**1. Regresión Lineal**

* La **ecuación normal** y la descomposición en valores singulares (SVD) son métodos comunes para ajustar modelos de regresión lineal. Sin embargo, se vuelven poco eficientes cuando hay un gran número de características.
* **El costo computacional** aumenta a medida que crece el número de características, lo que hace que estos métodos no sean prácticos para datasets muy grandes.

**2. Descenso de Gradiente**

* **El descenso de gradiente** es un algoritmo genérico de optimización que ajusta los parámetros iterativamente para minimizar una función de costo.
* Un parámetro clave es la **tasa de aprendizaje**, que controla el tamaño de los pasos hacia la solución óptima. Si es demasiado pequeña, el algoritmo será lento. Si es demasiado alta, podría saltarse el mínimo y no converger.
* Existen diferentes variantes del descenso de gradiente:
  + **Batch Gradient Descent (descenso de gradiente por lotes)**: Utiliza todo el conjunto de entrenamiento en cada paso. Es más lento, pero puede converger de manera segura cuando se usa adecuadamente.
  + **Stochastic Gradient Descent (descenso de gradiente estocástico)**: Actualiza los parámetros para cada instancia de manera individual, lo que lo hace más rápido y adecuado para datasets grandes. Sin embargo, sufre de alta variabilidad y puede no llegar a un mínimo exacto.
  + **Mini-batch Gradient Descent (descenso de gradiente en mini lotes)**: Es una combinación de los dos anteriores, actualizando los parámetros usando pequeños subconjuntos del conjunto de entrenamiento. Es más eficiente y estable que el estocástico, pero no tan preciso como el batch.

Lo ideal sería es esperar lo suficiente y no tener un learning rate muy alto para que el gradiente se aproxime mucho al global mínimo.

Al utilizar Descenso de degradado, debe asegurarse de que todas las entidades tengan una escala similar (por ejemplo, usando la clase StandardScaler de Scikit-Learn), o de lo contrario llevará mucho más tiempo converger.

**3. Regularización**

* Se presentan varias técnicas para mejorar el rendimiento del modelo evitando el sobreajuste:
  + **Ridge Regression**: Añade un término de regularización que penaliza los coeficientes elevados.
  + **Lasso Regression**: Similar a Ridge, pero puede forzar algunos coeficientes a ser exactamente cero, eliminando características no útiles.
  + **Elastic Net**: Una combinación de Ridge y Lasso, útil cuando se sospecha que solo algunas características son importantes.

**4. Detención Temprana (Early Stopping)**

* Es una técnica de regularización que detiene el entrenamiento cuando el error de validación comienza a aumentar, lo que indica que el modelo está comenzando a sobreajustarse a los datos de entrenamiento.

**5. Desafíos en el Descenso de Gradiente**

* Los modelos pueden tener terrenos de optimización complicados, con múltiples mínimos locales y mesetas. Es importante ajustar correctamente la tasa de aprendizaje y utilizar técnicas como el escalado de características para acelerar la convergencia.

**Puntos clave para el examen:**

1. **Conceptos básicos de los diferentes tipos de descenso de gradiente** (batch, estocástico, mini-lote) y cuándo usarlos.
2. **Impacto de la tasa de aprendizaje** en la convergencia del algoritmo.
3. **Regularización** (Ridge, Lasso, Elastic Net) y cómo ayudan a evitar el sobreajuste.
4. **Detención temprana** como una técnica efectiva para regularizar y evitar el sobreajuste en modelos iterativos.
5. **Complejidad computacional** en la regresión lineal y el manejo de conjuntos de datos grandes.

**1. Tipos de Descenso de Gradiente**

El descenso de gradiente es un algoritmo que se utiliza para minimizar funciones de costo en una amplia variedad de modelos. Aquí veremos los tres tipos principales y cuándo deberías usarlos.

