anticipado por la comunidad de aprendizaje automático.

19.5.2 Otras formas de inferencia aprendida

Esta estrategia de inferencia aproximada aprendida también se ha aplicado a otros modelos. Salakhutdinov y Larochelle(2010) mostró que un solo paso en una red de inferencia aprendida podría generar una inferencia más rápida que iterar las ecuaciones de punto fijo de campo medio en un DBM. El procedimiento de entrenamiento se basa en ejecutar la red de inferencia, luego aplicar un paso de campo medio para mejorar sus estimaciones y entrenar la red de inferencia para generar esta estimación refinada en lugar de su estimación original.

Ya hemos visto en la sección 14.8 que el modelo predictivo de descomposición dispersa entrena una red codificadora poco profunda para predecir un código disperso para la entrada. Esto puede verse como un híbrido entre un codificador automático y una codificación dispersa. Es posible idear una semántica probabilística para el modelo, bajo la cual se puede considerar que el codificador realiza una inferencia MAP aproximada aprendida. Debido a su codificador poco profundo, PSD no puede implementar el tipo de competencia entre unidades que hemos visto en la inferencia de campo medio. Sin embargo, ese problema se puede remediar entrenando un codificador profundo para realizar una inferencia aproximada aprendida, como en la técnica ISTA (Gregor y Le Cun,2010b).

La inferencia aproximada aprendida se ha convertido recientemente en uno de los enfoques dominantes para el modelado generativo, en la forma del codificador automático variacional (Reyma,2013;Rezende et al.,2014). En este enfoque elegante, no hay necesidad de construir objetivos explícitos para la red de inferencia. En cambio, la red de inferencia simplemente se usa para definir \mathcal{L} , y luego los parámetros de la red de inferencia se adaptan para aumentar \mathcal{L} . Este modelo se describe en profundidad más adelante, en la sección 20.10.3.

Mediante la inferencia aproximada, es posible entrenar y utilizar una amplia variedad de modelos. Muchos de estos modelos se describen en el próximo capítulo.

capítulo 20

Modelos generativos profundos

En este capítulo, presentamos varios de los tipos específicos de modelos generativos que se pueden construir y entrenar usando las técnicas presentadas en los capítulos.[dieciséis-19](#). Todos estos modelos representan distribuciones de probabilidad sobre múltiples variables de alguna manera. Algunos permiten evaluar explícitamente la función de distribución de probabilidad. Otros no permiten la evaluación de la función de distribución de probabilidad, pero admiten operaciones que implícitamente requieren conocerla, como la extracción de muestras de la distribución. Algunos de estos modelos son modelos probabilísticos estructurados descritos en términos de gráficos y factores, utilizando el lenguaje de los modelos gráficos presentado en el capítulo[dieciséis](#). Otros no pueden describirse fácilmente en términos de factores, pero no obstante representan distribuciones de probabilidad.

20.1 Máquinas de Boltzmann

Las máquinas de Boltzmann se introdujeron originalmente como un enfoque "conexionista" general para aprender distribuciones de probabilidad arbitrarias sobre vectores binarios ([Fahlman et al., 1983; ackley et al., 1985; Hinton et al., 1984; Hinton y Sejnowski, 1986](#)). Las variantes de la máquina de Boltzmann que incluyen otros tipos de variables hace tiempo que superaron la popularidad de la original. En esta sección, presentamos brevemente la máquina binaria de Boltzmann y analizamos los problemas que surgen al intentar entrenar y realizar inferencias en el modelo.

Definimos la máquina de Boltzmann sobre una d -vector aleatorio binario dimensional $\mathbf{X} \in \{0,1\}^d$. La máquina de Boltzmann es un modelo basado en energía (sección[16.2.4](#)),

lo que significa que definimos la distribución de probabilidad conjunta usando una función de energía:

$$PAG(X) = \frac{\text{Exp}(-E(X))}{Z}, \quad (20.1)$$

dónde $m(X)$ es la función de energía y Z es la función de partición que asegura que $\sum PAG(X) = 1$. La función de energía de la máquina de Boltzmann está dada por

$$m(X) = -X \cdot Ux - b \cdot X, \quad (20.2)$$

dónde U es la matriz de "peso" de los parámetros del modelo y b es el vector de parámetros de sesgo.

En el escenario general de la máquina de Boltzmann, se nos da un conjunto de ejemplos de entrenamiento, cada uno de los cuales son n -dimensional. Ecuación 20.1 describe la distribución de probabilidad conjunta sobre las variables observadas. Si bien este escenario es ciertamente viable, limita los tipos de interacciones entre las variables observadas a las descritas por la matriz de ponderación. Específicamente, significa que la probabilidad de que una unidad esté encendida viene dada por un modelo lineal (regresión logística) a partir de los valores de las otras unidades.

La máquina de Boltzmann se vuelve más poderosa cuando no se observan todas las variables. En este caso, las variables latentes pueden actuar de manera similar a las unidades ocultas en un perceptrón multicapa y modelar interacciones de orden superior entre las unidades visibles. Así como la adición de unidades ocultas para convertir la regresión logística en un MLP da como resultado que el MLP sea un aproximador universal de funciones, una máquina de Boltzmann con unidades ocultas ya no se limita a modelar relaciones lineales entre variables. En cambio, la máquina de Boltzmann se convierte en un aproximador universal de funciones de masa de probabilidad sobre variables discretas (Le Roux y Bengio, 2008).

Formalmente, descomponemos las unidades X en dos subconjuntos: las unidades visibles v y las unidades latentes (u o ocultas) h . La función de energía se convierte en

$$m(v, h) = -v \cdot Rv - v \cdot \zeta Qh - h \cdot Sh - b \cdot v - c \cdot H. \quad (20.3)$$

Aprendizaje automático de Boltzmann Los algoritmos de aprendizaje para las máquinas de Boltzmann generalmente se basan en la máxima verosimilitud. Todas las máquinas de Boltzmann tienen una función de partición intratable, por lo que el gradiente de máxima verosimilitud debe aproximarse utilizando las técnicas descritas en el capítulo 18.

Una propiedad interesante de las máquinas de Boltzmann cuando se entrena con reglas de aprendizaje basadas en la máxima verosimilitud es que la actualización para un peso particular que conecta dos unidades depende solo de las estadísticas de esas dos unidades, recopiladas bajo diferentes distribuciones: $PAG_{\text{modelo}}(v)$ y $PAG_{\text{datos}}(v)PAG_{\text{modelo}}(h / v)$. El resto de

La red participa en la configuración de esas estadísticas, pero el peso se puede actualizar sin saber nada sobre el resto de la red o cómo se produjeron esas estadísticas. Esto significa que la regla de aprendizaje es "local", lo que hace que el aprendizaje automático de Boltzmann sea biológicamente plausible. Es concebible que si cada neurona fuera una variable aleatoria en una máquina de Boltzmann, entonces los axones y las dendritas que conectan dos variables aleatorias podrían aprender solo observando el patrón de disparo de las células que realmente tocan físicamente. En particular, en la fase positiva, dos unidades que frecuentemente se activan juntas ven fortalecida su conexión. Este es un ejemplo de una regla de aprendizaje de Hebbian ([Hebb,1949](#)) a menudo resumido con el mnemotécnico "disparar juntos, conectar juntos". Las reglas de aprendizaje de Hebb se encuentran entre las explicaciones hipotéticas más antiguas para el aprendizaje en sistemas biológicos y siguen siendo relevantes en la actualidad ([Giudice et al.,2009](#)).

Otros algoritmos de aprendizaje que usan más información que las estadísticas locales parecen requerir que hipoteticemos la existencia de más maquinaria que esta. Por ejemplo, para que el cerebro implemente la retropropagación en un perceptrón multicapa, parece necesario que el cerebro mantenga una red de comunicación secundaria para transmitir información de gradiente hacia atrás a través de la red. Se han hecho propuestas para implementaciones (y aproximaciones) biológicamente plausibles de la propagación hacia atrás ([Hinton,2007a;bengio,2015](#)) pero quedan por validar, y [bengio\(2015\)](#) vincula la retropropagación de gradientes con la inferencia en modelos basados en energía similares a la máquina de Boltzmann (pero con variables latentes continuas).

La fase negativa del aprendizaje automático de Boltzmann es algo más difícil de explicar desde un punto de vista biológico. Como se argumenta en la sección [18.2](#), el sueño onírico puede ser una forma de muestreo de fase negativa. Sin embargo, esta idea es más especulativa.

20.2 Máquinas de Boltzmann restringidas

Inventado bajo el nombre [armonio](#)([Smolensky,1986](#)), las máquinas restringidas de Boltzmann son algunos de los componentes básicos más comunes de los modelos probabilísticos profundos. Hemos descrito brevemente los RBM anteriormente, en la sección [16.7.1](#). Aquí repasamos la información anterior y entramos en más detalle. Los RBM son modelos gráficos probabilísticos no dirigidos que contienen una capa de variables observables y una sola capa de variables latentes. Los RBM se pueden apilar (uno encima del otro) para formar modelos más profundos. Ver figura [20.1](#) para algunos ejemplos. En particular figura [20.1a](#) muestra la estructura gráfica del propio RBM. Es un gráfico bipartito, sin conexiones permitidas entre ninguna variable en la capa observada o entre ninguna unidad en la capa latente.

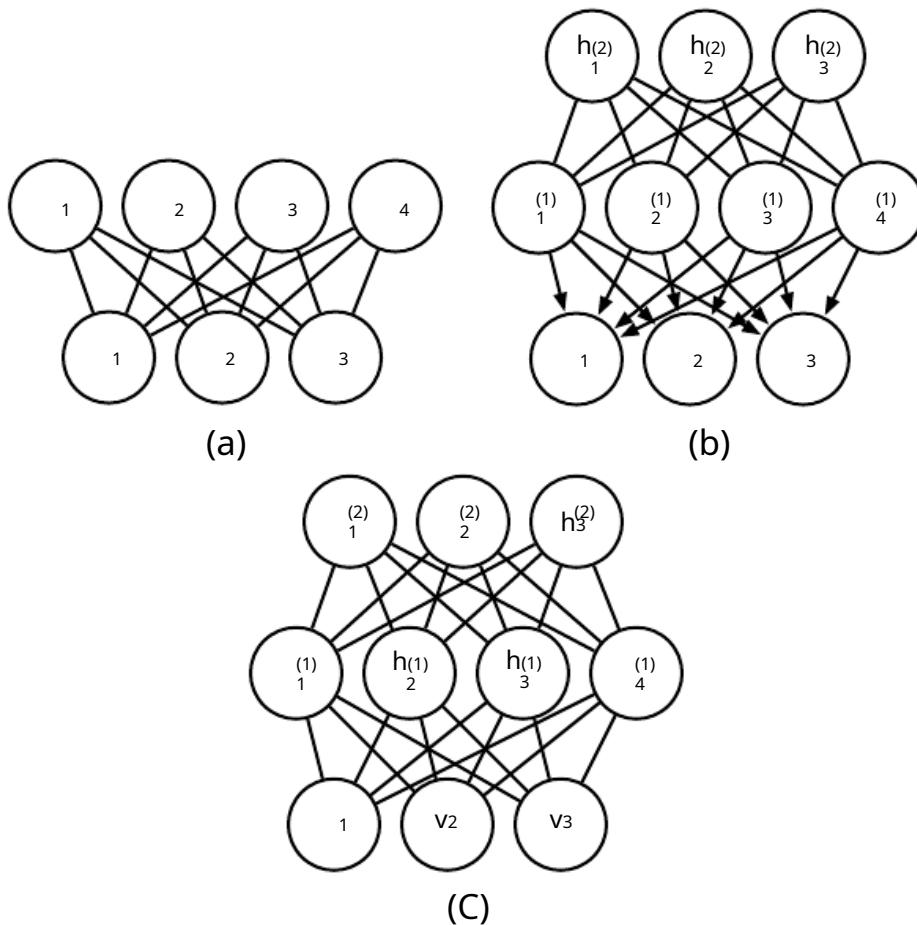


Figura 20.1: Ejemplos de modelos que se pueden construir con máquinas Boltzmann restringidas.

