Autor: María José
Gómez Silva

CAPÍTULO 1. INTRODUCCIÓN AL MACHINE LEARNING.

1.5 Función de Pérdidas y Gradient Descent

FUNCIÓN DE PÉRDIDAS

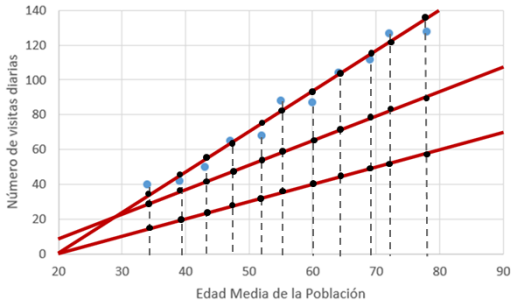
Como ya se ha comentado en secciones anteriores, el ajuste de los parámetros del modelo se realiza a partir del error cometido por las predicciones. En otras palabras, la decisión de cuánto hay que aumentar o disminuir los valores de los parámetros del modelo, se calcula en función de cuánto se ha desviado el modelo con respecto a algún tipo de referencia. En el caso del aprendizaje supervisado la referencia a seguir viene dada por las etiquetas de los datos.

La función encargada de medir la desviación del modelo con respecto a su objetivo se denomina *función de pérdidas o de costes*. En algunos textos también se le denomina *función objetivo*, dado que implícitamente define el objetivo final del modelo.

Como regla general, la función de pérdidas calcula la media del error cometido por el modelo para un conjunto de datos de entrada. En los ejemplos anteriores hemos medido el error, para cada dato de entrada, como la diferencia entre la predicción del modelo y la etiqueta correspondiente. Sin embargo, existen multitud de métricas disponibles para calcular tal error, dependiendo de las necesidades del problema. En consecuencia, la literatura presenta una gran variedad de funciones de pérdida con distintas formulaciones, cada una de ellas específica para un tipo de entrenamiento, de modelo, o adaptada a las características de los datos. Por ejemplo, la función de pérdidas denominada *Focal Loss Function* presenta una formulación que la hace apropiada para resolver problemas de clasificación cuando el número de ejemplos de datos disponibles de cada clase está muy desbalanceado.

En cada iteración del entrenamiento, la función de pérdidas mide la precisión del modelo con los valores actuales de los parámetros. La Figura 1 muestra los valores de la función de pérdidas calculados para distintos valores de los parámetros *a* y *b*. En este caso, se ha calculado el error como el valor absoluto de la diferencia entre las predicciones *y*, y las etiquetas de los datos, *y'*.

$$y_i = a + b * x_i$$



x=[34, 39, 43, 47, 52, 55, 60, 64, 69, 72, 78]
y'=[40, 42, 50, 65, 68, 88, 87, 104, 112, 127, 128]

a=-46.7 y *b*=2.33 $f_{loss} = 4$
y=[33, 44, 54, 63, 74, 81, 93, 102, 114, 121, 135]

a=-18 y *b*=1.4 $f_{loss} = 18.8$
y=[30, 37, 42, 48, 55, 59, 66, 72, 79, 83, 91]

a=-20 y *b*=1 $f_{loss} = 43.6$
y=[14, 19, 23, 27, 32, 35, 40, 44, 49, 52, 58]

Figura 1. Cálculo de la función de pérdidas para distintos valores de los parámetros del modelo.

Por lo tanto, el valor de la función de pérdidas depende de los valores de los parámetros del modelo. El objetivo del entrenamiento es encontrar los valores de los parámetros del modelo que minimizan el valor de la función de pérdidas.

GRADIENT DESCENT

En la sección anterior, se concluyó que el objetivo del entrenamiento es encontrar los valores de los parámetros del modelo que minimizan la función de pérdidas. La siguiente pregunta a plantearse sería ¿cómo se realiza esa minimización?

Si se conociese a priori como varía el valor de las pérdidas en función de los valores de los parámetros del modelo, podríamos buscar el valor mínimo mediante métodos aritméticos. En la Figura 2 se muestra un ejemplo de cómo varía el valor de las pérdidas para distintos valores de los parámetros *a* y *b*, dando lugar a una superficie. Sin embargo, la forma de tal superficie puede ser desconocida a priori o muy difícil de obtener. A medida que aumenta el número de parámetros, aumentan las dimensiones del problema y su complejidad. Además, el valor de la función de pérdidas también depende de la ecuación que define al modelo.

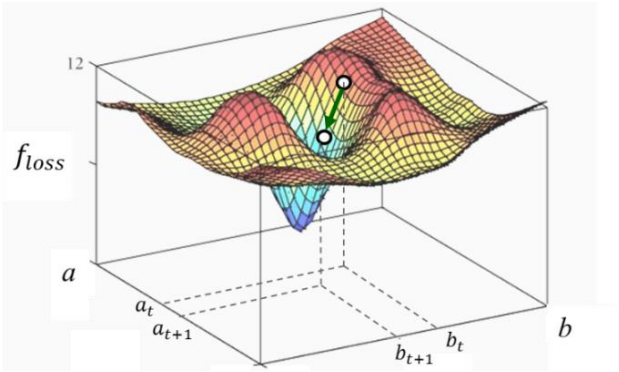


Figura 2. Valor de las pérdidas en función de los valores de los parámetros a y b.

