Algoritmos básicos de optimización

Descenso del Gradiente

El Descenso del Gradiente es un algoritmo fundamental en el campo del aprendizaje automático, utilizado para minimizar una función de costo y, por ende, mejorar el rendimiento de un modelo. El algoritmo tiene los siguientes pasos:

1. Inicialización de pesos:

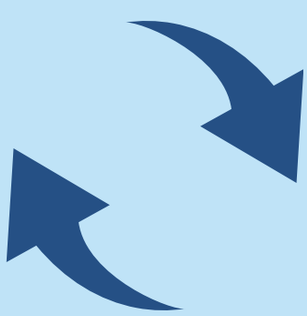
Comienza con la inicialización de los pesos del modelo de manera aleatoria o mediante algún otro método.

2. Evaluación del gradiente:

Calcula el gradiente de la función de costo con respecto a los pesos actuales. El gradiente indica la dirección y la tasa de cambio más pronunciada de la función en el punto actual.

3. Actualización de pesos:

Ajusta los pesos en la dirección opuesta al gradiente para minimizar la función de costo. Esto se realiza multiplicando el gradiente por una tasa de aprendizaje y restando el resultado de los pesos actuales.



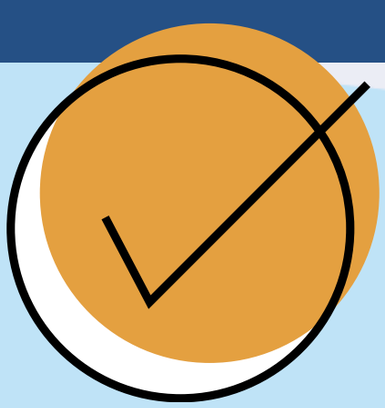
4. Iteración:

Repita los pasos 2 y 3 hasta que se cumpla algún criterio de parada, como un número fijo de iteraciones o una convergencia aceptable.

Descenso del Gradiente

El propósito del Descenso del Gradiente, es encontrar el mínimo global o local de una función de costo al seguir la pendiente descendente. La tasa de aprendizaje controla el tamaño de los pasos que se dan en cada iteración; una tasa de aprendizaje demasiado grande puede hacer que el algoritmo diverja, mientras que una tasa demasiado pequeña puede llevar a convergencia lenta.

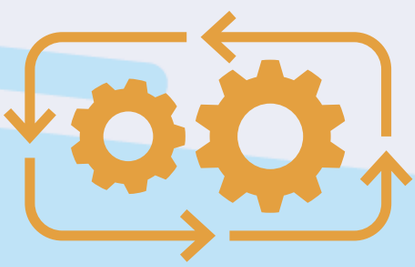
El Descenso del Gradiente es un proceso iterativo que ajusta los pesos del modelo para minimizar la función de costo, utilizando la información proporcionada por el gradiente de la función. Este algoritmo es esencial en la optimización de modelos en el aprendizaje automático y forma la base de muchos otros algoritmos más avanzados.



Descenso del Gradiente Estocástico

Es una variante del Descenso del Gradiente que difiere en la forma en que se actualizan los pesos del modelo. A continuación, se presenta una explicación del algoritmo y sus diferencias con respecto al Descenso del Gradiente clásico.

1. Inicialización de pesos:



Al igual que en el Descenso del Gradiente, comienza con la inicialización de los pesos del modelo.

2. Iteración por muestras

A diferencia del Descenso del Gradiente, en cada iteración, en lugar de calcular el gradiente utilizando todo el conjunto de datos de entrenamiento, SGD utiliza una sola muestra de datos aleatoria para calcular el gradiente. Esto introduce un componente estocástico en el proceso.

3. Actualización de pesos:

Los pesos se ajustan después de cada muestra calculando el gradiente basado únicamente en esa muestra y aplicando la actualización. Este proceso se repite para todas las muestras en el conjunto de datos de entrenamiento.

4. Reordenamiento de datos:

Después de pasar por todo el conjunto de datos una vez, generalmente se realiza un reordenamiento aleatorio de las muestras antes de comenzar la siguiente época de iteraciones. Esto ayuda a que el modelo no se ajuste demasiado a patrones específicos de las muestras en un orden particular.

Diferencias claves con respecto al Descenso del Gradiente clásico

1

Menos requisitos computacionales

Al utilizar solo una muestra a la vez, SGD es computacionalmente menos costoso en cada iteración, lo que permite entrenar modelos más grandes en conjuntos de datos más grandes.

Mayor variabilidad

Debido a la selección estocástica de muestras, la variabilidad en las actualizaciones de peso es mayor, esto puede ayudar a evitar mínimos locales y, en algunos casos, contribuir a un mejor rendimiento en generalización.

2

3

Menor suavización

La variabilidad, sin embargo, puede hacer que la convergencia no sea tan suave como en el Descenso del Gradiente, y a veces la función de costo puede mostrar más fluctuaciones.

El Descenso del Gradiente Estocástico es una versión más eficiente del Descenso del Gradiente, especialmente útil cuando se trabaja con grandes conjuntos de datos. Aunque puede ser más ruidoso, a menudo converge más rápidamente.

4

Exploración de cómo estos algoritmos se aplican en el contexto de modelos profundos

La aplicación de los algoritmos de Descenso de Gradiente y Descenso de Gradiente Estocástico (SGD) en redes neuronales sigue un proceso general de optimización durante el entrenamiento.