(a) La propia máquina restringida de Boltzmann es un modelo gráfico no dirigido basado en un gráfico bipartito, con unidades visibles en una parte del gráfico y unidades ocultas en la otra parte. No hay conexiones entre las unidades visibles, ni conexiones entre las unidades ocultas. Por lo general, todas las unidades visibles están conectadas a todas las unidades ocultas, pero es posible construir RBM escasamente conectados, como los RBM convolucionales. (b) Una red de creencias profundas es un modelo gráfico híbrido que incluye conexiones dirigidas y no dirigidas. Al igual que un RBM, no tiene conexiones intracapa. Sin embargo, una DBN tiene varias capas ocultas y, por lo tanto, existen conexiones entre unidades ocultas que se encuentran en capas separadas. Todas las distribuciones de probabilidad condicional locales que necesita la red de creencias profundas se copian directamente de las distribuciones de probabilidad condicional locales de sus RBM constituyentes. Alternativamente, también podríamos representar la red de creencias profundas con un gráfico completamente no dirigido, pero necesitaría conexiones intracapa para capturar las dependencias entre los padres. (c) Una máquina profunda de Boltzmann es un modelo gráfico no dirigido con varias capas de variables latentes. Al igual que los RBM y los DBN, los DBM carecen de conexiones intracapa. Los DBM están menos vinculados a los RBM que los DBN. Al inicializar un DBM desde una pila de RBM, es necesario modificar ligeramente los parámetros de RBM. Algunos tipos de DBM se pueden entrenar sin entrenar primero un conjunto de RBM.

Comenzamos con la versión binaria de la máquina de Boltzmann restringida, pero como veremos más adelante existen extensiones a otros tipos de unidades visibles y ocultas.

Más formalmente, dejemos que la capa observada consista en un conjunto de *n* variables aleatorias binarias a las que nos referimos colectivamente con el vector v . Nos referimos a la capa latente u oculta de m variables aleatorias binarias como h .

Al igual que la máquina de Boltzmann general, la máquina de Boltzmann restringida es un modelo basado en energía con la distribución de probabilidad conjunta especificada por su energía función:

$$PAG(v = v, h = h) = \frac{1}{Z} \text{Exp}(-E(v, h)). \quad (20.4)$$

La función de energía para un RBM está dada por

$$m(v, h) = -b \cdot v - c \cdot h - v \cdot h \quad (20.5)$$

y Z es la constante de normalización conocida como función de partición:

$$Z = \sum_{v} \sum_{h} \text{Exp}\{-m(v, h)\}. \quad (20.6)$$

Es evidente a partir de la definición de la función de partición Z que el ingenuo método de computación Z (sumando exhaustivamente sobre todos los estados) podría ser computacionalmente intratable, a menos que un algoritmo ingeniosamente diseñado pudiera explotar las regularidades en la distribución de probabilidad para calcular Z más rápido. En el caso de máquinas Boltzmann restringidas, [Largo y Servedio \(2010\)](#) demostró formalmente que la función de partición Z es intratable. La función de partición intratable Z implica que la distribución de probabilidad conjunta normalizada $PAG(v)$ también es intratable de evaluar.

20.2.1 Distribuciones condicionales

Aunque $PAG(v)$ es intratable, la estructura gráfica bipartita de la RBM tiene la propiedad muy especial de que sus distribuciones condicionales $PAG(h | v)$ y $PAG(v | h)$ son factoriales y relativamente simples de calcular y muestrear.

Deducir las distribuciones condicionales de la distribución conjunta es sencillo:

$$PAG(h | v) = \frac{PAG(h, v)}{PAG(v)} \quad (20.7)$$

$$= \frac{1}{PAG(v)Z} \text{Exp}[-b \cdot v - c \cdot h - v \cdot h] \quad (20.8)$$

$$= \frac{1}{Z} \text{Exp}[-c \cdot h - v \cdot h] \quad (20.9)$$

$$= \frac{1}{Z} \exp \left[\sum_{j=1}^n C_j h_j + \sum_{j=1}^n v \cdot W_{:,j} h_j \right] \quad (20.10)$$

$$= \frac{1}{Z} \exp \left[\sum_{j=1}^n C_j h_j + v \cdot W_{:,j} h_j \right] \quad (20.11)$$

Ya que estamos condicionando en las unidades visibles v , podemos tratarlos como constantes con respecto a la distribución $PAG(h/v)$. La naturaleza factorial del condicional $PAG(h/v)$ se sigue inmediatamente de nuestra capacidad para escribir la probabilidad conjunta sobre el vector h como el producto de distribuciones (no normalizadas) sobre los elementos individuales, h_j . Ahora es una simple cuestión de normalizar las distribuciones sobre el binario individual h_j .

$$PAG(h_j=1/v) = \frac{PAG(h_j=1/v)}{PAG(h_j=0/v) + PAG(h_j=1/v)} \quad (20.12)$$

$$= \frac{\exp \{C_j + v \cdot W_{:,j}\}}{\exp \{0\} + \exp \{C_j + v \cdot W_{:,j}\}} \quad (20.13)$$

$$= \sigma(C_j + v \cdot W_{:,j}) \quad (20.14)$$

Ahora podemos expresar el condicional completo sobre la capa oculta como el factorial distribución:

$$PAG(h/v) = \prod_{j=1}^n \sigma(2h_j - 1) \cdot (C_j + v \cdot W_{:,j}). \quad (20.15)$$

Una derivación similar mostrará que la otra condición que nos interesa, $PAG(v/h)$, es también una distribución factorial:

$$PAG(v/h) = \prod_{i=1}^m \sigma((2v_i - 1) \cdot (b_i + \text{¿Qué?}))_i. \quad (20.16)$$

20.2.2 Máquinas de Boltzmann con restricciones de entrenamiento

Porque la RBM admite la evaluación y diferenciación eficiente de $PAG(v)$ y muestreo MCMC eficiente en forma de muestreo de bloque Gibbs, se puede entrenar fácilmente con cualquiera de las técnicas descritas en el capítulo 18 para entrenar modelos que tienen funciones de partición intratables. Esto incluye CD, SML (PCD), comparación de proporciones, etc. En comparación con otros modelos no dirigidos utilizados en el aprendizaje profundo, el RBM es relativamente sencillo de entrenar porque podemos calcular $PAG(h/v)$

exactamente en forma cerrada. Algunos otros modelos profundos, como la máquina profunda de Boltzmann, combinan la dificultad de una función de partición intratable y la dificultad de una inferencia intratable.

20.3 Redes de creencias profundas

Redes de creencias profundas(DBN) fueron uno de los primeros modelos no convolucionales en admitir con éxito el entrenamiento de arquitecturas profundas ([Hinton et al., 2006](#); [Hinton, 2007b](#)). La introducción de redes de creencias profundas en 2006 inició el renacimiento actual del aprendizaje profundo. Antes de la introducción de las redes de creencias profundas, los modelos profundos se consideraban demasiado difíciles de optimizar. Las máquinas kernel con funciones objetivas convexas dominaron el panorama de la investigación. Las redes de creencias profundas demostraron que las arquitecturas profundas pueden tener éxito al superar a las máquinas de vectores de soporte kernelizadas en el conjunto de datos MNIST ([Hinton et al., 2006](#)). Hoy en día, las redes de creencias profundas en su mayoría han caído en desgracia y rara vez se usan, incluso en comparación con otros algoritmos de aprendizaje generativo o no supervisados, pero aún son merecidamente reconocidas por su importante papel en la historia del aprendizaje profundo.

Las redes de creencias profundas son modelos generativos con varias capas de variables latentes. Las variables latentes suelen ser binarias, mientras que las unidades visibles pueden ser binarias o reales. No hay conexiones intracapa. Por lo general, cada unidad en cada capa está conectada a cada unidad en cada capa vecina, aunque es posible construir DBN más escasamente conectadas. Las conexiones entre las dos capas superiores no están dirigidas. Las conexiones entre todas las demás capas están dirigidas, con las flechas apuntando hacia la capa más cercana a los datos. Ver figura20.1b para un ejemplo.

Una DBN con y_0 capas ocultas contiene y_0 matrices de peso: $W_{(1)}, \dots, W_{(y_0)}$. también contiene $y_0 + 1$ vectores de sesgo: $b_{(0)}, \dots, b_{(y_0)}$, con $b_{(0)}$ proporcionando los sesgos para la capa visible. La distribución de probabilidad representada por la DBN está dada por

$$PAG(h_{(y_0)}, h_{(y_0-1)}) = \text{Exp} \left[b_{(y_0)} + \sum_{i=1}^{h_{(y_0)}} h_{(y_0-1)}^i - \sum_{i=1}^{h_{(y_0-1)}} h_{(y_0-1)}^i - \sum_{j=1}^{h_{(y_0-1)}} W_{(y_0)}^j h_{(y_0)}^j \right], \quad (20.17)$$

$$PAG(h_{(k)}) = 1 / \text{Exp} \left[\sum_{i=1}^{h_{(k)}} b_{(k)}^i + \sum_{j=1}^{h_{(k)}} W_{(k+1)}^j h_{(k+1)}^j \right] \quad \forall i, \forall k \in 1, \dots, l-2, \quad (20.18)$$

$$PAG(v=1/h_{(1)}) = \text{Exp} \left[b_{(0)} + \sum_{i=1}^{h_{(1)}} W_{(1)}^i h_{(1)}^i \right] \quad (20.19)$$

En el caso de unidades visibles de valor real, sustituir

$$v \sim \text{norte} \quad v; b_{(0)} + W_{(1)} \cdot h_{(1)}, \beta^{-1} \quad (20.20)$$

con β -diagonal para la maleabilidad. Las generalizaciones a otras unidades visibles de la familia exponencial son sencillas, al menos en teoría. Una DBN con una sola capa oculta es solo una RBM.

Para generar una muestra a partir de una DBN, primero ejecutamos varios pasos de muestreo de Gibbs en las dos capas superiores ocultas. Esta etapa consiste esencialmente en extraer una muestra del RBM definido por las dos capas ocultas superiores. Luego podemos usar un solo paso de muestreo ancestral a través del resto del modelo para extraer una muestra de las unidades visibles.

Las redes de creencias profundas incurren en muchos de los problemas asociados con los modelos dirigidos y los modelos no dirigidos.

La inferencia en una red de creencias profundas es intratable debido al efecto de explicación dentro de cada capa dirigida y debido a la interacción entre las dos capas ocultas que tienen conexiones no dirigidas. Evaluar o maximizar el límite inferior de la evidencia estándar en el log-verosimilitud también es intratable, porque el límite inferior de la evidencia toma la expectativa de camarillas cuyo tamaño es igual al ancho de la red.

Evaluar o maximizar la probabilidad logarítmica requiere no solo enfrentar el problema de la inferencia intratable para marginar las variables latentes, sino también el problema de una función de partición intratable dentro del modelo no dirigido de las dos capas superiores.

Para entrenar una red de creencias profundas, uno comienza entrenando un RBM para maximizar $m_{\text{iv} \sim p_{\text{datos}}} \log p_{\text{pag}}(v)$ usando divergencia contrastiva o máxima verosimilitud estocástica. Los parámetros de la RBM definen entonces los parámetros de la primera capa de la DBN. A continuación, se entrena un segundo RBM para maximizar aproximadamente

$$m_{\text{iv} \sim p_{\text{datos}}} m_{h_{(1)} \sim p_{\text{ag}}(h_{(1)})} \log p_{\text{pag}}(h_{(1)}) \quad (20.21)$$

dónde $p_{\text{ag}}(h_{(1)})$ es la distribución de probabilidad representada por el primer RBM y $p_{\text{ag}}(h_{(2)})$ es la distribución de probabilidad representada por el segundo RBM. En otras palabras, el segundo RBM se entrena para modelar la distribución definida mediante el muestreo de las unidades ocultas del primer RBM, cuando el primer RBM está controlado por los datos. Este procedimiento se puede repetir indefinidamente, para agregar tantas capas a la DBN como se desee, con cada nueva RBM modelando las muestras de la anterior. Cada RBM define otra capa de la DBN. Este procedimiento puede justificarse como el aumento de un límite inferior variacional en la verosimilitud logarítmica de los datos bajo la DBN ([Hinton et al., 2006](#)).

En la mayoría de las aplicaciones, no se hace ningún esfuerzo por entrenar conjuntamente la DBN una vez que se completa el procedimiento codicioso por capas. Sin embargo, es posible realizar un ajuste fino generativo utilizando el algoritmo de despertar-dormir.

