Machine Learning

Rodrigo Barrera

Las metaheurísticas son estrategias de alto nivel que guían el proceso de búsqueda de soluciones óptimas en problemas de optimización, incluyendo el ajuste de hiperparámetros en modelos de aprendizaje automático. Estas técnicas son especialmente útiles cuando se enfrentan a problemas complejos y de alta dimensión donde los métodos de optimización tradicionales pueden no ser efectivos o eficientes. A continuación, se describen algunas de las metaheurísticas más comunes para la optimización y el ajuste de hiperparámetros.

Algoritmos genéticos (AG)

Técnicas de búsqueda inspiradas en los principios de la evolución biológica y la selección natural. Operan mediante la generación de una población de soluciones candidatas (individuos), y luego aplican operadores genéticos como la selección, la cruzamiento (recombinación) y la mutación sobre esta población para crear nuevas generaciones de soluciones. Los AG son particularmente útiles para explorar grandes espacios de búsqueda y han sido ampliamente utilizados para el ajuste de hiperparámetros en modelos complejos.

Optimización por enjambre de partículas (PSO)

Simula el comportamiento social de enjambres, como bancos de peces o bandadas de pájaros. En PSO, cada solución candidata en el espacio de búsqueda es considerada una partícula, y el algoritmo ajusta la trayectoria de cada partícula hacia sus propias mejores posiciones conocidas y hacia las mejores posiciones encontradas por el enjambre, buscando óptimos globales. PSO se utiliza para ajustar hiperparámetros debido a su simplicidad y eficacia en problemas de alta dimensión.

Búsqueda tabú (BT)

La búsqueda tabú es una técnica que utiliza una lista tabú para almacenar los movimientos recientemente realizados (o soluciones visitadas) para evitar ciclos y fomentar la exploración de nuevas áreas en el espacio de búsqueda. La BT es efectiva para problemas de optimización combinatoria y se ha utilizado en el ajuste de hiperparámetros para evitar la reevaluación de configuraciones previamente examinadas.

Algoritmo de búsqueda tabú

Algorithm 1 Esquema básico de la búsqueda tabú

1: Establecer $t = 0$.
2: Generar una solución inicial x.
3: Inicializar las listas tabú $T \leftarrow \emptyset$ y el tamaño de la lista tabú L.
4: **repeat**
5: Establecer el conjunto de candidatos $A(x,t) = \{x' \in N(x) \setminus T(x,t) \cup \tilde{T}(x,t)\}$.
6: Encontrar el mejor x de $A(x,t)$: Establecer $x' = \arg\min_{y \in A(x,t)} f(y)$.
7: Si $f(x')$ es mejor que $f(x)$, entonces $x \leftarrow x'$.
8: Actualizar las listas tabú y los criterios de aspiración.
9: Si la lista tabú T está llena, reemplazar características antiguas de T.
10: Establecer $t = t + 1$.
11: **until** se cumpla el criterio de terminación

- El teorema del no existe tal cosa como un almuerzo gratis (No Free Lunch Theorem, NFL) es un concepto fundamental en teoría de la optimización y el aprendizaje automático que establece que ningún algoritmo de optimización o búsqueda es universalmente superior a otro cuando se considera el rendimiento promedio sobre todas las posibles funciones de costo. En otras palabras, un algoritmo que es el mejor para resolver un tipo específico de problema no necesariamente será el mejor para resolver otro tipo.
- El teorema puede ser formulado de diversas maneras, dependiendo del contexto, pero una versión general del teorema se puede expresar de la siguiente manera:

- Para cualquier algoritmo de búsqueda y optimización A, el rendimiento promedio de A sobre todas las posibles funciones objetivo f es igual al rendimiento promedio de cualquier otro algoritmo de búsqueda y optimización B sobre todas las posibles funciones objetivo.

Demostración esquemática

1. **Definición de rendimiento y espacio de funciones:**
   - Definimos el **rendimiento** de un algoritmo en términos de la calidad de la solución encontrada o el número de evaluaciones necesarias.
   - Consideramos el **espacio de todas las posibles funciones objetivo**, desde las más simples hasta las más complejas.
2. **Distribución uniforme y rendimiento promedio:**
   - Asumimos una **distribución uniforme** de funciones en este espacio, haciendo que todas las funciones sean igualmente probables.
   - El **rendimiento promedio** de un algoritmo se calcula como el promedio de su rendimiento sobre todas las posibles funciones objetivo.