* **Batch Gradient Descent (Descenso de Gradiente por Lotes)**:
  + **Descripción**: Este algoritmo calcula el gradiente utilizando todo el conjunto de datos en cada iteración. Esto garantiza que cada actualización sea precisa, pero también es muy costoso computacionalmente.
  + **Ventajas**:
    - Convergencia estable.
    - Ideal para conjuntos de datos pequeños, ya que puede calcularse con rapidez.
  + **Desventajas**:
    - Si tienes un conjunto de datos muy grande, calcular el gradiente en cada paso puede ser extremadamente lento.
    - No se puede utilizar para datasets demasiado grandes que no caben en memoria.
  + **Cuándo usarlo**: Cuando tienes un conjunto de datos de tamaño manejable y necesitas precisión en cada paso del algoritmo.
* **Stochastic Gradient Descent (Descenso de Gradiente Estocástico)**:
  + **Descripción**: A diferencia del batch, este algoritmo calcula el gradiente utilizando solo una instancia del conjunto de datos en cada iteración. Esto lo hace mucho más rápido, pero también introduce variabilidad (ruido) en el proceso de optimización.
  + **Ventajas**:
    - Mucho más rápido que el batch, ya que procesa una instancia a la vez.
    - Puede manejar conjuntos de datos muy grandes y puede usarse con datos "streaming".
  + **Desventajas**:
    - Es menos preciso y los parámetros del modelo fluctúan constantemente, lo que significa que no alcanzará exactamente el mínimo.
    - Puede saltarse el mínimo global debido a su naturaleza estocástica.
  + **Cuándo usarlo**: Cuando trabajas con conjuntos de datos muy grandes o en situaciones donde necesitas actualizaciones rápidas, pero puedes tolerar algo de ruido en el ajuste.
* **Mini-batch Gradient Descent (Descenso de Gradiente en Mini Lotes)**:
  + **Descripción**: Es una mezcla entre los dos anteriores. Calcula el gradiente usando pequeños subconjuntos (mini-lotes) de datos en cada iteración. Esto hace que sea más rápido que el batch y más estable que el estocástico.
  + **Ventajas**:
    - Es más rápido que el batch y más estable que el estocástico.
    - Se puede optimizar mejor en hardware como GPUs, donde se procesan operaciones matriciales.
  + **Desventajas**:
    - Aunque es más eficiente, no alcanza el mínimo con la misma precisión que el batch.
  + **Cuándo usarlo**: Cuando quieres un compromiso entre la estabilidad del batch y la velocidad del estocástico. Es la opción preferida en muchos casos prácticos.

**2. Impacto de la Tasa de Aprendizaje en la Convergencia**

La tasa de aprendizaje (η\etaη) es un hiperparámetro crucial en los algoritmos de descenso de gradiente. Controla el tamaño de los pasos que el algoritmo da hacia el mínimo de la función de costo.

* **Si la tasa de aprendizaje es demasiado pequeña**:
  + **Qué sucede**: El algoritmo toma pasos muy pequeños, por lo que tardará mucho en converger.
  + **Problemas**: Podrías necesitar miles de iteraciones para acercarte al mínimo.
  + **Ejemplo visual**: Imagina que estás bajando una colina, pero tus pasos son tan pequeños que tardas mucho en llegar abajo.
* **Si la tasa de aprendizaje es demasiado grande**:
  + **Qué sucede**: Los pasos que toma el algoritmo son demasiado grandes y puede "saltar" el mínimo, oscilando alrededor de él sin converger.
  + **Problemas**: En los peores casos, el algoritmo no converge y los valores de los parámetros pueden divergir, haciendo que no encuentres una buena solución.
  + **Ejemplo visual**: Es como si bajaras una colina con saltos enormes, a veces subiendo más que bajando.
* **Solución**:
  + Ajustar la tasa de aprendizaje es un proceso delicado. Si no sabes cuál usar, puedes realizar una **búsqueda en cuadrícula (grid search)** o utilizar técnicas de **tasa de aprendizaje adaptable**, como en **descenso de gradiente estocástico**.

**3. Regularización: Ridge, Lasso, Elastic Net**

La regularización es una técnica que ayuda a evitar el sobreajuste, especialmente cuando tienes muchas características irrelevantes o redundantes en tu modelo.

* **Ridge Regression (Regresión Ridge)**:
  + **Descripción**: Añade una penalización al tamaño de los coeficientes del modelo (α∑i=1nθi2\alpha \sum\_{i=1}^n \theta\_i^2α∑i=1n​θi2​). Esta penalización hace que los coeficientes más grandes se reduzcan.
  + **Ventaja**: Reduce el sobreajuste, especialmente cuando tienes muchas características.
  + **Uso típico**: Se usa cuando se espera que muchas características tengan algún efecto, pero no quieres que dominen el modelo.
* **Lasso Regression**:
  + **Descripción**: También añade una penalización, pero en este caso, la penalización es sobre la magnitud absoluta de los coeficientes (α∑i=1n∣θi∣\alpha \sum\_{i=1}^n |\theta\_i|α∑i=1n​∣θi​∣).
  + **Ventaja**: Puede forzar algunos coeficientes a ser exactamente cero, eliminando características irrelevantes.
  + **Uso típico**: Es útil cuando se espera que solo unas pocas características sean realmente importantes. Lasso puede eliminar automáticamente las características irrelevantes.
* **Elastic Net**:
  + **Descripción**: Combina Ridge y Lasso. Puedes controlar la proporción de la penalización Ridge y Lasso mediante un parámetro rrr (donde r=0r = 0r=0 es Ridge y r=1r = 1r=1 es Lasso).
  + **Ventaja**: Es útil cuando no estás seguro de cuántas características son relevantes y quieres un balance entre Ridge y Lasso.
  + **Uso típico**: Se recomienda cuando hay muchas características y cuando se sospecha que algunas de ellas son muy correlacionadas.