La DBN entrenada puede usarse directamente como un modelo generativo, pero la mayor parte del interés en las DBN surgió de su capacidad para mejorar los modelos de clasificación. Podemos tomar los pesos del DBN y usarlos para definir un MLP:

$$h_{(1)} = \sigma b_{(1)} + v \cdot W_{(1)}. \quad (20.22)$$

$$h_{(yo)} = \sigma b_{(yo)} + h_{(yo-1)} \cdot W_{(yo)} \quad \forall yo \in 2, \dots, metro, \quad (20.23)$$

Después de inicializar este MLP con los pesos y sesgos aprendidos a través del entrenamiento generativo de la DBN, podemos entrenar el MLP para realizar una tarea de clasificación. Esta capacitación adicional del MLP es un ejemplo de ajuste fino discriminativo.

Esta elección específica de MLP es algo arbitraria, en comparación con muchas de las ecuaciones de inferencia del capítulo 19 que se derivan de los primeros principios. Este MLP es una opción heurística que parece funcionar bien en la práctica y se usa consistentemente en la literatura. Muchas técnicas de inferencia aproximadas están motivadas por su capacidad para encontrar un máximo *ajustado* límite inferior variacional en el log-verosimilitud bajo algún conjunto de restricciones. Se puede construir un límite inferior variacional en el log-verosimilitud utilizando las expectativas de unidades ocultas definidas por el MLP de DBN, pero esto es cierto para *cualquier* distribución de probabilidad sobre las unidades ocultas, y no hay razón para creer que este MLP proporcione un límite particularmente estrecho. En particular, el MLP ignora muchas interacciones importantes en el modelo gráfico DBN. El MLP propaga información hacia arriba desde las unidades visibles hasta las unidades ocultas más profundas, pero no propaga ninguna información hacia abajo o hacia los lados. El modelo gráfico DBN explica las interacciones entre todas las unidades ocultas dentro de la misma capa, así como las interacciones de arriba hacia abajo entre las capas.

Si bien la probabilidad logarítmica de una DBN es intratable, se puede aproximar con AIS ([Salakhutdinov y Murray, 2008](#)). Esto permite evaluar su calidad como modelo generativo.

El término "red de creencias profundas" se usa comúnmente de manera incorrecta para referirse a cualquier tipo de red neuronal profunda, incluso redes sin semántica variable latente. El término "red de creencias profundas" debe referirse específicamente a modelos con conexiones no dirigidas en la capa más profunda y conexiones dirigidas que apuntan hacia abajo entre todos los demás pares de capas consecutivas.

El término "red de creencias profundas" también puede causar cierta confusión porque el término "red de creencias" a veces se usa para referirse a modelos puramente dirigidos, mientras que las redes de creencias profundas contienen una capa no dirigida. Las redes de creencias profundas también comparten el acrónimo DBN con las redes bayesianas dinámicas ([Decano y Kanazawa, 1989](#)), que son redes bayesianas para representar cadenas de Markov.

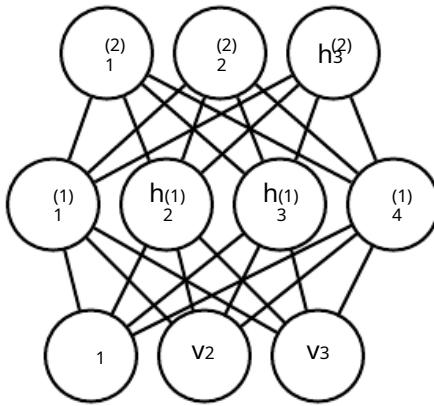


Figura 20.2: El modelo gráfico para una máquina profunda de Boltzmann con una capa visible (abajo) y dos capas ocultas. Las conexiones son solo entre unidades en capas vecinas. No hay conexiones de capa intracapa.

20.4 Máquinas profundas de Boltzmann

Amáquina profunda de Boltzmann DBM ([Salakhutdinov y Hinton, 2009a](#)) es otro tipo de modelo profundo y generativo. A diferencia de la red de creencias profundas (DBN), es un modelo totalmente no dirigido. A diferencia del RBM, el DBM tiene varias capas de variables latentes (los RBM tienen solo una). Pero al igual que la RBM, dentro de cada capa, cada una de las variables son mutuamente independientes, condicionadas por las variables de las capas vecinas. Ver figura [20.2](#) para la estructura del gráfico. Las máquinas Deep Boltzmann se han aplicado a una variedad de tareas, incluido el modelado de documentos ([Srivastava et al., 2013](#)).

Al igual que los RBM y los DBN, los DBM generalmente contienen solo unidades binarias, como suponemos para simplificar nuestra presentación del modelo, pero es sencillo incluir unidades visibles de valor real.

Un DBM es un modelo basado en energía, lo que significa que la distribución de probabilidad conjunta sobre las variables del modelo está parametrizada por una función de energía m_i . En el caso de una máquina Boltzmann profunda con una capa visible, v , y tres capas ocultas, $h(1)$, $h(2)$ y $h(3)$, la probabilidad conjunta viene dada por:

$$PAGv, h(1), h(2), h(3) = \frac{1}{Z(\theta)} \exp^{-m_i(v, h(1), h(2), h(3); \theta)}. \quad (20.24)$$

Para simplificar nuestra presentación, omitimos los parámetros de sesgo a continuación. La función de energía DBM se define entonces de la siguiente manera:

$$m_i(v, h(1), h(2), h(3); \theta) = -v \cdot W(1)h(1) - h(1) \cdot W(2)h(2) - h(2) \cdot W(3)h(3). \quad (20.25)$$

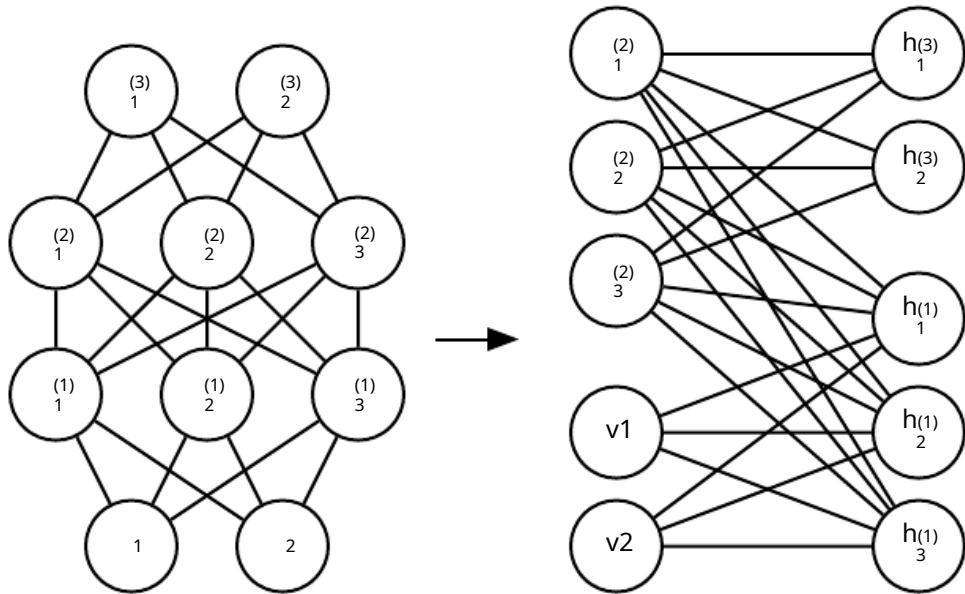


Figura 20.3: Una máquina profunda de Boltzmann, reorganizada para revelar su estructura gráfica bipartita.

En comparación con la función de energía RBM (ecuación20.5), la función de energía DBM incluye conexiones entre las unidades ocultas (variables latentes) en forma de matrices de peso ($W_{(2)}$ y $W_{(3)}$). Como veremos, estas conexiones tienen consecuencias significativas tanto para el comportamiento del modelo como para la forma en que realizamos la inferencia en el modelo.

En comparación con las máquinas Boltzmann totalmente conectadas (con cada unidad conectada a cada otra unidad), el DBM ofrece algunas ventajas similares a las que ofrece el RBM. En concreto, como se ilustra en la figura20.3, las capas de DBM se pueden organizar en un gráfico bipartito, con capas impares en un lado y capas pares en el otro. Esto implica inmediatamente que cuando condicionamos las variables en la capa par, las variables en las capas impares se vuelven condicionalmente independientes. Por supuesto, cuando condicionamos las variables en las capas impares, las variables en las capas pares también se vuelven condicionalmente independientes.

La estructura bipartita del DBM significa que podemos aplicar las mismas ecuaciones que hemos usado previamente para las distribuciones condicionales de un RBM para determinar las distribuciones condicionales en un DBM. Las unidades dentro de una capa son condicionalmente independientes entre sí dados los valores de las capas vecinas, por lo que las distribuciones sobre variables binarias pueden describirse completamente mediante los parámetros de Bernoulli que dan la probabilidad de que cada unidad esté activa. En nuestro ejemplo con dos capas ocultas, las probabilidades de activación vienen dadas por:

$$PAG(v=1 / h_{(1)}) = \sigma W_{(1)} h_{(1)}, \quad (20.26)$$

$$PAG(h_i | v, h_{(2)}) = \sigma(v \cdot W_{(1),i} + W_{(2),i}^T h_{(2)}) \quad (20.27)$$

y

$$PAG(h_{(2)} | v, h_{(1)}) = \sigma(h_{(1)} \cdot W_{(2)}) \quad (20.28)$$

La estructura bipartita hace que el muestreo de Gibbs en una máquina profunda de Boltzmann sea eficiente. El enfoque ingenuo del muestreo de Gibbs es actualizar solo una variable a la vez. Los RBM permiten que todas las unidades visibles se actualicen en un bloque y todas las unidades ocultas se actualicen en un segundo bloque. Uno podría suponer ingenuamente que un DBM con y capas requiere $y+1$ actualizaciones, con cada iteración actualizando un bloque que consta de una capa de unidades. En cambio, es posible actualizar todas las unidades en solo dos iteraciones. El muestreo de Gibbs se puede dividir en dos bloques de actualizaciones, uno que incluye todas las capas pares (incluida la capa visible) y el otro que incluye todas las capas impares. Debido al patrón de conexión DBM bipartito, dadas las capas pares, la distribución sobre las capas impares es factorial y, por lo tanto, se puede muestrear de forma simultánea e independiente como un bloque. Asimismo, dadas las capas impares, las capas pares se pueden muestrear de forma simultánea e independiente como un bloque. El muestreo eficiente es especialmente importante para el entrenamiento con el algoritmo de máxima verosimilitud estocástica.

20.4.1 Propiedades interesantes

Las máquinas Deep Boltzmann tienen muchas propiedades interesantes.

Los DBM se desarrollaron después de los DBN. En comparación con las DBN, la distribución posterior $PAG(h | v)$ es más simple para los DBM. De manera un tanto contraria a la intuición, la simplicidad de esta distribución posterior permite aproximaciones más ricas de la posterior. En el caso de la DBN, realizamos la clasificación utilizando un procedimiento de inferencia aproximada motivado heurísticamente, en el que suponemos que un valor razonable para la expectativa de campo medio de las unidades ocultas puede proporcionarse mediante un paso ascendente a través de la red en un MLP que utiliza funciones de activación sigmoide y los mismos pesos que el DBN original. *Cualquier* distribución $q(h)$ se puede utilizar para obtener un límite inferior variacional en la verosimilitud logarítmica. Por lo tanto, este procedimiento heurístico nos permite obtener dicho límite. Sin embargo, el límite no está optimizado explícitamente de ninguna manera, por lo que el límite puede estar lejos de ser ajustado. En particular, la estimación heurística deignora las interacciones entre unidades ocultas dentro de la misma capa, así como la influencia de retroalimentación de arriba hacia abajo de unidades ocultas en capas más profundas en unidades ocultas que están más cerca de la entrada. Debido a que el procedimiento de inferencia basado en MLP heurístico en la DBN no puede dar cuenta de estas interacciones, el resultado es presumiblemente lejos

desde lo óptimo. En DBM, todas las unidades ocultas dentro de una capa son condicionalmente independientes dadas las otras capas. Esta falta de interacción intracapa hace posible el uso de ecuaciones de punto fijo para optimizar realmente el límite inferior variacional y encontrar las verdaderas expectativas óptimas de campo medio (dentro de cierta tolerancia numérica).

