Función de Pérdida

Nicolás Kossacoff

Noviembre 2024

1. Introducción

Una **función de pérdida** mide que tan buena es la predicción de nuestro modelo, $\hat{y}_i$, con respecto al verdadero valor (ground-truth), y_i. Por ejemplo, en un problema de clasificación, la función de pérdida compara las probabilidades estimadas por el modelo con el ground-truth. En ese caso toma valores altos si el modelo le da más probabilidad a una clase que no es la correcta y toma valores chicos si el modelo le da más probabilidad a la clase correcta.

Cuando entrenamos una red neuronal, buscamos encontrar los parámetros que minimicen la función de pérdida. Esos parámetros nos devuelven el modelo que mejor ajusta a los datos.

2. Máxima Verosimilitud

Supongamos que queremos que nuestro modelo estime la **distribución condicional** de y_i dado x_i, como en la figura 1. Para lograr esto primero definimos una distribución paramétrica, $Pr(y|\theta)$, la cual elegimos en base al dominio de y (e.g., si $y \in \mathbb{R}$, entonces podríamos elegir una distribución Gaussiana), y luego adaptamos nuestro modelo para que compute uno o más parámetros de esa distribución (e.g., en el caso de la distribución Gaussiana, tendríamos una red neuronal con dos neuronas de salida, una para la media y otra para la varianza).

Ahora para cada observación x_i tenemos una distribución con parámetros θ_i. Cada observación de y_i tiene que tener una probabilidad alta bajo esta distribución estimada. Dicho esto, elegimos los parámetros de nuestro modelo tal que maximicen la probabilidad conjunta de nuestros datos[1], es decir, que hagan más probable observar y_i dado que observamos x_i (mirar figura 2). A la probabilidad conjunta se la conoce

[1] Acá estamos, implícitamente, haciendo dos supuestos: (i) asumimos que todas las observaciones siguen la misma distribución conjunta, y (ii) asumimos que las distribuciones son independientes.

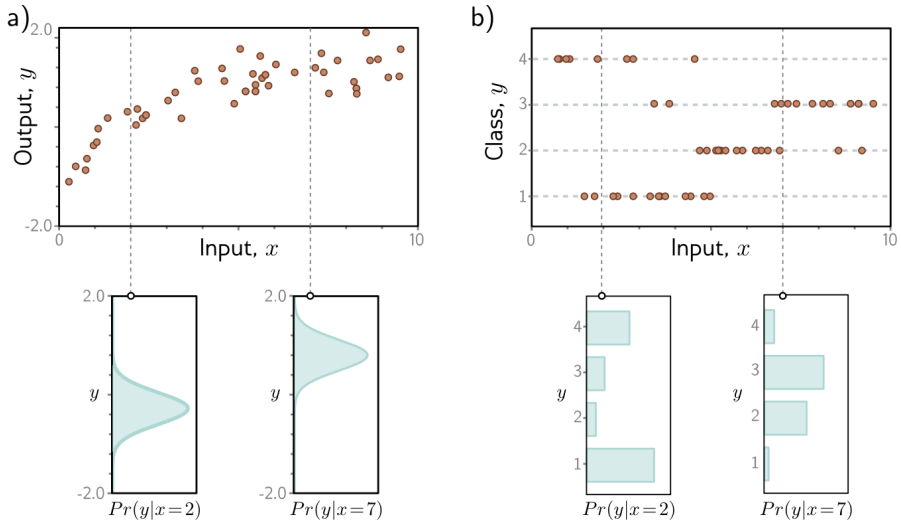

Figura 1: *Izquierda.* Estimación de la distribución condicional de y_i dada la observación x_i en un contexto de regresión. *Derecha.* Estimación de la distribución condicional en un contexto de clasificación. **Source:** Price, S. (2024). *Understanding Deep Learning* [Figura 5.1, p. 57]

como función de verosimilitud.

