Capítulo 2: Optimización Estocástica: Algoritmos y sus Aplicaciones

En este capítulo se describen los algoritmos estocásticos utilizados en la optimización de modelos de ciencia de datos, particularmente en problemas de gran escala.

2.1 Descenso de Gradiente Estocástico (SGD)

El Descenso de Gradiente Estocástico (SGD, por sus siglas en inglés) es uno de los métodos más populares en la optimización de modelos de aprendizaje automático y ciencia de datos. Se trata de un método iterativo para encontrar el mínimo de una función de costo (o función de pérdida), que se utiliza comúnmente para entrenar redes neuronales y otros modelos de predicción (Robbins & Monro, 1951; Bottou, 2010).

Fundamentos del Descenso de Gradiente Estocástico

El SGD es una variante del algoritmo de descenso de gradiente clásico, que actualiza los parámetros del modelo utilizando solo un subconjunto aleatorio (o "minilote") de los datos en cada iteración, en lugar de utilizar todo el conjunto de datos. Esto hace que el proceso de optimización sea más rápido y adecuado para trabajar con grandes volúmenes de datos (Kingma & Ba, 2015; Bottou, 2010).

La actualización de los parámetros del modelo en el SGD se realiza mediante la siguiente fórmula:

$$\theta_{t+1} = \theta_t - \eta \nabla_\theta L(\theta_t; x_i, y_i)$$

Donde: - θ_t representa los parámetros del modelo en la iteración t, - η es la tasa de aprendizaje (un parámetro que controla el tamaño de los pasos), - $\nabla_\theta L(\theta_t; x_i, y_i)$ es el gradiente de la función de pérdida L con respecto a los parámetros θ, evaluado en un solo punto de los datos (x_i, y_i) (Kingma & Ba, 2015).

La ventaja del SGD es que, en lugar de calcular el gradiente para todo el conjunto de datos, lo hace solo para un ejemplo de entrenamiento, lo que acelera considerablemente el proceso.

Convergencia de SGD

El algoritmo de SGD tiene la ventaja de converger más rápido que el descenso de gradiente clásico, especialmente cuando se manejan grandes volúmenes de datos. Sin embargo, debido a la aleatoriedad del proceso, puede presentar oscilaciones durante el proceso de convergencia. Para mitigar esto, se utilizan técnicas de decaimiento de la tasa de aprendizaje, como la reducción exponencial de η a medida que el número de iteraciones aumenta (Robbins & Monro, 1951).

Variantes del Descenso de Gradiente Estocástico

SGD con momento

En este enfoque, se agrega un término de "momento" que ayuda a reducir las oscilaciones y acelera la convergencia. La actualización de los parámetros con momento se realiza de la siguiente forma:

$$v_{t+1} = \beta v_t + (1 - \beta)\nabla_\theta L(\theta_t; x_i, y_i)$$
$$\theta_{t+1} = \theta_t - \eta v_{t+1}$$

Donde v_t es el momento acumulado y β es un parámetro que controla la influencia del momento en la actualización (Kingma & Ba, 2015).

SGD con mini-lotes

Esta variante divide los datos en pequeños subconjuntos (mini-lotes) y calcula el gradiente para cada mini-lote, en lugar de usar un solo ejemplo de entrenamiento a la vez. Esto puede mejorar la eficiencia del proceso de optimización (Bottou, 2010).

Ejemplo de Aplicación

Imaginemos que estamos entrenando un modelo de regresión lineal utilizando SGD. Supongamos que la función de pérdida es el error cuadrático medio (MSE), que se define como:

$$L(\theta) = \frac{1}{n}\sum_{i=1}^{n}(y_i - \theta_0 - \theta_1 x_i)^2$$

Donde n es el número de muestras, x_i y y_i son las características y etiquetas de los datos de entrenamiento, y θ_0, θ_1 son los parámetros del modelo (intercepto y pendiente).

Usando el SGD, la actualización de los parámetros en cada iteración sería:

$$\theta_0^{t+1} = \theta_0^t - \eta \cdot \frac{\partial L}{\partial \theta_0}$$

$$\theta_1^{t+1} = \theta_1^t - \eta \cdot \frac{\partial L}{\partial \theta_1}$$

El gradiente para θ_0 y θ_1 se calcula a partir de un solo ejemplo de los datos de entrenamiento:

$$\frac{\partial L}{\partial \theta_0} = -2(y_i - \theta_0 - \theta_1 x_i)$$

$$\frac{\partial L}{\partial \theta_1} = -2(y_i - \theta_0 - \theta_1 x_i)x_i$$

Este proceso se repite iterativamente hasta que los parámetros converjan a los valores óptimos.

2.2 Métodos Adaptativos (Adam, RMSProp, Adagrad)

Los métodos adaptativos son una clase de algoritmos de optimización que ajustan dinámicamente la tasa de aprendizaje durante el proceso de entrenamiento. Estos métodos buscan mejorar la eficiencia de la optimización ajustando la tasa de aprendizaje de forma individual para cada parámetro, lo que permite una mayor estabilidad y rendimiento en modelos complejos y grandes volúmenes de datos [**?**].