El uso del campo medio adecuado permite que el procedimiento de inferencia aproximado para DBM capture la influencia de las interacciones de retroalimentación de arriba hacia abajo. Esto hace que los DBM sean interesantes desde el punto de vista de la neurociencia, porque se sabe que el cerebro humano usa muchas conexiones de retroalimentación de arriba hacia abajo. Debido a esta propiedad, los DBM se han utilizado como modelos computacionales de fenómenos neurocientíficos reales.[Serie et al., 2010](#); [reichert et al., 2011](#)).

Una propiedad desafortunada de los DBM es que el muestreo de ellos es relativamente difícil. Los DBN solo necesitan usar el muestreo MCMC en su par superior de capas. Las otras capas se utilizan solo al final del proceso de muestreo, en un paso de muestreo ancestral eficiente. Para generar una muestra de un DBM, es necesario usar MCMC en todas las capas, con cada capa del modelo participando en cada transición de la cadena de Markov.

20.4.2 Inferencia de campo medio DBM

La distribución condicional sobre una capa DBM dadas las capas vecinas es factorial. En el ejemplo del DBM con dos capas ocultas, estas distribuciones son $PAG(v / h_1)$, $PAG(h_1 / v, h_2)$ y $PAG(h_2 / h_1)$. La distribución sobre *todo* las capas ocultas generalmente no se factorizan debido a las interacciones entre las capas. En el ejemplo con dos capas ocultas, $PAG(h_1, h_2 / v)$ no factoriza debido a los pesos de interacción $W_{(2)}$ entre h_1 y h_2 que hacen que estas variables sean mutuamente dependientes.

Como fue el caso con la DBN, nos queda buscar métodos para aproximar la distribución posterior de la DBM. Sin embargo, a diferencia de la DBN, la distribución posterior de la DBM sobre sus unidades ocultas, aunque complicada, es fácil de aproximar con una aproximación variacional (como se analiza en la sección [19.4](#)), específicamente una aproximación de campo medio. La aproximación de campo medio es una forma simple de inferencia variacional, donde restringimos la distribución de aproximación a distribuciones totalmente factoriales. En el contexto de los DBM, las ecuaciones de campo medio capturan las interacciones bidireccionales entre capas. En esta sección derivamos el procedimiento de inferencia aproximada iterativa introducido originalmente en [Salakhutdinov y Hinton \(2009a\)](#).

En las aproximaciones variacionales a la inferencia, nos acercamos a la tarea de aproximar

emparejar una distribución objetivo particular, en nuestro caso, la distribución posterior sobre las unidades ocultas dadas las unidades visibles, por alguna familia de distribuciones razonablemente simple. En el caso de la aproximación de campo medio, la familia de aproximación es el conjunto de distribuciones donde las unidades ocultas son condicionalmente independientes.

Ahora desarrollamos el enfoque de campo medio para el ejemplo con dos capas ocultas. Dejar $q(h_{(1)}, h_{(2)} / v)$ ser la aproximación de $PAG(h_{(1)}, h_{(2)} / v)$. La suposición de campo medio implica que

$$q(h_{(1)}, h_{(2)} / v) = \prod_j q(h_{(1)}^j / v) \prod_k q(h_{(2)}^k / v). \quad (20.29)$$

La aproximación del campo medio intenta encontrar un miembro de esta familia de distribuciones que mejor se ajuste a la distribución posterior verdadera $PAG(h_{(1)}, h_{(2)} / v)$. Es importante destacar que el proceso de inferencia debe ejecutarse nuevamente para encontrar una distribución diferente cada vez que usamos un nuevo valor de v .

Uno puede concebir muchas formas de medir qué tan bien $q(h / v)$ fit $PAG(h / v)$. El enfoque de campo medio es minimizar

$$KL(QP) = \sum_h q(h_{(1)}, h_{(2)} / v) \text{ registro} \frac{q(h_{(1)}, h_{(2)} / v)}{PAG(h_{(1)}, h_{(2)} / v)}. \quad (20.30)$$

En general, no tenemos que proporcionar una forma paramétrica de la distribución aproximada más allá de hacer cumplir los supuestos de independencia. El procedimiento de aproximación variacional generalmente puede recuperar una forma funcional de la distribución aproximada. Sin embargo, en el caso de un supuesto de campo medio sobre unidades ocultas binarias (el caso que estamos desarrollando aquí) no hay pérdida de generalidad resultante de fijar una parametrización del modelo de antemano.

Parametrizamos q como producto de las distribuciones de Bernoulli, es decir, asociamos la probabilidad de cada elemento de $h_{(1)}$ con un parámetro. En concreto, para cada j , $h_{(1)}^j = q(h_{(1)}^j | j = 1 / v)$, donde $\hat{h}_{(1)}^j \in [0, 1]$ y para cada k , $h_{(2)}^k = q(h_{(2)}^k | k = 1 / v)$, donde $\hat{h}_{(2)}^k \in [0, 1]$. Así tenemos la siguiente aproximación a la posterior:

$$q(h_{(1)}, h_{(2)} / v) = \prod_j q(h_{(1)}^j / v) \prod_k q(h_{(2)}^k / v) \quad (20.31)$$

$$= \prod_j (\hat{h}_{(1)}^j)^{h_{(1)}^j} (1 - \hat{h}_{(1)}^j)^{1-h_{(1)}^j} \times \prod_k (\hat{h}_k^{(2)})^{h_k^{(2)}} (1 - \hat{h}_k^{(2)})^{1-h_k^{(2)}}. \quad (20.32)$$

Por supuesto, para DBM con más capas, la parametrización posterior aproximada se puede extender de la manera obvia, explotando la estructura bipartita del gráfico.

para actualizar todas las capas pares simultáneamente y luego actualizar todas las capas impares simultáneamente, siguiendo el mismo programa que el muestreo de Gibbs.

Ahora que hemos especificado nuestra familia de distribuciones aproximadas q , queda por concretar un procedimiento para elegir al miembro de esta familia que mejor encaje PAG . La forma más directa de hacer esto es usar las ecuaciones de campo promedio especificadas por la ecuación 19.56. Estas ecuaciones se derivaron resolviendo donde las derivadas del límite inferior variacional son cero. Describen de manera abstracta cómo optimizar el límite inferior variacional para cualquier modelo, simplemente tomando expectativas con respecto a q .

Aplicando estas ecuaciones generales, obtenemos las reglas de actualización (nuevamente, ignorando los términos de sesgo):

$$\hat{h}_j^{(1)} = \sigma \left(\sum_i v_i W_{j|i}^{(1)} + \sum_k W_{jk}^{(2)} \hat{h}_k^{(2)} \right), \quad \forall j \quad (20.33)$$

$$\hat{h}_k^{(2)} = \sigma \left(\sum_j W_{jk}^{(2)} \hat{h}_j^{(1)} \right), \quad \forall k. \quad (20.34)$$

En un punto fijo de este sistema de ecuaciones, tenemos un máximo local del límite inferior variacional $L(q)$. Por lo tanto, estas ecuaciones de actualización de punto fijo definen una algoritmo iterativo donde alternamos actualizaciones de $\hat{h}_j^{(1)}$ (usando la ecuación 20.33) y actualizaciones de $\hat{h}_k^{(2)}$ (usando la ecuación 20.34). En problemas pequeños como MNIST, como pocos ya que diez iteraciones pueden ser suficientes para encontrar un gradiente de fase positivo aproximado para el aprendizaje, y cincuenta suelen ser suficientes para obtener una representación de alta calidad de un solo ejemplo específico que se utilizará para una clasificación de alta precisión. Extender la inferencia variacional aproximada a DBM más profundos es sencillo.

20.4.3 Aprendizaje de parámetros DBM

El aprendizaje en el DBM debe enfrentar tanto el desafío de una función de partición intratable, usando las técnicas del capítulo 18, y el desafío de una distribución posterior intratable, utilizando las técnicas del capítulo 19.

Como se describe en la sección 20.4.2, la inferencia variacional permite la construcción de una distribución $q(h | v)$ que se aproxima a lo intratable $PAG(h | v)$. Entonces, el aprendizaje procede maximizando $L(v, Q, \theta)$, el límite inferior variacional del logaritmo de verosimilitud intratable, registro $PAG(v; \theta)$.

Para una máquina profunda de Boltzmann con dos capas ocultas, L es dado por

$$L(Q, \theta) = \sum_{i,j} v_i W_{y_i, j, h_j}^{(1)} + \sum_{j,k} \hat{h}_{j,k}^{(1)} W_{j, k, h_k}^{(2)} - \text{registroZ}(\theta) + H(q). \quad (20.35)$$

Esta expresión todavía contiene la función de partición de registro, $\text{registroZ}(\theta)$. Debido a que una máquina Boltzmann profunda contiene máquinas Boltzmann restringidas como componentes, los resultados de dureza para calcular la función de partición y el muestreo que se aplican a las máquinas Boltzmann restringidas también se aplican a las máquinas Boltzmann profundas. Esto significa que evaluar la función de masa de probabilidad de una máquina de Boltzmann requiere métodos aproximados como el muestreo de importancia recocido. Asimismo, entrenar el modelo requiere aproximaciones al gradiente de la función de partición logarítmica. Ver capítulo 18 para una descripción general de estos métodos. Los DBM normalmente se entran utilizando la máxima verosimilitud estocástica. Muchas de las otras técnicas descritas en el capítulo 18 no son aplicables. Las técnicas como la pseudoverosimilitud requieren la capacidad de evaluar las probabilidades no normalizadas, en lugar de simplemente obtener un límite inferior variacional sobre ellas. La divergencia contrastiva es lenta para las máquinas de Boltzmann profundas porque no permiten un muestreo eficiente de las unidades ocultas dadas las unidades visibles; en cambio, la divergencia contrastiva requeriría quemarse en una cadena de Markov cada vez que se necesita una nueva muestra de fase negativa.

La versión no variacional del algoritmo de máxima verosimilitud estocástica se discutió anteriormente, en la sección 18.2. La máxima verosimilitud estocástica variacional aplicada al DBM se proporciona en el algoritmo 20.1. Recuerde que describimos una variante simplificada del DBM que carece de parámetros de sesgo; incluirlos es trivial.

20.4.4 Entrenamiento previo por capas

Desafortunadamente, entrenar un DBM utilizando la máxima verosimilitud estocástica (como se describe arriba) a partir de una inicialización aleatoria generalmente resulta en fallas. En algunos casos, el modelo no aprende a representar adecuadamente la distribución. En otros casos, el DBM puede representar bien la distribución, pero sin mayor probabilidad que la que se podría obtener con solo un RBM. Un DBM con pesos muy pequeños en todos menos en la primera capa representa aproximadamente la misma distribución que un RBM.

Se han desarrollado varias técnicas que permiten el entrenamiento conjunto y se describen en la sección 20.4.5. Sin embargo, el método original y más popular para superar el problema del entrenamiento conjunto de los DBM es el preentrenamiento codicioso por capas. En este método, cada capa del DBM se entrena de forma aislada como un RBM. La primera capa está entrenada para modelar los datos de entrada. Cada RBM posterior se entrena para modelar muestras de la distribución posterior del RBM anterior. Después de todo el

Algoritmo 20.1 El algoritmo de máxima verosimilitud estocástica variacional para entrenar un DBM con dos capas ocultas.

Colocar σ , el tamaño del paso, a un pequeño número positivo

Colocar k , el número de pasos de Gibbs, lo suficientemente alto como para permitir una cadena de Markov de $pag(v, h_{(1)}, h_{(2)}; \theta + \Delta\theta)$ para quemar, a partir de muestras de $depag(v, h_{(1)}, h_{(2)}; \theta)$. Inicializar tres matrices, \tilde{V} , $\tilde{H}_{(1)}$ y $\tilde{H}_{(2)}$ cada uno con $metrofilas$ establecidas en valores aleatorios (p. ej., de distribuciones de Bernoulli, posiblemente con marginales coincidentes con los marginales del modelo).

mientras no convergente (bucle de aprendizaje) **hacer**

Muestra un mini lote de $metro$ ejemplos de los datos de entrenamiento y organícelos como las filas de una matriz de diseño V .