**4. Detención Temprana (Early Stopping)**

Esta técnica consiste en detener el entrenamiento de un modelo cuando el error de validación comienza a aumentar. Esto se debe a que el modelo está comenzando a **sobreajustarse** a los datos de entrenamiento.

* **Cómo funciona**:
  + Durante el entrenamiento, se monitoriza el error de validación (que es diferente del error de entrenamiento).
  + Si este error comienza a aumentar mientras que el error de entrenamiento sigue disminuyendo, es una señal de que el modelo ha aprendido demasiado de los datos de entrenamiento y está perdiendo capacidad de generalización.
  + El algoritmo detiene el entrenamiento en ese punto y devuelve los parámetros del modelo correspondientes al mínimo error de validación.
* **Ventajas**:
  + Es una forma eficiente de regularización, especialmente en métodos iterativos como el descenso de gradiente.
  + Ayuda a prevenir el sobreajuste sin necesidad de ajustar manualmente la cantidad de iteraciones.
* **Cuándo usarlo**: Es útil cuando se entrenan modelos iterativos como las redes neuronales o cualquier método que utilice el descenso de gradiente.

**5. Desafíos en el Descenso de Gradiente**

El descenso de gradiente puede enfrentar ciertos desafíos debido a la forma de la función de costo.

* **Mínimos locales**:
  + Algunos modelos tienen funciones de costo complicadas con múltiples mínimos locales. El descenso de gradiente puede quedarse atrapado en uno de estos mínimos, especialmente en modelos no lineales.
  + **Solución**: El uso de **descenso de gradiente estocástico** o una **tasa de aprendizaje adaptable** puede ayudar a salir de los mínimos locales, ya que la aleatoriedad del proceso ayuda a saltar de estos puntos.
* **Plateaus y mesetas**:
  + Algunas funciones de costo tienen áreas donde el gradiente es muy pequeño, lo que hace que el descenso de gradiente sea muy lento.
  + **Solución**: Aumentar la tasa de aprendizaje puede acelerar el proceso en estas áreas, pero hay que tener cuidado de no hacerla demasiado alta.
* **Escalado de características**:
  + Si las características no están escaladas (es decir, tienen rangos muy diferentes), el descenso de gradiente puede ser extremadamente lento porque la función de costo tiene una forma alargada.
  + **Solución**: Usar técnicas de escalado de características como el **StandardScaler** de Scikit-Learn.

|  |  |  |  |
| --- | --- | --- | --- |
| **Método** | **Ventajas** | **Desventajas** | **Cuándo Usar** |
| Batch Gradient Descent | Convergencia estable, buena precisión en cada paso | Lento con conjuntos de datos grandes, no escalable | Conjuntos de datos pequeños, cuando la precisión es crucial |
| Stochastic Gradient Descent | Más rápido, útil para grandes conjuntos de datos | Alta variabilidad, no alcanza el mínimo exacto | Conjuntos de datos muy grandes, cuando se necesita rapidez |
| Mini-batch Gradient Descent | Compromiso entre velocidad y estabilidad | Menos preciso que Batch GD | Para mejorar la estabilidad y rapidez con grandes volúmenes de datos |
| Ridge Regression | Reduce sobreajuste, mantiene todas las características | No selecciona características irrelevantes | Cuando todas las características son relevantes |
| Lasso Regression | Elimina características irrelevantes, útil en conjuntos de datos dispersos | Puede ser inestable si las características son muy correlacionadas | Cuando solo unas pocas características son importantes |
| Elastic Net | Combina las ventajas de Ridge y Lasso, balancea entre regularización | Mayor complejidad y tiempo de ajuste de hiperparámetros | Cuando no se sabe cuántas características son relevantes |
| Early Stopping | Previene sobreajuste, eficiente para regularización en modelos iterativos | Difícil de determinar cuándo detenerse en algunos casos | Para evitar sobreajuste en métodos iterativos |

Tabla

Descripción generada automáticamente