Levenberg-Marquardt (LM) algoritması, özellikle yapay sinir ağlarının eğitimi için yaygın olarak kullanılan bir optimizasyon algoritmasıdır. Bu algoritma, hem Gauss-Newton hem de Gradient Descent yöntemlerinin avantajlarını birleştirir ve hızla yakınsayan bir çözüm sağlar. LM algoritması, özellikle orta boyutlu veri setleri ve modeller için uygundur.

Levenberg-Marquardt Algoritmasının Temel Prensipleri

1. Temel Kavramlar

- **Hata Fonksiyonu:** $E(\mathbf{w}) = \frac{1}{2}\sum_{i=1}^{N}(y_i - \hat{y}_i)^2$
  - Burada, y_i gerçek çıktı, $\hat{y}_i$ ise modelin tahmin ettiği çıktıdır.
  - Hata fonksiyonu, modelin tahminleri ile gerçek değerler arasındaki farkın karesini alarak toplam hata miktarını ölçer.

2. Güncelleme Kuralı

LM algoritması, ağırlık güncellemesini şu şekilde gerçekleştirir:
$\mathbf{w}_{k+1} = \mathbf{w}_k - (\mathbf{J}^T\mathbf{J} + \lambda\mathbf{I})^{-1}\mathbf{J}^T\mathbf{e}$

- Burada, $\mathbf{J}$ Jacobian matrisi (hata fonksiyonunun türevlerini içerir).
- λ damping parametresi (uyarlanabilir ve iterasyonlara bağlı olarak değişir).
- $\mathbf{I}$ birim matris.
- $\mathbf{e}$ hata vektörü (gerçek ve tahmin edilen değerler arasındaki fark).

$$J(\underline{x}) = \begin{bmatrix} \frac{\partial e_1(\underline{x})}{\partial x_1} & \frac{\partial e_1(\underline{x})}{\partial x_2} & \cdots & \frac{\partial e_1(\underline{x})}{\partial x_n} \\ \frac{\partial e_2(\underline{x})}{\partial x_1} & \frac{\partial e_2(\underline{x})}{\partial x_2} & \cdots & \frac{\partial e_2(\underline{x})}{\partial x_n} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial e_N(\underline{x})}{\partial x_1} & \frac{\partial e_N(\underline{x})}{\partial x_2} & \cdots & \frac{\partial e_N(\underline{x})}{\partial x_n} \end{bmatrix}$$

$$\overline{W}^{(t+1)} = \overline{W}^{(t)} - \overline{H}^{-1}\overline{g}$$

$$\overline{g} = \overline{J}^T\overline{e}$$

Adımlar

1. **Başlatma:** Başlangıç ağırlıklarını ($\mathbf{w}_0$) belirleyin ve damping parametresini (λ) seçin.
2. **Hata Hesaplama**: Mevcut ağırlıklarla modelin hata fonksiyonunu ($E(\mathbf{w})$) hesaplayın.
3. **Jacobian Matrisi**: Mevcut ağırlıklarla Jacobian matrisini ($\mathbf{J}$) hesaplayın.
4. **Güncelleme**:
   - Damping parametresi (λ) küçükse, Gauss-Newton yöntemi gibi davranır.
   - Damping parametresi (λ) büyükse, Gradient Descent yöntemi gibi davranır.
5. **Ağırlıkların Güncellenmesi**: Ağırlıkları güncellemek için yukarıdaki güncelleme kuralını kullanın.
6. **Hata Kontrolü**: Yeni ağırlıklarla hata fonksiyonunu yeniden hesaplayın.
   - Eğer hata azalmışsa, λ'yı küçültün ve yeni ağırlıkları kabul edin.
   - Eğer hata artmışsa, λ'yı büyütün ve eski ağırlıkları koruyun.
7. **Durdurma Kriteri:** Belirli bir durdurma kriteri (örneğin, maksimum epoch sayısı veya minimum hata eşiği) karşılanana kadar adımları tekrarlayın.