Inicializar matrices $\hat{H}_{(1)}$ y $\hat{H}_{(2)}$, posiblemente a los marginales del modelo. **mientras** no convergente (información media del campo) bucle de referencia **hacer**

$$\hat{H}_{(1)} \leftarrow \sigma \cdot V W_{(1)} + \hat{H}_{(2)} W_{(2)}.$$

$$\hat{H}_{(2)} \leftarrow \sigma \cdot \hat{H}_{(1)} W_{(2)}.$$

terminar mientras

$$\Delta W_{(1)} \leftarrow 1 \cdot metro V \cdot \hat{H}_{(1)}$$

$$\Delta W_{(2)} \leftarrow 1 \cdot metro \hat{H}_{(1)} \cdot \hat{H}_{(2)}$$

para $y_o = 1$ a k (muestreo de Gibbs) **hacer**

Bloque de Gibbs 1:

$$\forall y_o, j, \tilde{V}_{y_o, j} \text{ muestreado de } PAG(\tilde{V}_{y_o, j} = 1) = \sigma$$

$$W_{(1)} \tilde{H}_{(1)} \\ j,: \quad i,: \\ \hat{H}_{(1)}^T W_{(2), j}.$$

$$\forall i, j, \tilde{H}_{(2)y_o, j} \text{ muestreado de } PAG(H_{(2)y_o, j} = 1) = \sigma$$

$$\tilde{V}_{i,:} W_{:,j}^{(1)} + H_{i,:} W_{j,:}^{(2)}.$$

Bloque de Gibbs 2:

$$\forall i, j, \tilde{H}_{(1)y_o, j} \text{ muestreado de } PAG(H_{(1)y_o, j} = 1) = \sigma$$

$$\tilde{V}_{i,:} W_{:,j}^{(1)} + H_{i,:} W_{j,:}^{(2)}.$$

fin para

$$\Delta W_{(1)} \leftarrow \Delta \cdot W_{(1)} - 1 \cdot metro V \cdot \tilde{H}_{(1)}^T$$

$$\Delta W_{(2)} \leftarrow \Delta \cdot W_{(2)} - 1 \cdot metro \tilde{H}_{(1)}^T - \tilde{H}_{(2)}^T$$

$W_{(1)} \leftarrow W_{(1)} + \Delta W_{(1)}$ (esta es una ilustración de dibujos animados, en la práctica use un algoritmo más efectivo, como el impulso con una tasa de aprendizaje decreciente)

$$W_{(2)} \leftarrow W_{(2)} + \Delta W_{(2)}$$

terminar mientras

Los RBM se han entrenado de esta manera, se pueden combinar para formar un DBM. A continuación, el DBM puede entrenarse con PCD. Por lo general, el entrenamiento de PCD hará solo un pequeño cambio en los parámetros del modelo y su rendimiento medido por la probabilidad logarítmica que asigna a los datos, o su capacidad para clasificar las entradas. Ver figura 20.4 para ver una ilustración del procedimiento de entrenamiento.

Este procedimiento de entrenamiento codicioso por capas no es solo un ascenso coordinado. Se parece un poco al ascenso coordinado porque optimizamos un subconjunto de los parámetros en cada paso. Los dos métodos difieren porque el procedimiento de entrenamiento codicioso por capas utiliza una función objetivo diferente en cada paso.

El preentrenamiento codicioso por capas de un DBM difiere del entrenamiento previo codicioso por capas de un DBN. Los parámetros de cada RBM individual pueden copiarse directamente a la DBN correspondiente. En el caso de la DBM, los parámetros de la RBM deben modificarse antes de su inclusión en la DBM. Una capa en el medio de la pila de RBM se entrena solo con entrada de abajo hacia arriba, pero después de que la pila se combine para formar el DBM, la capa tendrá tanto entrada de abajo hacia arriba como de arriba hacia abajo. Para dar cuenta de este efecto, [Salakhutdinov y Hinton\(2009a\)](#) abogan por dividir los pesos de todos menos el RBM superior e inferior por la mitad antes de insertarlos en el DBM. Además, el RBM inferior debe entrenarse utilizando dos "copias" de cada unidad visible y los pesos vinculados para que sean iguales entre las dos copias. Esto significa que los pesos se duplican efectivamente durante el paso hacia arriba. De manera similar, el RBM superior debe entrenarse con dos copias de la capa superior.

Obtener resultados de última generación con la máquina profunda de Boltzmann requiere una modificación del algoritmo SML estándar, que consiste en utilizar una pequeña cantidad de campo medio durante la fase negativa del paso de entrenamiento conjunto de PCD ([Salakhutdinov y Hinton,2009a](#)). Específicamente, la expectativa del gradiente de energía debe calcularse con respecto a la distribución media del campo en la que todas las unidades son independientes entre sí. Los parámetros de esta distribución de campo medio deben obtenerse ejecutando las ecuaciones de punto fijo de campo medio en un solo paso. Ver [Buen compañero et al.\(2013b\)](#) para una comparación del rendimiento de DBM centrados con y sin el uso de campo medio parcial en la fase negativa.

20.4.5 Entrenamiento conjunto de máquinas de Boltzmann profundas

Los DBM clásicos requieren un entrenamiento previo codicioso y no supervisado, y para realizar bien la clasificación, requieren un clasificador basado en MLP separado además de las características ocultas que extraen. Esto tiene algunas propiedades indeseables. Es difícil realizar un seguimiento del rendimiento durante el entrenamiento porque no podemos evaluar las propiedades del DBM completo mientras entrenamos el primer RBM. Por lo tanto, es difícil decir qué tan bien nuestros hiperparámetros

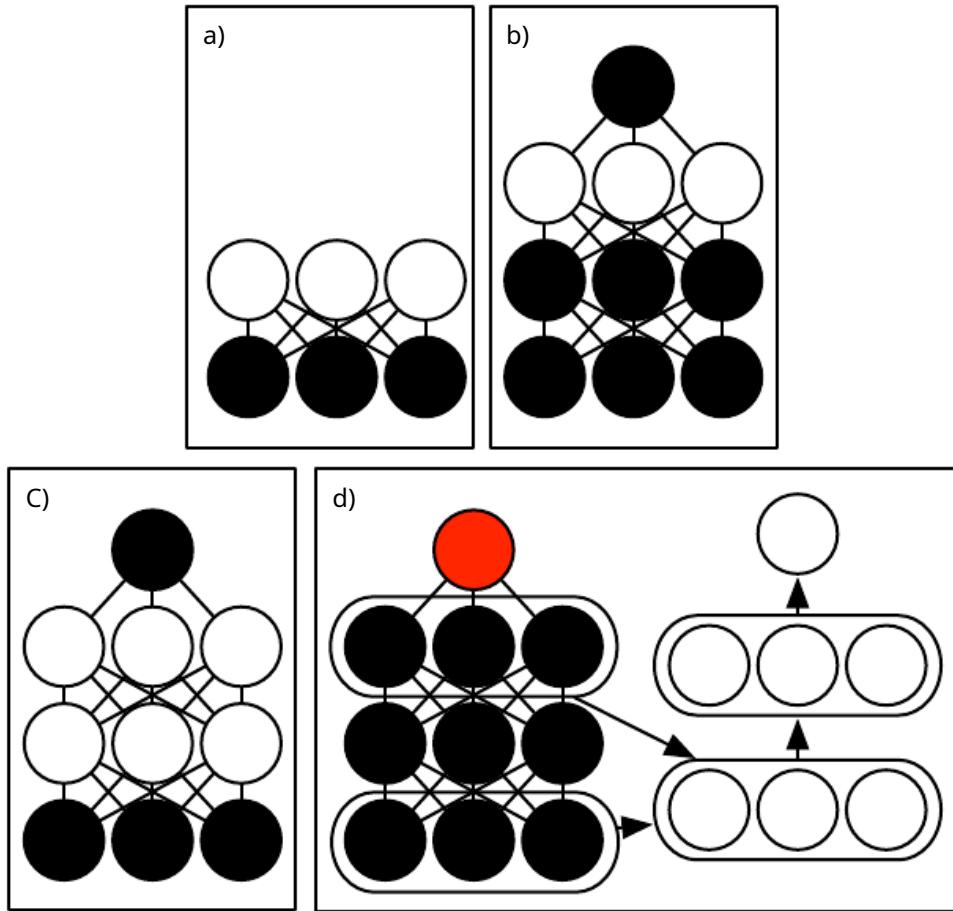


Figura 20.4: El procedimiento de entrenamiento profundo de la máquina Boltzmann utilizado para clasificar el conjunto de datos MNIST (Salakhutdinov y Hinton, 2009a; Srivastava et al., 2014). (a) Entrene un RBM usando CD para maximizar aproximadamente el registro $PAG(v)$. (b) Entrene un segundo RBM que modele $h_{(1)}$ y la clase de destino y usando CD- k para maximizar aproximadamente el registro $PAG(h_{(1)}, y)$ donde $h_{(1)}$ se extrae del primer RBM posterior condicionado a los datos. Aumentar k del 1 al 20 durante el aprendizaje. (c) Combine los dos RBM en un DBM. Entrénalo para maximizar aproximadamente el registro $PAG(v, y)$ utilizando la máxima verosimilitud estocástica con $k=5$. (d) Eliminar v y del modelo. Definir un nuevo conjunto de características $h_{(1)}$ y $h_{(2)}$ que se obtienen ejecutando la inferencia de campo medio en el modelo que carece de y . Utilice estas funciones como entrada para un MLP cuya estructura es la misma que una pasada adicional de campo medio, con una capa de salida adicional para la estimación de y . Inicialice los pesos de MLP para que sean los mismos que los pesos de DBM. Entrenar el MLP para maximizar aproximadamente el registro $PAG(y/v)$ usando descenso de abandono de gradiente estocástico. Figura reimpresa de (Buen compañero et al., 2013b).

están trabajando hasta bastante tarde en el proceso de formación. Las implementaciones de software de DBM deben tener muchos componentes diferentes para el entrenamiento de CD de RBM individuales, el entrenamiento de PCD del DBM completo y el entrenamiento basado en la propagación hacia atrás a través del MLP. Finalmente, el MLP sobre la máquina de Boltzmann pierde muchas de las ventajas del modelo probabilístico de la máquina de Boltzmann, como la capacidad de realizar inferencias cuando faltan algunos valores de entrada.

Hay dos formas principales de resolver el problema del entrenamiento conjunto de la máquina profunda de Boltzmann. El primero es el **máquina Boltzmann profunda centrada** ([Montavon y Müller, 2012](#)), que reparametriza el modelo para hacer que la hessiana de la función de costo esté mejor condicionada al inicio del proceso de aprendizaje. Esto produce un modelo que se puede entrenar sin una etapa de preentrenamiento codicioso por capas. El modelo resultante obtiene una excelente probabilidad logarítmica del conjunto de pruebas y produce muestras de alta calidad. Desafortunadamente, sigue sin poder competir con los MLP debidamente regularizados como clasificador. La segunda forma de entrenar conjuntamente una máquina profunda de Boltzmann es utilizar un **máquina Boltzmann profunda de predicción múltiple** ([Buen compañero et al., 2013b](#)). Este modelo utiliza un criterio de entrenamiento alternativo que permite el uso del algoritmo de retropropagación para evitar los problemas con las estimaciones MCMC del gradiente. Desafortunadamente, el nuevo criterio no conduce a buenas probabilidades o muestras, pero, en comparación con el enfoque MCMC, conduce a un rendimiento de clasificación superior y a la capacidad de razonar bien sobre las entradas faltantes.