Concepto y Aplicaciones

Uno de los principales problemas del descenso de gradiente estándar es la necesidad de una tasa de aprendizaje constante y predefinida, que puede no ser ideal para todos los parámetros y puede variar a lo largo de las iteraciones. Los métodos adaptativos solucionan esto ajustando la tasa de aprendizaje de manera automática. Entre los métodos más conocidos se encuentran **Adam (Adaptive Moment Estimation)**, **RMSProp (Root Mean Square Propagation)** y **Adagrad (Adaptive Gradient Algorithm)**, que ajustan las tasas de aprendizaje basadas en las estimaciones de los momentos del gradiente.

Adam (Adaptive Moment Estimation)

Adam es uno de los métodos adaptativos más populares debido a su eficacia y capacidad para manejar una variedad de tareas. Adam calcula las tasas de aprendizaje de manera adaptativa, utilizando tanto el primer momento (media) como el segundo momento (varianza) de los gradientes. Las actualizaciones de los parámetros se realizan según las siguientes fórmulas:

$$m_t = \beta_1 m_{t-1} + (1 - \beta_1)\nabla_\theta L(\theta_t) \tag{1}$$

$$v_t = \beta_2 v_{t-1} + (1 - \beta_2)\nabla_\theta L(\theta_t)^2 \tag{2}$$

$$\hat{m}_t = \frac{m_t}{1 - \beta_1^t}, \quad \hat{v}_t = \frac{v_t}{1 - \beta_2^t} \tag{3}$$

$$\theta_{t+1} = \theta_t - \eta \frac{\hat{m}_t}{\sqrt{\hat{v}_t} + \epsilon} \tag{4}$$

Donde β_1 y β_2 son factores de decaimiento, η es la tasa de aprendizaje, y ϵ es un pequeño valor para evitar la división por cero. Este algoritmo es especialmente eficiente para problemas con grandes volúmenes de datos y parámetros [**?**].

RMSProp (Root Mean Square Propagation)

RMSProp es otro algoritmo adaptativo que ajusta la tasa de aprendizaje utilizando una media móvil de los cuadrados del gradiente. La fórmula de actualización es:

$$v_t = \beta v_{t-1} + (1 - \beta)\nabla_\theta L(\theta_t)^2 \tag{5}$$

$$\theta_{t+1} = \theta_t - \frac{\eta}{\sqrt{v_t + \epsilon}} \nabla_\theta L(\theta_t) \tag{6}$$

En este caso, v_t es la estimación de la varianza de los gradientes, β es el factor de decaimiento, y ϵ es un valor pequeño para evitar divisiones por cero. RMSProp es muy útil en problemas no estacionarios y es particularmente eficiente para redes neuronales profundas y secuenciales [**?**].

Adagrad (Adaptive Gradient Algorithm)

Adagrad ajusta la tasa de aprendizaje de manera que los parámetros con gradientes más grandes reciben una tasa de aprendizaje más pequeña, y viceversa. La actualización de los parámetros es:

$$G_t = G_{t-1} + \nabla_\theta L(\theta_t)^2 \tag{7}$$

$$\theta_{t+1} = \theta_t - \frac{\eta}{\sqrt{G_t + \epsilon}} \nabla_\theta L(\theta_t) \tag{8}$$

Donde G_t es la suma acumulada de los cuadrados de los gradientes hasta el paso t, y ϵ es un valor pequeño para evitar divisiones por cero. Adagrad es efectivo en problemas con características dispersas, como en la clasificación de texto o el análisis de datos esparzos [**?**].

Comparación de Métodos

- **Adam** es generalmente el más eficiente para una variedad de tareas debido a su combinación de momentos y su capacidad para adaptarse a las características del problema, especialmente cuando se usan redes neuronales profundas.
- **RMSProp** es particularmente útil en problemas no estacionarios y puede ser más efectivo que Adam en algunos contextos de aprendizaje en línea.
- **Adagrad** funciona bien para problemas donde los datos son dispersos, pero puede sufrir de una disminución excesiva de la tasa de aprendizaje en problemas más complejos.

Ejemplo

Supongamos que estamos entrenando un modelo de regresión utilizando Adagrad. La función de pérdida es el error cuadrático medio (MSE) y la actualización de los parámetros se realiza utilizando la fórmula de Adagrad. A medida que avanzan las iteraciones, la tasa de aprendizaje de los parámetros disminuye, lo que permite que el modelo aprenda de manera más eficiente y estable en función de los gradientes acumulados. Esto puede ayudar a acelerar la convergencia en comparación con el descenso de gradiente estándar.