Independencia del algoritmo y conclusión

3. Independencia del algoritmo:

- Dada la igual probabilidad de todas las funciones y la ausencia de información previa, el rendimiento promedio de cualquier algoritmo debe ser igual al de cualquier otro algoritmo.

4. Conclusión:

- No existe un algoritmo "universalmente óptimo". La efectividad de un algoritmo depende de las características específicas del problema que se está abordando.

Nota: Aunque el teorema se aplica en un contexto teórico idealizado, ciertos algoritmos son preferidos en la práctica para problemas específicos, gracias al conocimiento previo sobre su estructura.

- La mayoría de los algoritmos de ML implican optimización
- Minimizar/maximizar una función $f(\boldsymbol{x})$ alterando $\boldsymbol{x}$
- Generalmente expresado como minimización. La maximización se logra minimizando $-f(x)$
- $f(\boldsymbol{x})$ se llama objetivo o criterio. En la minimización también se le llama función de pérdida, costo o error
- Un ejemplo es el de mínimos cuadrados lineales $f(x) = \frac{1}{2}||Ax - b||$
- Se denota el valor óptimo como $\boldsymbol{x}^* = \text{argmin}\, f(\boldsymbol{x})$

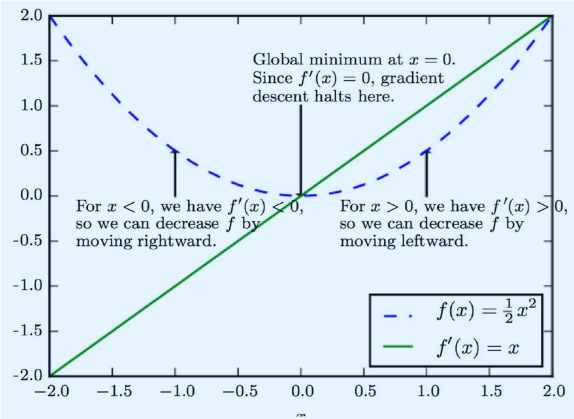
- Considera la función $y = f(x), x, y$ números reales.
- La derivada de la función se denota como $f'(x)$ o como $\frac{dy}{dx}$
- La derivada $f'(x)$ da la pendiente de $f(x)$ en el punto x

Especifica cómo escalar un pequeño cambio en la entrada para obtener un cambio correspondiente en la salida:

$$f(x + \varepsilon) \approx f(x) + \varepsilon f'(x)$$

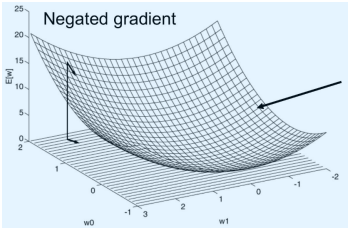
- Indica cómo hacer un pequeño cambio en la entrada para hacer una pequeña mejora en y.
- Sabemos que $f\left(x - \varepsilon \operatorname{sign}\left(f'(x)\right)\right)$ es menor que $f(x)$ para ε pequeño. Por lo tanto, podemos reducir $f(x)$ moviendo x en pequeños pasos con el signo opuesto de la derivada.
- Esta técnica se llama descenso de gradiente (Cauchy 1847)

- La función dada es $f(x) = \frac{1}{2}x^2$, que tiene una forma de cuenco con un mínimo global en $x = 0$.
- Dado que $f'(x) = x$.
- Para $x > 0$, $f(x)$ aumenta con x y $f'(x) > 0$.
- Para $x < 0$, $f(x)$ disminuye con x y $f'(x) < 0$.
- Usa $f'(x)$ para seguir la función cuesta abajo.
- Reduce $f(x)$ yendo en la dirección opuesta al signo de la derivada $f'(x)$.


Global minimum at $x = 0$.
Since $f'(x) = 0$, gradient descent halts here.
For $x < 0$, we have $f'(x) < 0$, so we can decrease f by moving rightward.
For $x > 0$, we have $f'(x) > 0$, so we can decrease f by moving leftward.
$f(x) = \frac{1}{2}x^2$
$f'(x) = x$
2.0
1.5
1.0
0.5
0.0
-0.5
-1.0
-1.5
-2.0
−2.0
−1.5
−1.0
−0.5
0.0
0.5
1.0
1.5
2.0

Aplicación en ML: Minimizar el error

Dirección en el plano $w_0 - w_1$ que produce el descenso más pronunciado
La superficie de error muestra la conveniencia de cada vector de peso, una parábola con un único mínimo global.