El truco de centrado para la máquina de Boltzmann es más fácil de describir si volvemos a la visión general de una máquina de Boltzmann que consta de un conjunto de unidades X con una matriz de pesos t y sesgos b . Recuperar de la ecuación [20.2](#) que la función de energía está dada por

$$mi(X) = -X \cdot Ux - b \cdot X. \quad (20.36)$$

Uso de diferentes patrones de dispersión en la matriz de ponderación t , podemos implementar estructuras de máquinas de Boltzmann, como RBM o DBM con diferentes números de capas. Esto se logra dividiendo X en unidades visibles y ocultas y poniendo a cero los elementos de t para unidades que no interactúan. La máquina de Boltzmann centrada introduce un vector μ que se resta de todos los estados:

$$mi(X; Ub) = -(x - \mu) \cdot tu(x - \mu) - (x - \mu) \cdot b. \quad (20.37)$$

Típicamente μ es un hiperparámetro fijo al comienzo del entrenamiento. Por lo general, se elige para asegurarse de que $x - \mu \approx 0$ cuando se inicializa el modelo. Esta reparametrización no cambia el conjunto de distribuciones de probabilidad que el modelo puede representar, pero sí cambia la dinámica de descenso de gradiente estocástico aplicada a la verosimilitud. En concreto, en muchos casos, esta reparametrización resulta

en una matriz hessiana mejor acondicionada.[melchore et al.](#)(2013) confirmaron experimentalmente que el condicionamiento de la matriz hessiana mejora, y observaron que el truco de centrado es equivalente a otra técnica de aprendizaje automático de Boltzmann, **eldegradado mejorado**([Cho et al.](#),2011). El acondicionamiento mejorado de la matriz Hessian permite que el aprendizaje tenga éxito, incluso en casos difíciles como entrenar una máquina profunda de Boltzmann con múltiples capas.

El otro enfoque para el entrenamiento conjunto de máquinas de Boltzmann profundas es la máquina de Boltzmann profunda de predicción múltiple (MP-DBM), que funciona al ver las ecuaciones de campo medio como si definieran una familia de redes recurrentes para resolver aproximadamente todos los problemas de inferencia posibles ([Buen compañero et al.](#),2013b). En lugar de entrenar el modelo para maximizar la probabilidad, el modelo se entrena para que cada red recurrente obtenga una respuesta precisa al problema de inferencia correspondiente. El proceso de entrenamiento se ilustra en la figura.[20.5](#). Consiste en muestrear aleatoriamente un ejemplo de entrenamiento, muestrear aleatoriamente un subconjunto de entradas a la red de inferencia y luego entrenar la red de inferencia para predecir los valores de las unidades restantes.

Este principio general de retropropagación a través del gráfico computacional para la inferencia aproximada se ha aplicado a otros modelos ([Stoyanov et al.](#),2011; [freno et al.](#),2013). En estos modelos y en el MP-DBM, la pérdida final no es el límite inferior de la probabilidad. En cambio, la pérdida final generalmente se basa en la distribución condicional aproximada que la red de inferencia aproximada impone sobre los valores faltantes. Esto significa que el entrenamiento de estos modelos está algo motivado heurísticamente. Si inspeccionamos el $\text{pag}(\nu)$ representado por la máquina de Boltzmann aprendida por el MP-DBM, tiende a ser algo defectuoso, en el sentido de que el muestreo de Gibbs produce muestras deficientes.

La retropropagación a través del gráfico de inferencia tiene dos ventajas principales. Primero, entrena el modelo tal como se usa realmente, con inferencia aproximada. Esto significa que la inferencia aproximada, por ejemplo, para completar las entradas que faltan o para realizar una clasificación a pesar de la presencia de entradas que faltan, es más precisa en el MP-DBM que en el DBM original. El DBM original no hace un clasificador preciso por sí mismo; los mejores resultados de clasificación con el DBM original se basaron en el entrenamiento de un clasificador independiente para usar características extraídas por el DBM, en lugar de usar la inferencia en el DBM para calcular la distribución sobre las etiquetas de clase. La inferencia de campo medio en MP-DBM funciona bien como clasificador sin modificaciones especiales. La otra ventaja de la retropropagación a través de la inferencia aproximada es que la retropropagación calcula el gradiente exacto de la pérdida. Esto es mejor para la optimización que los gradientes aproximados del entrenamiento SML, que sufren tanto de sesgo como de varianza. Esto probablemente explica por qué MP-

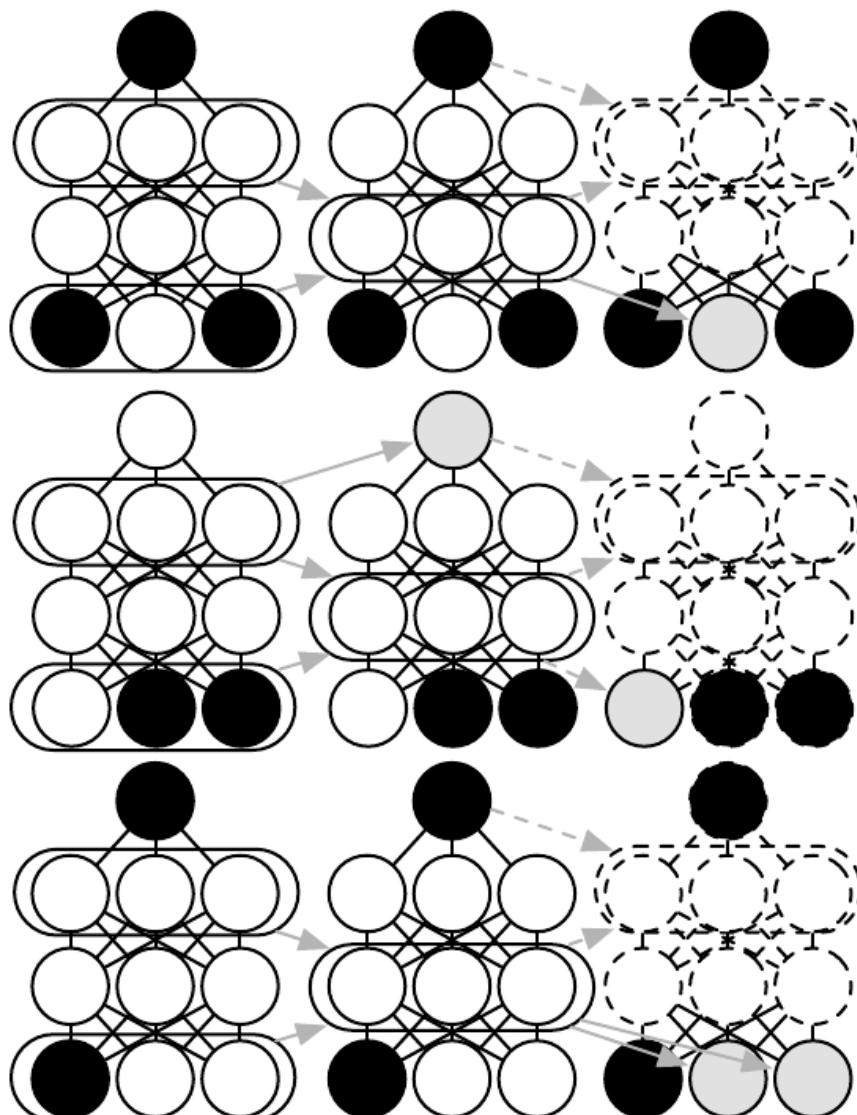


Figura 20.5: Una ilustración del proceso de entrenamiento de predicción múltiple para una máquina profunda de Boltzmann. Cada fila indica un ejemplo diferente dentro de un minibatch para el mismo paso de entrenamiento. Cada columna representa un paso de tiempo dentro del proceso de inferencia del campo medio. Para cada ejemplo, muestreamos un subconjunto de las variables de datos para que sirvan como entradas para el proceso de inferencia. Estas variables están sombreadas en negro para indicar el acondicionamiento. Luego ejecutamos el proceso de inferencia del campo medio, con flechas que indican qué variables influyen en qué otras variables en el proceso. En aplicaciones prácticas, desplegamos el campo medio en varios pasos. En esta ilustración, desenrollamos solo dos pasos. Las flechas discontinuas indican cómo se podría desarrollar el proceso para obtener más pasos. Las variables de datos que no se usaron como entradas para el proceso de inferencia se convierten en objetivos, sombreados en gris. Podemos ver el proceso de inferencia de cada ejemplo como una red recurrente. Usamos gradiente descendente y retropropagación para entrenar estas redes recurrentes para producir los objetivos correctos dadas sus entradas. Esto entrena el proceso de campo medio para que el MP-DBM produzca estimaciones precisas. Figura adaptada de [Buen compañero et al. \(2013b\)](#).

Los DBM pueden entrenarse de manera conjunta, mientras que los DBM requieren un preentrenamiento codicioso por capas. La desventaja de la retropropagación a través del gráfico de inferencia aproximada es que no proporciona una forma de optimizar la verosimilitud logarítmica, sino una aproximación heurística de la pseudoverosimilitud generalizada.

El MP-DBM inspiró la NADE- k ([Raiko et al., 2014](#)) extensión al marco NADE, que se describe en la sección[20.10.10](#).

El MP-DBM tiene algunas conexiones con la deserción. Dropout comparte los mismos parámetros entre muchos gráficos computacionales diferentes, siendo la diferencia entre cada gráfico si incluye o excluye cada unidad. El MP-DBM también comparte parámetros en muchos gráficos computacionales. En el caso del MP-DBM, la diferencia entre los gráficos es si se observa o no cada unidad de entrada. Cuando no se observa una unidad, el MP-DBM no la elimina por completo como lo hace el abandono. En cambio, el MP-DBM lo trata como una variable latente para ser inferida. Uno podría imaginar aplicar abandono al MP-DBM eliminando adicionalmente algunas unidades en lugar de hacerlas latentes.

20.5 Máquinas de Boltzmann para datos con valores reales

Si bien las máquinas de Boltzmann se desarrollaron originalmente para su uso con datos binarios, muchas aplicaciones, como el modelado de imágenes y audio, parecen requerir la capacidad de representar distribuciones de probabilidad sobre valores reales. En algunos casos, es posible tratar los datos de valor real en el intervalo [0, 1] como si representaran la expectativa de una variable binaria. Por ejemplo,[Hinton\(2000\)](#) trata las imágenes en escala de grises en el conjunto de entrenamiento como si definieran valores de probabilidad [0,1]. Cada píxel define la probabilidad de que un valor binario sea 1, y todos los píxeles binarios se muestran independientemente unos de otros. Este es un procedimiento común para evaluar modelos binarios en conjuntos de datos de imágenes en escala de grises. Sin embargo, no es un enfoque particularmente satisfactorio desde el punto de vista teórico, y las imágenes binarias muestreadas de forma independiente de esta manera tienen una apariencia ruidosa. En esta sección, presentamos máquinas de Boltzmann que definen una densidad de probabilidad sobre datos de valor real.

20.5.1 RBM gaussiana-bernoulli

Se pueden desarrollar máquinas de Boltzmann restringidas para muchas distribuciones condicionales de familias exponenciales ([brotando et al., 2005](#)). De estos, el más común es el RBM con unidades ocultas binarias y unidades visibles de valor real, siendo la distribución condicional sobre las unidades visibles una distribución gaussiana cuya media es una función de las unidades ocultas.

Hay muchas formas de parametrizar RBM de Gaussian-Bernoulli. Una opción es utilizar una matriz de covarianza o una matriz de precisión para la distribución gaussiana. Aquí presentamos la formulación de precisión. La modificación para obtener la formulación de covarianza es directa. Deseamos tener la distribución condicional

$$p_{\text{ag}}(v | h) = \text{norte}(v; \zeta \text{Qué}? - 1). \quad (20.38)$$

Podemos encontrar los términos que necesitamos agregar a la función de energía expandiendo la distribución condicional logarítmica no normalizada:

$$\text{registro} \text{norte}(v; \zeta \text{Qué}? - 1) = -\frac{1}{2}(v - Wh) \cdot \beta(v - Wh) + F(\beta). \quad (20.39)$$

Aquí F encapsula todos los términos que son una función solo de los parámetros y no de las variables aleatorias en el modelo. podemos descartar F porque su único papel es normalizar la distribución, y la función de partición de cualquier función de energía que elijamos desempeñará ese papel.

Si incluimos todos los términos (con el signo invertido) que implican v de la ecuación 20.39 en nuestra función de energía y no agrega ningún otro término que involucre v , entonces nuestra función de energía representará el condicional deseado $p_{\text{ag}}(v | h)$.