$$\begin{aligned}\hat{W} &= \arg\max_{W} \prod_{i=1}^{n} Pr\left(y_i|x_i\right) \\ &= \arg\max_{W} \prod_{i=1}^{n} Pr\left(y_i|\theta_i\right) \\ &= \arg\max_{W} \prod_{i=1}^{n} Pr\left(y_i|f(x_i, W)\right)\end{aligned}$$

Aplicamos un logaritmo para obtener la función de log-verosimilitud:

$$L(W) = \sum_{i=1}^{n} \log\left(Pr\left(y_i|f(x_i, W)\right)\right)$$

Dado que el logaritmo es una función monótona creciente (i.e., si $z > z'$ entonces $\log(z) > \log(z')$, y viceversa), entonces el máximo es el mismo para ambas funciones. Esto quiere decir que si encontramos los parámetros que maximizan la log-verosimilitud, implícitamente vamos a encontrar los parámetros que maximizan la verosimilitud.

Como se suele minimizar la función de pérdida, y no maximizar, agregamos un menos adelante:

$$L(W) = \sum_{i=1}^{n} \log\left(Pr\left(y_i|f(x_i, W)\right)\right) \tag{1}$$

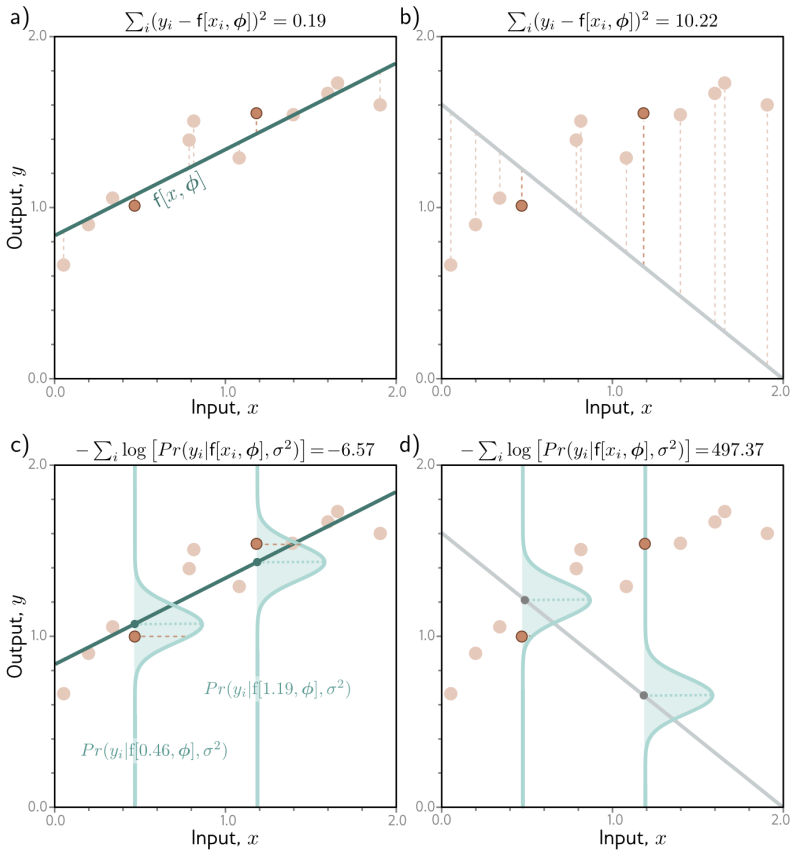

Figura 2: *Izquierda.* En la figura a) tenemos un modelo que ajusta bien a los datos. Lo que ocurre es que la probabilidad de observar y_i (punto rojo) dada la distribución Gaussiana estimada por nuestro modelo es alta (figura c). *Derecha.* En la figura b) tenemos el caso contrario en donde el modelo no ajusta bien a los datos. Lo que ocurre es que la probabilidad de observar y_i dada la distribución estimada por el modelo es baja. **Source:** Price, S. (2024). *Understanding Deep Learning* [Figura 5.4, p. 63]

