Peran Turunan dalam Gradient Descent

Turunan menunjukkan **seberapa cepat dan ke arah mana** suatu fungsi berubah terhadap variabelnya.

Jika kita punya fungsi *Loss* (L(w)), di mana (w) adalah parameter model:

- **Turunan ($\frac{dL}{dw}$)** memberi tahu kita:
    - Apakah *loss* meningkat atau menurun ketika kita mengubah (w).
    - Seberapa besar perubahan *loss* tersebut untuk perubahan kecil pada (w).

Dengan informasi ini, kita dapat **mengubah parameter ke arah yang menurunkan loss** — itulah esensi *gradient descent*.

Rumus Umum Gradient Descent

$$w_{baru} = w_{lama} - \eta \frac{dL}{dw}$$

Keterangan:

- (η): *learning rate*, menentukan seberapa besar langkah perubahan.
- ($\frac{dL}{dw}$): turunan dari fungsi *loss* terhadap parameter (w).

Jadi, setiap iterasi akan menyesuaikan parameter sedikit demi sedikit ke arah *minimum loss* berdasarkan nilai turunan tersebut.

Contoh pada Regresi Linier

Misalkan modelnya: $\hat{y} = wx + b$ dan fungsi *loss* (Mean Squared Error): $L = \frac{1}{n} \sum_{i=1}^{n} (y_i - \hat{y}_i)^2$

Kita hitung turunan terhadap (w) dan (b): $\frac{dL}{dw} = -\frac{2}{n} \sum (y_i - \hat{y}_i) x_i$ $\frac{dL}{db} = -\frac{2}{n} \sum (y_i - \hat{y}_i)$

Dari turunan inilah algoritma *gradient descent* tahu apakah bobot perlu dinaikkan atau diturunkan untuk memperkecil kesalahan.

Pada Jaringan Saraf Tiruan (Neural Networks)

Konsep yang sama digunakan tetapi dengan kompleksitas lebih tinggi.

- Turunan digunakan dalam **backpropagation**, yaitu proses menghitung *gradient* dari *loss* terhadap setiap bobot dalam jaringan.
- Dengan menggunakan **aturan rantai (chain rule)** dari kalkulus, jaringan dapat menghitung bagaimana setiap bobot di setiap lapisan memengaruhi *loss akhir*.
- Setelah semua *gradient* dihitung, bobot diperbarui menggunakan rumus *gradient descent* (atau variannya seperti Adam, RMSProp, dll).