Tenemos cierta libertad con respecto a la otra distribución condicional, $p_{\text{ag}}(h | v)$. Tenga en cuenta que la ecuación 20.39 contiene un término

$$\frac{1}{2}h \cdot W \cdot \beta W h. \quad (20.40)$$

Este término no puede incluirse en su totalidad porque incluye $h_i h_j$ términos. Estos corresponden a los bordes entre las unidades ocultas. Si incluyéramos estos términos, tendríamos un modelo factorial lineal en lugar de una máquina de Boltzmann restringida. Al diseñar nuestra máquina Boltzmann, simplemente omitimos estos $h_i h_j$ términos cruzados. Omitirlos no cambia el condicional $p_{\text{ag}}(v | h)$ entonces ecuación 20.39 todavía se respeta. Sin embargo, todavía tenemos la opción de incluir los términos que involucran solo un h_i . Si asumimos una matriz de precisión diagonal, encontramos que para cada unidad h_i tenemos un término

$$\frac{1}{2}h_i - \sum_j \beta_j W_{ji}. \quad (20.41)$$

En lo anterior, usamos el hecho de que $h_i = h_j$ porque $h_i \in \{0, 1\}$. Si incluimos este término (con su signo invertido) en la función de energía, entonces naturalmente sesgará h_i para desactivarse cuando los pesos de esa unidad son grandes y están conectados a unidades visibles con alta precisión. La elección de incluir o no este término de sesgo no afecta la familia de distribuciones que el modelo puede representar (suponiendo que

incluimos parámetros de sesgo para las unidades ocultas), pero sí afecta la dinámica de aprendizaje del modelo. Incluir el término puede ayudar a que las activaciones de unidades ocultas sigan siendo razonables incluso cuando los pesos aumentan rápidamente en magnitud.

Una forma de definir la función de energía en un RBM de Gauss-Bernoulli es, por lo tanto,

$$m(v, h) = v(\frac{1}{2}\beta - \nu) - (\nu - \beta) \cdot Wh - \text{segundo} \cdot h \quad (20.42)$$

pero también podemos agregar términos adicionales o parametrizar la energía en términos de varianza en lugar de precisión, si así lo deseamos.

En esta derivación, no hemos incluido un término de sesgo en las unidades visibles, pero uno podría agregarse fácilmente. Una última fuente de variabilidad en la parametrización de un RBM Gaussiano-Bernoulli es la elección de cómo tratar la matriz de precisión. Puede fijarse en una constante (quizás estimada en función de la precisión marginal de los datos) o aprenderse. También puede ser un escalar multiplicado por la matriz identidad, o puede ser una matriz diagonal. Por lo general, no permitimos que la matriz de precisión no sea diagonal en este contexto, porque algunas operaciones en la distribución gaussiana requieren invertir la matriz, y una matriz diagonal se puede invertir trivialmente. En las secciones siguientes, veremos que otras formas de máquinas de Boltzmann permiten modelar la estructura de covarianza, usando varias técnicas para evitar invertir la matriz de precisión.

20.5.2 Modelos no dirigidos de covarianza condicional

Si bien el RBM gaussiano ha sido el modelo canónico de energía para datos de valor real, [Ranzato et al. \(2010a\)](#) argumentan que el sesgo inductivo de Gaussian RBM no se adapta bien a las variaciones estadísticas presentes en algunos tipos de datos de valor real, especialmente imágenes naturales. El problema es que gran parte del contenido de información presente en las imágenes naturales está incrustado en la covarianza entre píxeles en lugar de en los valores de píxeles sin procesar. En otras palabras, son las relaciones entre píxeles y no sus valores absolutos donde reside la mayor parte de la información útil en las imágenes. Dado que el RBM gaussiano solo modela la media condicional de la entrada dadas las unidades ocultas, no puede capturar información de covarianza condicional. En respuesta a estas críticas, se han propuesto modelos alternativos que intentan explicar mejor la covarianza de los datos de valor real. Estos modelos incluyen la media y la covarianza RBM (mcRBM¹), el producto medio de *t*-modelo de distribución (mPoT) y el RBM de punta y losa (ssRBM).

¹El término "mcRBM" se pronuncia diciendo el nombre de las letras MCRBM; el "mc" no se pronuncia como el "Mc" en "McDonald's".

Media y Covarianza RBM El mcRBM utiliza sus unidades ocultas para codificar de forma independiente la media condicional y la covarianza de todas las unidades observadas. La capa oculta mcRBM se divide en dos grupos de unidades: unidades medias y unidades de covarianza. El grupo que modela la media condicional es simplemente un RBM gaussiano. La otra mitad es una covarianza RBM (Ranzato et al., 2010a), también llamado cRBM, cuyos componentes modelan la estructura de covarianza condicional, como se describe a continuación.

Específicamente, con unidades medias binarias $h_{(metro)}$ y unidades de covarianza binaria $h_{(C)}$, el modelo mcRBM se define como la combinación de dos funciones de energía:

$$m_{imc}(x, h_{(metro)}, h_{(C)}) = m_{imetro}(x, h_{(metro)}) + m_{ic}(x, h_{(C)}), \quad (20.43)$$

dónde m_{imetro} es la función de energía RBM de Gaussian-Bernoulli estándar:²

$$m_{imetro}(x, h_{(metro)}) = \frac{1}{2} \sum_j (X \cdot W \cdot j^{h_{(metro)}} - b_{j(h_{(metro})})^2, \quad (20.44)$$

y m_{ic} es la función de energía cRBM que modela la información de covarianza condicional:

$$m_{ic}(x, h_{(C)}) = \frac{1}{2} \sum_j h_{(C)}^j \|X \cdot r^j\|^2 - b_{j(h_{(C)})}^2. \quad (20.45)$$

El parámetro r^j corresponde al vector de peso de covarianza asociado con $h_{(C)}$ y $b_{j(h_{(C)})}$ es un vector de compensaciones de covarianza. La función de energía combinada define una distribución conjunta:

$$p_{agmc}(x, h_{(metro)}, h_{(C)}) = \frac{1}{Z} \exp^{-m_{imc}(x, h_{(metro)}, h_{(C)})}, \quad (20.46)$$

y una distribución condicional correspondiente sobre las observaciones dadas $h_{(metro)}$ y $h_{(C)}$ como una distribución gaussiana multivariada:

$$p_{agmc}(x / h_{(metro)}, h_{(C)}) = \text{norte-} X; C_{X/h} \sum_j W \cdot j^{h_{(metro)}}, C_{X/h}^{mc}. \quad (20.47)$$

Tenga en cuenta que la matriz de covarianza $C_{X/h} = \sum_j h_{(j)}^T C_{rjr} h_{(j)}$ no es diagonal y eso W es la matriz de peso asociada con el RBM gaussiano que modela el

²Esta versión de la función de energía RBM de Gaussian-Bernoulli asume que los datos de la imagen tienen una media de cero, por píxel. Las compensaciones de píxeles se pueden agregar fácilmente al modelo para tener en cuenta las medias de píxeles distintas de cero.

medios condicionales. Es difícil entrenar el mcRBM mediante divergencia contrastiva o divergencia contrastiva persistente debido a su estructura de covarianza condicional no diagonal. CD y PCD requieren muestreo de la distribución conjunta $\text{dex}, h_{(\text{metro})}, h_{(C)}$ lo cual, en un RBM estándar, se logra mediante el muestreo de Gibbs sobre los condicionales. Sin embargo, en el mcRBM, el muestreo de $\text{pag}_{\text{mc}}(x / h_{(\text{metro})}, h_{(C)})$ requiere informática (C_{mc})-1 en cada iteración del aprendizaje. Esto puede ser una carga computacional poco práctica para observaciones más grandes. Ranzato y Hinton (2010) evitar el muestreo directo del condicional $\text{pag}_{\text{mc}}(x / h_{(\text{metro})}, h_{(C)})$ tomando muestras directamente de la marginal $\text{pag}(X)$ utilizando hamiltoniano (híbrido) Monte Carlo (Neal, 1993) sobre la energía libre mcRBM.

Producto medio de Student t -distribuciones El producto medio de Student t -modelo de distribución (mPoT) (Ranzato et al., 2010b) extiende el modelo PoT (Brotando et al., 2003a) de manera similar a cómo el mcRBM extiende el cRBM. Esto se logra al incluir medias gaussianas distintas de cero mediante la adición de unidades ocultas de tipo RBM gaussianas. Al igual que el mcRBM, la distribución condicional PoT sobre la observación es una distribución gaussiana multivariada (con covarianza no diagonal); sin embargo, a diferencia del mcRBM, la distribución condicional complementaria sobre las variables ocultas viene dada por distribuciones Gamma condicionalmente independientes. La distribución gamma $GRAMO(k, \theta)$ es una distribución de probabilidad sobre números reales positivos, con media $k\theta$. No es necesario tener una comprensión más detallada de la distribución Gamma para comprender las ideas básicas que subyacen al modelo mPoT.

La función de energía mPoT es:

$$mimPoT(x, h_{(\text{metro})}, h_{(C)}) \quad (20.48)$$

$$= mimetro(x, h_{(\text{metro})}) + \sum_j \frac{h_{(C)} - 1 + \frac{1}{2} \cdot r_j \cdot X}{j} + (1 - y) \text{registro} h_{(P)} \quad (20.49)$$

dónde r_j es el vector de peso de covarianza asociado con la unidad $h_{(C)}_j$ y $mimetro(x, h_{(\text{metro})})$ es como se define en la ecuación 20.44.

Al igual que con el mcRBM, la función de energía del modelo mPoT especifica una Gaussiana multivariante, con una distribución condicional sobre X que tiene covarianza no diagonal. El aprendizaje en el modelo mPoT, nuevamente, como el mcRBM, se complica por la incapacidad de muestrear del condicional gaussiano no diagonal. $\text{pag}_{\text{mPoT}}(x / h_{(\text{metro})}, h_{(C)})$, entonces Ranzato et al. (2010b) también defienden el muestreo directo de $\text{pag}(X)$ a través de Monte Carlo hamiltoniano (híbrido).

Máquinas Boltzmann restringidas con puntas y losas Máquinas Boltzmann restringidas de puntas y losas ([courville et al., 2011](#)) o ssRBM proporcionan otro medio para modelar la estructura de covarianza de los datos de valor real. En comparación con los mcRBM, los ssRBM tienen la ventaja de que no requieren inversión de matriz ni métodos hamiltonianos de Monte Carlo. Al igual que el modelo mcRBM y mPoT, las unidades ocultas binarias de ssRBM codifican la covarianza condicional entre píxeles mediante el uso de variables auxiliares de valor real.

El RBM de espiga y losa tiene dos conjuntos de unidades ocultas: binario **espiga** unidades h , y de valor **reallosa** unidades s . La media de las unidades visibles condicionadas a las unidades ocultas está dada por $(h-s)W$. En otras palabras, cada columna $W_{:,i}$ define un componente que puede aparecer en la entrada cuando $h=1$. La variable pico correspondiente h determina si ese componente está presente en absoluto. La variable de losa correspondiente s determina la intensidad de ese componente, si está presente. Cuando una variable de pico está activa, la variable de losa correspondiente agrega varianza a la entrada a lo largo del eje definido por $W_{:,i}$. Esto nos permite modelar la covarianza de las entradas. Afortunadamente, la divergencia contrastiva y la divergencia contrastiva persistente con el muestreo de Gibbs siguen siendo aplicables. No hay necesidad de invertir ninguna matriz.

Formalmente, el modelo ssRBM se define a través de su función de energía:

$$m_{iss}(x, s, h) = - \sum_i X W_{:,i} s_i h_i + \frac{1}{2} X \Lambda + \sum_i \Phi_i h_i X \quad (20.50)$$

$$+ \frac{1}{2} \sum_i a_i s_i^2 - \sum_i a_i \mu_i s_i h_i - \sum_i b_i h_i + \sum_i a_i \mu_i^2 h_i, \quad (20.51)$$

dónde b_i es el desplazamiento del pico h y Λ es una matriz de precisión diagonal sobre las observaciones X . El parámetro $a_i > 0$ es un parámetro de precisión escalar para la variable de losa de valor real s_i . El parámetro Φ_i es una matriz diagonal no negativa que define un h -penalización cuadrática modulada en X . Cada μ_i es un parámetro medio para la variable de losas s_i .