2.3 Aplicaciones en Conjuntos de Datos Peruanos

El uso de métodos de optimización estocástica como **Stochastic Gradient Descent (SGD)** y **Métodos Adaptativos (Adam, RMSProp, Adagrad)** se ha expandido significativamente en la investigación sobre ciencia de datos en Perú, particularmente en áreas como el análisis de grandes volúmenes de datos y el aprendizaje automático. Estos métodos se aplican a conjuntos de datos peruanos

en varios campos, tales como el análisis de imágenes satelitales para monitoreo ambiental, predicción de tendencias en la economía, y en el procesamiento de datos en redes sociales para comprender patrones de comportamiento.

2.3.1 Stochastic Gradient Descent (SGD)

SGD se ha utilizado en Perú para aplicaciones de predicción de series temporales, especialmente en el análisis de datos económicos y financieros. Por ejemplo, en la predicción de precios de commodities, donde se utiliza SGD para entrenar modelos que puedan predecir fluctuaciones de precios a partir de datos históricos de mercado. Además, en la clasificación de imágenes de satélite, SGD se usa para optimizar redes neuronales profundas, facilitando la identificación de patrones en imágenes de cultivos agrícolas, una tarea relevante en estudios sobre el uso de la tierra y el medio ambiente en Perú [**?**].

Fórmula de SGD:

$$\theta_{t+1} = \theta_t - \eta \nabla_\theta J(\theta_t)$$

donde:

- θ_t es el parámetro del modelo en el tiempo t,
- η es la tasa de aprendizaje,
- $J(\theta_t)$ es la función de costo, y
- $\nabla_\theta J(\theta_t)$ es el gradiente de la función de costo respecto a los parámetros.

Aplicación en Perú: Un ejemplo concreto es el uso de SGD en la predicción del rendimiento de cultivos agrícolas a partir de datos de sensores remotos. En estos casos, el modelo entrenado mediante SGD es capaz de predecir la cantidad de producto de ciertos cultivos en base a características meteorológicas y ambientales, lo que permite mejorar la eficiencia en la producción agrícola.

2.3.2 Métodos Adaptativos (Adam, RMSProp, Adagrad)

Los métodos adaptativos como **Adam**, **RMSProp** y **Adagrad** se utilizan en Perú en una variedad de sectores, como el análisis de datos financieros, la predicción de enfermedades, y el monitoreo del clima. En el contexto económico, estos métodos permiten optimizar modelos que analizan datos de grandes dimensiones, como los obtenidos en el análisis de transacciones bancarias o en la previsión de demanda de productos a nivel nacional.

Ejemplo con Adam: Adam se ha implementado para optimizar modelos predictivos que analizan datos de consumo energético en ciudades peruanas. Por

ejemplo, al ajustar los parámetros de un modelo de redes neuronales profundas que predicen la demanda de electricidad, Adam puede manejar la complejidad de los datos, mejorando la convergencia y la precisión de las predicciones [**?**].

Fórmula de Adam:

$$\hat{m}_t = \frac{m_t}{1-\beta_1^t}, \quad \hat{v}_t = \frac{v_t}{1-\beta_2^t}$$

$$\theta_{t+1} = \theta_t - \frac{\eta}{\sqrt{\hat{v}_t} + \epsilon}\hat{m}_t$$

donde:

- $\hat{m}_t$ y $\hat{v}_t$ son las estimaciones corregidas de primer y segundo momento,
- β_1 y β_2 son los parámetros de decaimiento,
- ϵ es un valor pequeño para evitar la división por cero.

Comparación: Si bien SGD es eficiente en tareas de optimización con grandes volúmenes de datos, los métodos adaptativos como Adam permiten una convergencia más rápida y una mayor precisión, especialmente en contextos donde los datos son ruidosos o los gradientes son esparsos. En las aplicaciones en Perú, los métodos adaptativos se han mostrado más efectivos para tareas complejas que involucran redes neuronales profundas y grandes bases de datos.

Referencias

- Robbins, H., & Monro, S. (1951). A Stochastic Approximation Method. *The Annals of Mathematical Statistics, 22(3)*, 400–407. https://doi.org/10.1214/aoms/1177729586

- Bottou, L. (2010). Large-Scale Machine Learning with Stochastic Gradient Descent. In *Proceedings of the 19th International Conference on Computational Statistics (COMPSTAT 2010)*, 177–186. https://doi.org/10.1007/978-3-7908-2604-3_17

- Kingma, D. P., & Ba, J. (2015). Adam: A method for stochastic optimization. *International Conference on Learning Representations (ICLR)*. https://doi.org/10.48550/arXiv.1412.6980

- Hinton, G. E. (2012). A practical guide to training restricted Boltzmann machines. *Proceedings of the 25th International Conference on Neural Information Processing Systems (NIPS 2012)*, 1–9. https://doi.org/10.1145/2993240.2993290

- Duchi, J., Hazan, E., & Singer, Y. (2011). Adaptive Subgradient Methods for Online Learning and Stochastic Optimization. *Journal of Machine Learning Research, 12*, 2121–2159. https://doi.org/10.1162/jmlr.2011.12.1.2121