Debido a la gran cantidad de parámetros que afectan al valor de las pérdidas, no es viable aplicar métodos de optimización aritméticos y se opta por seguir un proceso de exploración progresiva, paso a paso. De ahí el carácter iterativo o cíclico del entrenamiento.

En cada iteración del entrenamiento el valor de los parámetros debe ser actualizado. La idea es que esa actualización se realice de forma que, para los nuevos valores de los parámetros, la pérdida sea menor, o lo que es lo mismo, que en cada iteración el modelo sea más preciso y se desvíe menos de su objetivo.

De acuerdo con la figura anterior, se trataría de ir avanzado en la dirección en la que la superficie desciende hasta alcanzar el valor mínimo, o un valor cercano al mínimo. Para realizar ese proceso se emplea el método del *Descenso del Gradiente (Gradient Descent).*

Un gradiente en una operación matemática que describe una variación. Siguiendo con el ejemplo de la figura, si en un determinado momento del entrenamiento los parámetros del modelo toman los valores, a_t y b_t, podemos calcular el gradiente de la superficie en ese punto, y eso nos daría información de como varía la superficie en cualquier dirección desde ese punto. El gradiente nos da una idea de la pendiente en un punto. Lo que debemos hacer entonces es tomar la dirección que haga que el gradiente (la pendiente) sea negativo, es decir que la superficie descienda. Buscamos un descenso en el gradiente, de ahí el nombre del método.

El gradiente nos indica en qué dirección avanzar en cuanto a la actualización de los parámetros para minimizar el valor de las pérdidas, pero cuánto avanzamos viene determinado por otro parámetro, α, denominado *tasa de aprendizaje (learning rate*).

Finalmente, el valor de los parámetros, $p=[a,b,\ldots]$, es actualizado al final de cada ciclo de entrenamiento de acuerdo con la siguiente fórmula:

$$p_{t+1} = p_t - \alpha \nabla f_{loss}(p_t)$$

Donde p_t y p_{t+1} son los parámetros del modelo en un ciclo de entrenamiento y en el siguiente, y $\nabla f_{loss}(p_t)$ es el gradiente de la función de pérdidas en el punto dado por los valores de los parámetros p_t.

PROCESO ITERATIVO

Como se ha comentado en las secciones anteriores, el proceso de aprendizaje o entrenamiento de un modelo es iterativo. A continuación, se describe la secuencia de

pasos realizada en cada iteración, t, del algoritmo de aprendizaje del descenso del gradiente:

1. Se toman los datos de entrada X y sus etiquetas Y'.
2. Se calculan las predicciones del modelo Y.

   Las predicciones del modelo en cada iteración, t, depende de los valores de los parámetros en esa iteración, p_t.
3. Se calcula el valor de las pérdidas f_{loss}.

   La función de pérdidas compara Y' con Y. Dado que, Y es la predicción del modelo con los valores actuales de los parámetros, p_t, implícitamente la función de pérdidas depende de p_t.
4. Se calcula el gradiente de la función de pérdidas con respecto a los parámetros $\nabla f_{loss}(p_t)$.
5. Se actualizan los valores de los parámetros $p_{t+1} = p_t - \alpha \nabla f_{loss}(p_t)$.

   p_{t+1} serán los valores de los parámetros en la iteración siguiente.

   Por lo tanto, implícitamente se actualiza el modelo.

Dependiendo de la cantidad de datos tomados como datos de entrada en cada iteración, aparecen distintas versiones del método del descenso del gradiente:

Gradient Descent: En cada iteración, los datos de entrada son el conjunto completo de datos de entrenamiento. Por lo tanto, en cada iteración todos los datos disponibles para el entrenamiento son estudiados. La actualización final de los parámetros del modelo está basada en la desviación del modelo con respecto al objetivo medida para todos los datos de entrenamiento.

Stochastic Gradient Descent: En cada iteración, se toma como dato de entrada una sola muestra de los datos de entrenamiento. Por lo tanto, la actualización final de los parámetros del modelo está basada en la desviación del modelo con respecto al objetivo medida para una única muestra. Por ese motivo, los valores de los pesos pueden oscilar bastante de una iteración a la siguiente. En este caso, son necesarias tantas iteraciones como datos de entrenamiento haya, para llegar a estudiarlos todos.

Mini-batch Gradient Descent: En cada iteración, se toma como datos de entrada un subconjunto de los datos de entrenamiento. Se trata de una solución a medio camino entre las anteriores y es la más empleada actualmente. Cuando se dispone de grandes bases de datos para entrenar, es inviable tomar todos los datos de entrenamiento en cada iteración, debido a las limitaciones de memoria de los procesadores. Tomando subconjuntos de datos, se evitan las oscilaciones propias del *Stochastic Gradient Descent* y la lentitud en el proceso de aprendizaje, propia del *Gradient Descent*.