2.1. Inferencia

Muchas veces necesitamos una estimación puntual de y y no su distribución. Se puede reportar el máximo valor de y_i para la distribución estimada:

$$\hat{y} = \arg\max_{y} Pr(y|f(x_i, \hat{W}))$$

Por ejemplo, en un caso de clasificación binaria, donde $y \in \{0, 1\}$, la distribución estimada tiene solo dos posibles valores: $P(y_i = 1|x_i) = \lambda$ y $P(y_i = 0|x_i) = 1 - \lambda$. Si $\lambda > 1 - \lambda$, entonces reportamos $y_i = 1$, y reportamos $y_i = 0$ en caso contrario.

2.2. Criterio de Máxima Verosimilitud

El criterio de máxima verosimilitud nos permite construir funciones de pérdida que se ajusten al problema que queremos resolver. Los pasos a seguir son:

1. Definir una distribución paramétrica, $Pr(y|\theta)$, según el dominio de la predicción y.

2. Definir nuestra red neuronal para que estime los parámetros de la distribución, θ.

3. Entrenamos el modelo buscando los parámetros W que minimicen la función de pérdida, es decir, la función de log-verosimilitud para todos nuestros datos:

$$\hat{W} = \arg\max_{W} -\sum_{i=1}^{n} \log\left(Pr\left(y_i|f(x_i, W)\right)\right)$$

4. Para realizar inferencia con nuevas observaciones devolvemos la distribución estimada o una estimación puntual, que es equivalente al valor de y con mayor probabilidad.

2.2.1. Ejemplo: regresión

En este caso queremos predecir una variable $y \in \mathbb{R}$ usando un modelo $f(x, W)$. Primero definimos la distribución en base al dominio de y. Dado que $y \in \mathbb{R}$, elegimos una distribución Gaussiana:

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

Definimos nuestro modelo $f(x, W)$ para que estime uno o más parámetros de la distribución. Por ejemplo, podemos definir nuestra red neuronal tal que solo estime la media de la distribución, $\mu = f(x, W)$ (asumimos que σ^2 es una constante):

$$Pr\left(y|f(x, W), \sigma^2\right) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{(y-f(x,W))^2}{2\sigma^2}\right)$$

Ahora, si tomamos la función pérdida en (1) y reemplazamos $Pr(y_i|\theta)$ por la distribución Gaussiana obtenemos el **error cuadrático medio**, una función de pérdida muy utilizada para problemas de regresión:

$$\begin{aligned}
L\left(W\right) &= -\sum_{i=1}^{n} \log\left(Pr\left(y_i|f(x_i, W), \sigma^2\right)\right) \\
&= -\sum_{i=1}^{n} \log\left(\frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{(y_i - f(x_i, W))^2}{2\sigma^2}\right)\right) \\
&= -\sum_{i=1}^{n} \log\left(\frac{1}{\sqrt{2\pi\sigma^2}}\right)\left(-\frac{(y_i - f(x_i, W))^2}{2\sigma^2}\right) \\
&= -\sum_{i=1}^{n} \left(y_i - f(x_i, W)^2\right)
\end{aligned}$$

Por último, minimizamos la función de pérdida para encontrar los parámetros óptimos:

$$\hat{W} = \arg\min_{W} L(W) = \arg\min_{W} -\sum_{i=1}^{n} \left(y_i - f(x_i, W)^2\right)$$

2.2.2. Ejemplo: clasificación binaria

En este caso la variable y puede tomar dos posibles valores $\{0, 1\}$, a los cuales llamamos etiquetas. Buscamos que nuestro modelo $f(x, W)$ clasifique las observaciones x bajo alguna de estas etiquetas.