Descenso de Gradiente (Batch Gradient Descent)

1. Inicialización de Pesos

Comienza con la inicialización de los pesos de la red neuronal.

2. Paso hacia Adelante (Forward Pass)

Pase el conjunto completo de datos de entrenamiento a través de la red para calcular las predicciones y la función de pérdida.

3. Cálculo del Gradiente

Calcula el gradiente de la función de pérdida con respecto a los pesos utilizando el conjunto completo de datos de entrenamiento. Este paso implica retropropagación (backpropagation) para calcular los gradientes.

4. Actualización de Pesos

Utiliza el gradiente calculado para actualizar los pesos de acuerdo con la fórmula de actualización del Descenso de Gradiente. La fórmula general es: peso_nuevo = peso_viejo - tasa_aprendizaje * gradiente.

5. Iteración

Repita los pasos 2-4 hasta que la función de pérdida converja o hasta alcanzar un número predefinido de épocas.

Exploración de cómo estos algoritmos se aplican en el contexto de modelos profundos

Descenso de Gradiente Estocástico (SGD)

1. Inicialización de Pesos

al igual que en Batch Gradient Descent, comience con la inicialización de los pesos de la red neuronal.

2. Reordenamiento de Datos

Antes de cada época, se suele realizar un reordenamiento aleatorio de los datos de entrenamiento.

3. Iteración por Muestras

Para cada muestra se realiza los siguientes pasos:

• Paso hacia Adelante (Forward Pass): Pase la muestra a través de la red para calcular las predicciones y la función de pérdida.
• Cálculo del Gradiente: calcule el gradiente de la función de pérdida con respecto a los pesos utilizando solo esa muestra.

4. Iteración Completa

Repita el paso 3 para todas las muestras en el conjunto de datos antes de comenzar una nueva época.

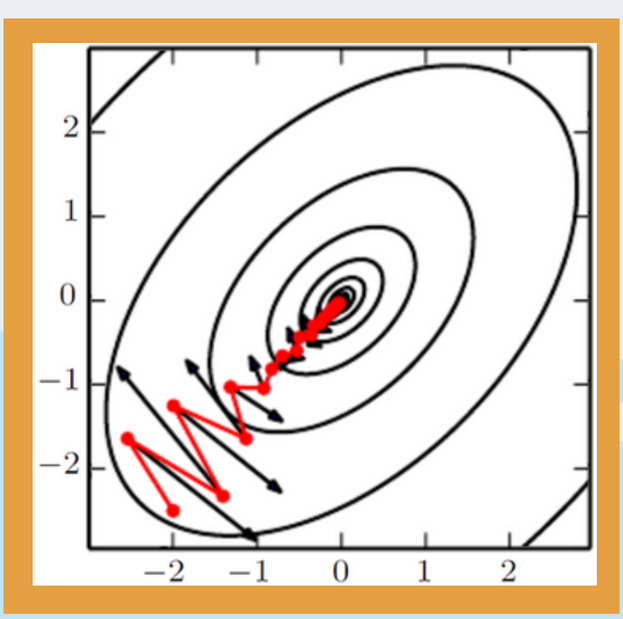
5. Iteración Global

Repita los pasos 2-4 para un número predefinido de épocas.

Los dos algoritmos anteriores se utilizan para minimizar la función de pérdida y ajustar los pesos de la red para mejorar las predicciones. La elección entre Gradient Descent y SGD a menudo depende de factores como el tamaño del conjunto de datos y la eficiencia computacional. El Descenso de Gradiente Estocástico es especialmente útil cuando se trabaja con grandes conjuntos de datos, ya que reduce los requisitos computacionales.

Método de Momentum en el descenso de gradiente

El método de Momentum, es una técnica utilizada en el Descenso de Gradiente para mejorar la convergencia, especialmente en presencia de gradientes ruidosos o superficies de pérdida irregulares. En lugar de confiar únicamente en el gradiente instantáneo en cada paso, el método de Momentum incorpora una especie de "inercia" o "momentum" basado en el historial de gradientes anteriores.



Veamos, a continuación, el paso a paso de esta técnica utilizada en el Descenso de Gradiente para mejorar la convergencia.

Proceso del Método de Momentum

1. Cálculo del Gradiente

En cada iteración, se calcula el gradiente de la función de pérdida con respecto a los parámetros del modelo.

2. Actualización de la velocidad (Momentum)

Se actualiza una variable llamada "velocidad" que representa la dirección y la magnitud acumulativa de los gradientes anteriores.

La actualización se realiza multiplicando la velocidad anterior por un factor de amortiguación (generalmente denotado como beta) y sumando el gradiente actual multiplicado por la tasa de aprendizaje (alfa).

3. Actualización de parámetros

Los parámetros del modelo se actualizan utilizando la velocidad calculada, la velocidad acumulativa actúa como un impulso que ayuda a superar pequeños baches o ruido en la superficie de pérdida.

Ventajas del método de Momentum

- Ayuda a acelerar la convergencia en direcciones consistentes del gradiente.
- Mitiga oscilaciones no deseadas, especialmente en superficies de pérdida con pequeñas curvas o ruido.
- El método de Momentum es especialmente útil en el entrenamiento de redes neuronales y otras tareas de optimización donde la superficie de pérdida puede ser compleja.