Con la distribución conjunta definida a través de la función de energía, es relativamente sencillo derivar las distribuciones condicionales ssRBM. Por ejemplo, al marginar las variables de losas, la distribución condicional sobre las observaciones dadas las variables pico binarias h es dado por:

$$p_{agss}(x / h) = \frac{1}{PAG(h)Z} \text{Exp}\{-m(x, s, h)\}ds \quad (20.52)$$

$$= norteX; C_{ss} \sum_i W_{:,i} \mu_i h_i C_{ss}^{-1} x/h \quad (20.53)$$

dónde $C_{ss/h} = \Lambda + \Phi_i h_i - i a_i \tilde{h}_i W_{:,i}^{-1}$. La última igualdad se cumple sólo si la matriz de covarianza $C_{ss/h}$ es definida positiva.

Activar por las variables pico significa que la verdadera distribución marginal sobre h -s es escaso. Esto es diferente de la codificación escasa, donde las muestras del modelo "casi nunca" (en el sentido teórico de la medida) contienen ceros en el código, y se requiere la inferencia MAP para imponer la escasez.

Al comparar el ssRBM con los modelos mcRBM y mPoT, el ssRBM parametriza la covarianza condicional de la observación de una manera significativamente diferente. forma-. El mcRBM y mP-o ambos modelan la estructura de covarianza de la observación como $\tilde{h}_j \tilde{h}_j^T$, utilizando la activación de las unidades ocultas $h_j > 0$ a imponer restricciones sobre la covarianza condicional en la dirección τ_j . Por el contrario, el ssRBM especifica la covarianza condicional de las observaciones utilizando las activaciones de picos ocultos $h=1$ para pellizcar la matriz de precisión a lo largo de la dirección especificada por el vector de peso correspondiente. La covarianza condicional de ssRBM es muy similar a la dada por un modelo diferente: el producto del análisis probabilístico de componentes principales (PoPPCA) ([Williams y Agakov, 2002](#)). En la configuración sobrecompleta, las activaciones dispersas con la parametrización ssRBM permiten una variación significativa (por encima de la variación nominal dada por $\Lambda-1$) solo en las direcciones seleccionadas de los escasamente activados h_i . En los modelos mcRBM o mPoT, una representación demasiado completa significaría que capturar la variación en una dirección particular en el espacio de observación requiere eliminar potencialmente todas las restricciones con proyección positiva en esa dirección. Esto sugeriría que estos modelos se adaptan menos a la configuración demasiado completa.

La principal desventaja de la máquina de Boltzmann restringida de puntas y losas es que algunos ajustes de los parámetros pueden corresponder a una matriz de covarianza que no es definida positiva. Tal matriz de covarianza coloca una probabilidad más no normalizada en valores que están más alejados de la media, lo que hace que la integral sobre todos los resultados posibles diverja. En general, este problema se puede evitar con simples trucos heurísticos. Todavía no hay ninguna solución teóricamente satisfactoria. El uso de optimización restringida para evitar explícitamente las regiones donde la probabilidad no está definida es difícil de hacer sin ser demasiado conservador y también evitando que el modelo acceda a regiones de alto rendimiento del espacio de parámetros.

Cualitativamente, las variantes convolucionales del ssRBM producen excelentes muestras de imágenes naturales. Algunos ejemplos se muestran en la figura [16.1](#).

El ssRBM permite varias extensiones. Incluyendo interacciones de orden superior y agrupación promedio de las variables de losa ([courville et al., 2014](#)) permite que el modelo aprenda características excelentes para un clasificador cuando los datos etiquetados son escasos. Agregar un

término a la función de energía que evita que la función de partición se vuelva indefinida da como resultado un modelo de codificación escasa, codificación dispersa de pico y losa ([Buen compañero et al., 2013d](#)), también conocido como S3C.

20.6 Máquinas convolucionales de Boltzmann

Como se vio en el capítulo 9, las entradas dimensionales extremadamente altas, como las imágenes, ejercen una gran presión sobre los requisitos de computación, memoria y estadísticas de los modelos de aprendizaje automático. Reemplazar la multiplicación de matrices por convolución discreta con un núcleo pequeño es la forma estándar de resolver estos problemas para entradas que tienen una estructura espacial o temporal invariable en la traducción. [Desjardins y Bengio\(2008\)](#) mostró que este enfoque funciona bien cuando se aplica a los RBM.

Las redes convolucionales profundas generalmente requieren una operación de agrupación para que el tamaño espacial de cada capa sucesiva disminuya. Las redes convolucionales de avance a menudo utilizan una función de agrupación, como el máximo de los elementos que se agruparán. No está claro cómo generalizar esto al entorno de los modelos basados en la energía. Podríamos introducir una unidad de agrupación binaria p sobre n unidades detectores binarias y hacer cumplir $p = \max(d)$ ajustando la función de energía para que sea ∞ cada vez que se viola esa restricción. Sin embargo, esto no escala bien, ya que requiere evaluar 2^n diferentes configuraciones de energía para calcular la constante de normalización. Para un pequeño 3×3 región de agrupación esto requiere $2^9 = 512$ evaluaciones de la función de energía por unidad de pooling!

[Sotavento et al.\(2009\)](#) desarrollaron una solución a este problema llamada **agrupación máxima probabilística** (no debe confundirse con la "agrupación estocástica", que es una técnica para construir implícitamente conjuntos de redes convolucionales de realimentación). La estrategia detrás de la agrupación máxima probabilística es restringir las unidades del detector para que, como máximo, una pueda estar activa a la vez. Esto significa que solo hay $n+1$ estados totales (un estado para cada uno de los n unidades detectores encendidas y un estado adicional correspondiente a todas las unidades detectores apagadas). La unidad de agrupación está encendida si y solo si una de las unidades detectores está encendida. Al estado con todas las unidades apagadas se le asigna energía cero. Podemos pensar en esto como la descripción de un modelo con una sola variable que tiene $n+1$ estados, o de manera equivalente como un modelo que tiene $n+1$ variables que asigna energía ∞ a todos menos $n+1$ asignaciones conjuntas de variables.

Si bien la agrupación máxima probabilística eficiente obliga a las unidades detectores a ser mutuamente excluyentes, lo que puede ser una restricción de regularización útil en algunos contextos o un límite dañino en la capacidad del modelo en otros contextos. Tampoco admite regiones de agrupación superpuestas. Por lo general, se requieren regiones de agrupación superpuestas para obtener el mejor rendimiento de las redes convolucionales de avance, por lo que esta restricción probablemente reduce en gran medida el rendimiento de Boltzmann convolucional.

máquinas.

Sotavento et al.(2009) demostró que la agrupación máxima probabilística podría usarse para construir máquinas de Boltzmann profundas convolucionales.³Este modelo puede realizar operaciones como completar las partes faltantes de su entrada. Si bien es intelectualmente atractivo, es difícil hacer que este modelo funcione en la práctica y, por lo general, no funciona tan bien como clasificador como las redes convolucionales tradicionales entrenadas con aprendizaje supervisado.

Muchos modelos convolucionales funcionan igual de bien con entradas de diferentes tamaños espaciales. Para las máquinas Boltzmann, es difícil cambiar el tamaño de entrada por una variedad de razones. La función de partición cambia a medida que cambia el tamaño de la entrada. Además, muchas redes convolucionales logran la invariancia del tamaño aumentando el tamaño de sus regiones de agrupación proporcionalmente al tamaño de la entrada, pero escalar las regiones de agrupación de máquinas de Boltzmann es complicado. Las redes neuronales convolucionales tradicionales pueden usar un número fijo de unidades de agrupación y aumentar dinámicamente el tamaño de sus regiones de agrupación para obtener una representación de tamaño fijo de una entrada de tamaño variable. Para las máquinas de Boltzmann, las grandes regiones de agrupación se vuelven demasiado costosas para el enfoque ingenuo. el enfoque de Sotavento et al.(2009) de hacer que cada una de las unidades detectoras en la misma región de agrupación sea mutuamente excluyente resuelve los problemas de cálculo, pero todavía no permite regiones de agrupación de tamaño variable. Por ejemplo, supongamos que aprendemos un modelo con 2×2 agrupación máxima probabilística sobre unidades de detectores que aprenden detectores de borde. Esto impone la restricción de que solo uno de estos bordes puede aparecer en cada 2×2 región. Si luego aumentamos el tamaño de la imagen de entrada en un 50 % en cada dirección, esperaríamos que el número de bordes aumentara correspondientemente. En cambio, si aumentamos el tamaño de las regiones de agrupación en un 50% en cada dirección para 3×3 , entonces la restricción de exclusividad mutua ahora especifica que cada uno de estos bordes solo puede aparecer una vez en un 3×3 región. A medida que hacemos crecer la imagen de entrada de un modelo de esta manera, el modelo genera bordes con menos densidad. Por supuesto, estos problemas solo surgen cuando el modelo debe usar cantidades variables de agrupación para emitir un vector de salida de tamaño fijo. Los modelos que utilizan la agrupación máxima probabilística aún pueden aceptar imágenes de entrada de tamaño variable siempre que la salida del modelo sea un mapa de características que pueda escalar en tamaño proporcional a la imagen de entrada.

Los píxeles en el límite de la imagen también plantean cierta dificultad, que se ve agravada por el hecho de que las conexiones en una máquina de Boltzmann son simétricas. Si no rellenamos implícitamente con ceros la entrada, entonces hay menos unidades ocultas que unidades visibles, y las unidades visibles en el límite de la imagen no se modelan.

³La publicación describe el modelo como una "red de creencias profundas", pero debido a que puede describirse como un modelo puramente no dirigido con actualizaciones de puntos fijos de campo medio manejables por capas, se ajusta mejor a la definición de una máquina de Boltzmann profunda.

bien porque se encuentran en el campo receptivo de menos unidades ocultas. Sin embargo, si implícitamente rellenamos con ceros la entrada, entonces las unidades ocultas en el límite son impulsadas por menos píxeles de entrada y es posible que no se activen cuando sea necesario.

20.7 Máquinas de Boltzmann para salidas estructuradas o secuenciales

En el escenario de salida estructurada, deseamos entrenar un modelo que pueda mapear desde alguna entrada X a alguna salida y , y las diferentes entradas de X están relacionados entre sí y deben obedecer algunas restricciones. Por ejemplo, en la tarea de síntesis de voz, y es una forma de onda, y toda la forma de onda debe sonar como un enunciado coherente.

Una forma natural de representar las relaciones entre las entradas en y es utilizar una distribución de probabilidad $p(y|X)$. Las máquinas de Boltzmann, extendidas para modelar distribuciones condicionales, pueden suministrar este modelo probabilístico.

La misma herramienta de modelado condicional con una máquina de Boltzmann se puede utilizar no solo para tareas de salida estructurada, sino también para el modelado de secuencias. En el último caso, en lugar de mapear una entrada X a una salida y , el modelo debe estimar una distribución de probabilidad sobre una secuencia de variables, $p(y|X(1), \dots, X(t))$. Las máquinas condicionales de Boltzmann pueden representar factores de la forma $p(y|X(t)/X(1), \dots, X(t-1))$ para cumplir con esta tarea.

Una importante tarea de modelado de secuencias para la industria de los videojuegos y el cine es el modelado de secuencias de ángulos de unión de esqueletos utilizados para renderizar personajes en 3D. Estas secuencias a menudo se recopilan mediante sistemas de captura de movimiento para registrar los movimientos de los actores. Un modelo probabilístico del movimiento de un personaje permite generar animaciones nuevas, inéditas pero realistas. Para resolver esta tarea de modelado de secuencias, [taylor et al. \(2007\)](#) introdujo un modelo RBM condicional $p(y|X(t)/X(1), \dots, X(t-m))$. Para pequeños m , el modelo es un RBM sobre $p(y|X(t))$ cuyos parámetros de sesgo son una función lineal de los valores anteriores de X . Cuando condicionamos a diferentes valores de $X(t-1)$ y variables anteriores, obtenemos un nuevo RBM sobre X . Los pesos en el RBM sobre X nunca cambian, pero condicionando diferentes valores pasados, podemos cambiar la probabilidad de que diferentes unidades ocultas en el RBM estén activas. Al activar y desactivar diferentes subconjuntos de unidades ocultas, podemos realizar grandes cambios en la distribución de probabilidad inducida en X . Otras variantes de RBM condicional ([Mnih et al., 2011](#)) y son posibles otras variantes de modelado de secuencias utilizando RBM condicionales ([Taylor y Hinton, 2009; Sutskever et al., 2009; Boulanger-Lewandowski et al., 2012](#)).

Otra tarea de modelado de secuencias es modelar la distribución sobre secuencias