Para este problema elegimos es la distribución Bernoulli, que se encuentra definida para el dominio $\{0, 1\}$. Esta distribución tiene un único parámetro, λ, el cual representa la probabilidad de que $y = 1$.

$$Pr(y|x) = Pr(y = 0|x)^{1-y} \cdot Pr(y = 1|x)^{y} = (1 - \lambda)^{1-y} \cdot \lambda^{y}$$

Como λ es una probabilidad, su valor está entre $[0, 1]$. Necesitamos entonces que nuestro modelo devuelva un valor entre $[0, 1]$ como estimación. Para eso, utilizamos como función de activación de nuestra neurona de salida a la función Sigmoidea:

$$z_i(x_i, W) = \frac{1}{1 + \exp\left(-f(x_i, W)\right)}$$

Si reemplazamos esto en la ecuación (1) obtenemos la **entropía cruzada binaria**, una función de pérdida muy utilizada para problemas de clasificación binaria:

$$L(W) = -\log\left([1 - z_i(x_i, W)]^{1-y_i} \cdot z_i(x_i, W)^{y_i}\right)$$
$$L(W) = -\log\left([1 - z_i(x_i, W)]^{1-y_i}\right) - \log\left(z_i(x_i, W)^{y_i}\right)$$

Por último, minimizamos la función de pérdida para encontrar los parámetros óptimos:

$$\hat{W} = \arg\min_{W} -\log\left([1 - z_i(x_i, W)]^{1-y_i}\right) - \log\left(z_i(x_i, W)^{y_i}\right)$$

2.2.3. Ejemplo: clasificación con múltiples clases

En este caso la variable y puede tomar K valores (etiquetas). Buscamos que nuestro modelo pueda asignarle una de estas K etiquetas a cada observación x.

Elegimos la distribución categórica, que se encuentra definida para este dominio. Esta distribución tiene K parámetros, donde cada uno de los parámetros define la probabilidad de cada etiqueta para la observación x, $Pr(y = k|x) = \lambda_k$. Nuestra red neuronal va a tener K neuronas de salida, una para cada parámetro.

Al igual que antes, necesitamos asegurarnos que todo λ_k sea una probabilidad. Esto no solo quiere decir que λ_k tome valores en $[0, 1]$, sino también que $\sum_k \lambda_k = 1$. Para que los valores de salida de nuestra red neuronal cumplan con estas condiciones, utilizamos una función de activación Softmax. Esta función toma el vector de resultados de nuestra red y devuelve un vector de la misma dimensión, pero en el que todos sus elementos toman valores en $[0, 1]$ y la suma es igual a 1.

Matemáticamente, al aplicar la función Softmax sobre el vector de valores de salida, f_y, obtenemos:

$$z_y = softmax(f_y) = \frac{\exp\left(f_y\right)}{\sum_{k=1}^{K} \exp\left(f_k\right)}$$

donde la función exponencial nos asegura que los resultados sean positivos y el denominador nos asegura que los K valores de salida sumen 1.

Reemplazamos en la ecuación (1) y obtenemos la **entropía cruzada multi-clase**, una función de pérdida muy utilizada para problemas de clasificación con más de dos clases:

$$
\begin{aligned}
L(W) &= -\sum_{i=1}^{n} \log\left(z_i(x, W)\right) \\
L(W) &= -\sum_{i=1}^{n} \log\left(softmax[f_y(x_i, W)]\right) \\
L(W) &= -\sum_{i=1}^{n} \log\left(\frac{\exp\left(f_{y_i}(x_i, W)\right)}{\sum_{k=1}^{K} \exp\left(f_k(x_i, W)\right)}\right) \\
L(W) &= -\sum_{i=1}^{n} \left[\left(f_{y_i}(x_i, W)\right) - \log\left(\sum_{k=1}^{K} \exp\left(f_k(x_i, W)\right)\right)\right]
\end{aligned}
$$

Por último, minimizamos la función de pérdida para encontrar los parámetros óptimos:

$$
\hat{W} = \arg\min_{W} -\sum_{i=1}^{n} \left[\left(f_{y_i}(x_i, W)\right) - \log\left(\sum_{k=1}^{K} \exp\left(f_k(x_i, W)\right)\right)\right]
$$