La búsqueda por descenso gradiente determina un vector de pesos w que minimiza $E(w)$ así:

- Partiendo de un vector de pesos inicial arbitrario
- Modificándolo repetidamente en pequeños pasos
- En cada paso, el vector de pesos se modifica en la dirección que produce el descenso más pronunciado a lo largo de la superficie de error

Definición de vector de gradiente

- El Gradiente (derivada) de E con respecto a cada componente del vector w

$$\nabla E[\vec{w}] \equiv \left[\frac{\partial E}{\partial w_0}, \frac{\partial E}{\partial w_1}, \cdots \frac{\partial E}{\partial w_n} \right]$$

- Nótese $\nabla E[\vec{w}]$ es un vector de derivadas parciales
- Especifica la dirección que produce el aumento más pronunciado en E
- El negativo de este vector especifica la dirección de la disminución más pronunciada

Derivada direccional

La derivada direccional en la dirección de $\mathbf{u}$ (un vector unitario) es la pendiente de la función f en la dirección de $\mathbf{u}$. Esto se evalúa como $\mathbf{u}^T \nabla_x f(x)$, donde $\nabla_x f(x)$ representa el gradiente de f respecto a x.

Derivada direccional

Para minimizar una función f, necesitamos identificar en qué dirección f disminuye más rápidamente desde un punto dado. Esta dirección es crucial para métodos iterativos de optimización, como el descenso de gradiente.

Expresión de minimización: la dirección de máximo descenso se halla minimizando la expresión:

$$\min_{\mathbf{u}, \|\mathbf{u}\|=1} \mathbf{u}^T \nabla f(x)$$

Restricción: considerando la restricción $\mathbf{u}^T\mathbf{u} = 1$, que indica que $\mathbf{u}$ es un vector unitario, la expresión se simplifica a:

$$\min_{\mathbf{u}} \|\nabla f(x)\| \cos\theta$$

donde θ es el ángulo entre $\mathbf{u}$ y el gradiente de f, $\nabla f(x)$.

Interpretación: el mínimo se alcanza cuando $\theta = \pi$, lo que implica que la dirección de máximo descenso es opuesta al gradiente de f.

Interpretación de la derivada direccional

La derivada direccional nos ayuda a entender cómo varía una función f en diferentes direcciones desde un punto dado. Al considerar la norma $||\mathbf{u}||_2 = 1$ para el vector direccional $\mathbf{u}$, nos centramos solo en la dirección, omitiendo la magnitud.

Simplificación: al ignorar los factores constantes, la búsqueda de la dirección óptima se reduce a:

$$\min_{\mathbf{u}} \cos\theta$$

donde θ es el ángulo entre $\mathbf{u}$ y $\nabla f(x)$.

Dirección de máximo descenso: el valor mínimo de $\cos\theta$ se alcanza cuando $\theta = \pi$, es decir, cuando $\mathbf{u}$ apunta en sentido opuesto al gradiente $\nabla f(x)$. Esto nos indica que la dirección de máximo descenso, donde f disminuye más rápidamente, es la opuesta al gradiente de f.

Aplicación en Optimización: En el contexto de la optimización, especialmente en el método de descenso de gradiente, se utiliza esta propiedad para actualizar iterativamente la solución actual moviéndose en la dirección opuesta al gradiente, es decir, hacia el mínimo local de f:

$$x_{\text{new}} = x_{\text{old}} - \alpha \nabla f(x_{\text{old}})$$

Método del descenso del gradiente

La regla del descenso de gradiente se expresa como $\vec{w} \leftarrow \vec{w} + \Delta\vec{w}$, donde $\Delta\vec{w}$ representa el cambio aplicado al vector de pesos $\vec{w}$ en cada iteración del algoritmo. Este cambio se define como $\Delta\vec{w} = -\eta\nabla E[\vec{w}]$, siendo η una constante positiva conocida como tasa de aprendizaje, que determina el tamaño del paso en la búsqueda de descenso de gradiente.

Método del descenso del gradiente

La tasa de aprendizaje es crucial porque un valor muy alto puede llevar a que el algoritmo sobrepase el mínimo, mientras que un valor muy bajo puede hacer que el algoritmo avance muy lentamente hacia el mínimo, aumentando el número de iteraciones necesarias para converger.

