Resumen del capítulo: Análisis de algoritmos

Complejidad computacional

El **tiempo de ejecución del algoritmo** no se mide en segundos. Está determinado por el número de operaciones elementales que realiza el algoritmo: sumas, multiplicaciones y comparaciones. El tiempo de ejecución en cualquier computadora generalmente se denomina **tiempo de ejecución real**. El tiempo de ejecución de un algoritmo también está determinado por sus argumentos.

Es imposible calcular el tiempo de ejecución de los algoritmos complejos. Es por eso que determinamos su **complejidad computacional**, o **tiempo de ejecución asintótico.** El término se basa en la palabra **asíntota**, que define una línea recta a la que se acerca una curva, pero no la cruza.

Vamos a expresar la longitud de la lista como n. El tiempo de trabajo es una función de n, *escrita como* $T(n)$. El tiempo de ejecución asintótico de un algoritmo muestra cómo $T(n)$ crece a medida que n se incrementa.

Cuando $T(n)$ es un polinomio, el tiempo de ejecución asintótico es igual al término de la potencia más alta sin coeficiente (por ejemplo, n^2 en lugar de $5n^2$). A medida que n alcanza valores mayores, los demás términos pierden importancia.

- Si $T(n) = 4n + 3$, el tiempo de ejecución asintótico $T(n) \backsim n$. El algoritmo tiene una **complejidad lineal**. El símbolo de la tilde ($\backsim$) significa que el tiempo de ejecución asintótico es n.
- Si $T(n) = 5n^2 + 3n - 1$, el tiempo de ejecución asintótico $T(n) \backsim n^2$. El algoritmo tiene una **complejidad cuadrática**.
- Si $T(n) = 10n^3 - 2n^2 + 5$, entonces $T(n) \backsim n^3$. El algoritmo tiene **complejidad cúbica**.

- Si $T(n) = 10$, entonces $T(n) \backsim 1$. El algoritmo tiene **complejidad constante**, es decir, no depende de n.

Tiempo de entrenamiento de un modelo de regresión lineal

El objetivo del entrenamiento de una regresión lineal se representa de la siguiente manera:

$$\mathbf{w} = \arg\min_{\mathbf{w}} \text{MSE}(\mathbf{X}\mathbf{w}, \mathbf{y})$$

Los pesos se calculan mediante esta fórmula:

$$\mathbf{w} = (\mathbf{X}^\top \mathbf{X})^{-1} \mathbf{X}^\top \mathbf{y}$$

Determinaremos la complejidad computacional del cálculo de pesos, pero primero repasemos algunos aspectos:

- Vamos a expresar el número de observaciones en el conjunto de entrenamiento como n y el número de características como p
- El tamaño de la matriz X será n x p y el tamaño del vector y será n
- La complejidad computacional se expresará como $T(n, p)$ porque depende de dos parámetros, n y p

Para calcular la complejidad del entrenamiento del algoritmo, hay que sumar las respuestas:

$T(n, p) \backsim np^2 + p^3 + np^2 + np$

Normalmente hay menos características que observaciones, lo que significa que $p < n$. Si multiplicamos ambas partes por p^2, obtenemos $p^3 < np^2$. Tomando solo el término con la mayor potencia, obtenemos: $T(n, p) \backsim np^2$. Si hay muchas características, el entrenamiento llevará más tiempo.

Métodos iterativos

La siguiente fórmula se emplea como **método directo** para entrenar modelos de regresión lineal:

$$\mathbf{w} = (\mathbf{X}^\top \mathbf{X})^{-1} \mathbf{X}^\top \mathbf{y}$$

Los métodos directos ayudan a encontrar una solución precisa utilizando una fórmula o un algoritmo determinado. Su complejidad computacional es independiente de los datos.

Otro enfoque para entrenar modelos de regresión lineal es el uso de **métodos iterativos** o **algoritmos iterativos**. Sin embargo, estos no te darán una solución precisa, sino solo una aproximada. El algoritmo realiza iteraciones similares repetidamente y la solución se vuelve más precisa con cada paso. En el caso de que no se necesite una gran precisión, bastará con unas pocas iteraciones.

La complejidad computacional de los métodos iterativos depende del número de pasos realizados, que puede verse afectado por la cantidad de datos.

Método de bisección

Vamos a encontrar la solución de la ecuación $f(x) = 0$ mediante un método iterativo. Vamos a definir $f(x)$ como una **función continua**. Esto significa que su gráfico se puede trazar sin levantar el lápiz del papel.

El **método de bisección** nos ayudará a resolver nuestra ecuación. Este toma una función continua y el segmento $[a, b]$ como entrada. Los valores $f(a)$ y $f(b)$ tienen signos diferentes.

Cuando se cumplen estas dos condiciones:

1. la función es continua
2. los valores de los extremos del segmento tienen signos diferentes

entonces la raíz de la ecuación se encuentra en algún punto del segmento dado.

En cada iteración, el método de bisección:

- comprueba si algún valor $f(a)$ o $f(b)$ es igual a cero. Si lo es, ya tenemos la solución;
- encuentra el centro del segmento $c = (a + b)/2$;
- compara el signo de $f(c)$ con los signos de $f(a)$ and $f(b)$:
  - Si $f(c)$ y $f(a)$ tienen signos diferentes, la raíz se encuentra en el segmento $[a, c]$. El algoritmo analizará este segmento en su siguiente iteración.
  - Si $f(c)$ y $f(b)$ tienen signos diferentes, la raíz se encuentra en el segmento $[b, c]$. El algoritmo analizará este segmento en su siguiente iteración.
  - Los signos de $f(a)$ y $f(b)$ son diferentes, por lo que no hay más opciones.

La exactitud de la solución se suele elegir de antemano, por ejemplo, *e* (margen de error) *= 0.000001*. En cada iteración, el segmento con la raíz se divide entre 2. Una vez que se alcance una longitud de segmento inferior a *e*, el algoritmo podrá detenerse. Esta condición se denomina **criterio de parada**.

Comparación de métodos

Una gran parte de este curso se centra en el **descenso de gradiente**, que es el método iterativo clave para el machine learning. Muchos algoritmos de entrenamiento se basan en él porque tiene muchas ventajas en comparación con los métodos directos, por ejemplo:

- Funciona más rápido con grandes conjuntos de datos en regresiones lineales con la función de pérdida *ECM*.
- También es apropiado para regresiones lineales con otras funciones de pérdida (no todas tienen fórmulas).
- Puede utilizarse para entrenar redes neuronales que también carecen de fórmulas directas.