Método del descenso del gradiente

La forma componente del descenso de gradiente se puede escribir como $w_i \leftarrow w_i + \Delta w_i$, donde Δw_i se define como $\Delta w_i = -\eta \frac{\partial E}{\partial w_i}$. Esta forma descompone el ajuste de los pesos en sus componentes individuales, aplicando el gradiente de la función de error E respecto a cada peso w_i individualmente. De esta manera, cada peso w_i se actualiza en la dirección que minimiza la función de error, utilizando el gradiente de E con respecto a ese peso específico para determinar el ajuste necesario.

Método del descenso del gradiente

- El gradiente indica la dirección de mayor pendiente ascendente, mientras que el gradiente negativo señala hacia la pendiente descendente más pronunciada.
- Por lo tanto, podemos disminuir f moviéndonos en la dirección del gradiente negativo.
- Esto se conoce como el método del descenso más pronunciado o descenso del gradiente.
- El descenso más pronunciado propone un nuevo punto

$$\boldsymbol{x}' = \boldsymbol{x} - \eta \nabla_x f(\boldsymbol{x})$$

donde η es la tasa de aprendizaje, un escalar positivo. Se establece en una constante pequeña.

Algoritmo

- $\boldsymbol{\theta}^1$ // Punto de inicio inicial
- f // Función a minimizar
- δ // Umbral de convergencia
- $t \leftarrow 1$
- Repetir
  - $\boldsymbol{\theta}^{t+1} \leftarrow \boldsymbol{\theta}^t - \eta \nabla f(\boldsymbol{\theta}^t)$
  - $t \leftarrow t + 1$
- Mientras $\|\boldsymbol{\theta}^t - \boldsymbol{\theta}^{t-1}\| > \delta$
- Devolver $(\boldsymbol{\theta}^t)$

Algoritmo

La expansión de Taylor de la función $f(\theta)$ en el vecindario de $\boldsymbol{\theta}^t$ es
$f(\boldsymbol{\theta}) \approx f(\boldsymbol{\theta}^t) + (\boldsymbol{\theta} - \boldsymbol{\theta}^t)^T \nabla f(\boldsymbol{\theta}^t)$.
Sea $\theta = \theta^{t+1} = \theta^t + h$, entonces $f(\boldsymbol{\theta}^{t+1}) \approx f(\boldsymbol{\theta}^t) + h\nabla f(\boldsymbol{\theta}^t)$.
La derivada de $f(\theta^{t+1})$ respecto a h es $\nabla f(\boldsymbol{\theta}^t)$.
En $h = \nabla f(\boldsymbol{\theta}^t)$ ocurre un máximo (ya que h^2 es positivo) y en $h = -\nabla f(\boldsymbol{\theta}^t)$ ocurre un mínimo.

Algoritmo

Alternativamente,
La pendiente $\nabla f(\theta^t)$ indica la dirección de ascenso más pronunciado. Si damos un paso η en la dirección opuesta, disminuimos el valor de f.
Ejemplo unidimensional: Sea $f(\theta) = \theta^2$.
Esta función tiene un mínimo en $\theta = 0$ que queremos determinar usando el descenso de gradiente.
Tenemos $f'(\theta) = 2\theta$.
Para el descenso de gradiente, actualizamos por $-f'(\theta)$.
Si $\theta^t > 0$ entonces $\theta^{t+1} < \theta^t$.
Si $\theta^t < 0$ entonces $f'(\theta^t) = 2\theta^t$ es negativo, así $\theta^{t+1} > \theta^t$.

Algoritmo

En el contexto del descenso de gradiente aplicadoa la regresión de mínimos cuadrados, el criterio a minimizar es la función $f(x) = \frac{1}{2}\|Ax - b\|^2$, donde A es una matriz de diseño y b es el vector de términos constantes. La regresión de mínimos cuadrados busca minimizar $E_0(w) = \frac{1}{2}\sum_{i=1}^{N}(t_i - w^T\phi(x_i))^2$, donde $\phi(x_i)$ representa las características o transformaciones de los datos de entrada x_i, y t_i son los valores objetivo.

El gradiente de $f(x)$ con respecto a $\boldsymbol{x}$ es $\nabla_{\boldsymbol{x}} f(\boldsymbol{x}) = A^T(A\boldsymbol{x} - b) = A^T A\boldsymbol{x} - A^T b$, que proporciona la dirección en la que la función f aumenta más rápidamente. Para minimizar f, el algoritmo de descenso de gradiente actualiza $\boldsymbol{x}$ en la dirección opuesta al gradiente.

El algoritmo de descenso de gradiente para este caso se puede describir de la siguiente manera:
1. Se establece un tamaño de paso η y una tolerancia δ, ambos valores pequeños y positivos.
2. Mientras la norma $\left\|A^T A\boldsymbol{x} - A^T \boldsymbol{b}\right\|_2$ sea mayor que δ, se realiza lo siguiente: - Se actualiza $\boldsymbol{x}$ según la regla $\boldsymbol{x} \leftarrow \boldsymbol{x} - \eta(A^T A\boldsymbol{x} - A^T \boldsymbol{b})$.

Este proceso se repite hasta que el gradiente sea lo suficientemente pequeño, es decir, menor que δ, lo cual indica que $\boldsymbol{x}$ está cerca de un mínimo de la función de coste f. La elección de η es crucial; si es demasiado grande, el algoritmo puede oscilar o diverger, mientras que si es demasiado pequeño, la convergencia puede ser muy lenta.

El **Descenso de Gradiente Estocástico** (DGE) es un método de optimización utilizado para minimizar una función objetivo, particularmente en el contexto de aprendizaje automático y deep learning. A diferencia del descenso de gradiente convencional, que utiliza todo el conjunto de datos para calcular el gradiente de la función de coste, el DGE actualiza los parámetros del modelo de manera iterativa utilizando solo un subconjunto aleatorio de los datos, típicamente un solo ejemplo, en cada paso. Esto lo hace especialmente útil para conjuntos de datos grandes debido a su eficiencia computacional.
La actualización de los parámetros del modelo en el DGE se realiza según la siguiente fórmula:

$$\theta = \theta - \eta \nabla Q_i(\theta) \tag{1}$$

donde θ representa los parámetros del modelo, η es la tasa de aprendizaje y $\nabla Q_i(\theta)$ es el gradiente de la función de coste calculado con respecto a un subconjunto aleatorio i de los datos.

El DGE introduce ruido en el proceso de optimización, lo que puede ayudar a evitar mínimos locales y promover la convergencia hacia mínimos globales en funciones de coste complejas y no convexas. Sin embargo, este ruido también puede hacer que la convergencia sea más irregular en comparación con el descenso de gradiente tradicional.
Tanto la estimación estadística como el aprendizaje automático consideran el problema de minimizar una función objetivo que tiene la forma de una suma:

$$Q(w) = \frac{1}{n} \sum_{i=1}^{n} Q_i(w), \tag{2}$$

donde el parámetro w que minimiza $Q(w)$ debe ser estimado. Cada función sumando Q_i está típicamente asociada con la i-ésima observación en el conjunto de datos (utilizado para el entrenamiento).

En estadísticas clásicas, los problemas de minimización de suma surgen en los métodos de mínimos cuadrados y en la estimación de máxima verosimilitud (para observaciones independientes). La clase general de estimadores que surgen como minimizadores de sumas se llaman M-estimadores. Sin embargo, en estadística, se ha reconocido desde hace tiempo que exigir incluso la minimización local es demasiado restrictivo para algunos problemas de estimación de máxima verosimilitud. Por lo tanto, los teóricos estadísticos contemporáneos a menudo consideran puntos estacionarios de la función de verosimilitud (o ceros de su derivada, la función de puntuación, y otras ecuaciones de estimación).

Mínimos cuadrados

La función de error $E_D(\mathbf{w}) = \frac{1}{2}\sum_{n=1}^{N}\{t_n - \mathbf{w}^T\phi(\mathbf{x}_n)\}^2$ suma sobre los datos, denotando $E_D(\mathbf{w}) = \Sigma_n E_n$, se actualiza el vector de parámetros $\mathbf{w}$ utilizando la siguiente regla de actualización:

$$\mathbf{w}^{(\tau+1)} = \mathbf{w}^{(\tau)} - \eta\nabla E_n$$

Sustituyendo la derivada en la regla de actualización, obtenemos:

$$\mathbf{w}^{(\tau+1)} = \mathbf{w}^{(\tau)} + \eta(t_n - \mathbf{w}^{(\tau)T}\phi(\mathbf{x}_n))\phi(\mathbf{x}_n)$$

Aquí, $\mathbf{w}$ se inicializa a algún vector de inicio $\mathbf{w}^{(0)}$, y η se elige con cuidado para asegurar la convergencia del algoritmo.