Logistic Regression Optimization using Gradient Descent and Newton’s Method

GRADIENT DESCENT VS NEWTON'S METHOD IN LOGISTIC REGRESSION

- Optimizing Logistic Regression Cost Function
- Comparison of Convergence Speed & Stability (Graphic: one curve, GD takes many small steps, Newton jumps closer to the minimum)

LOGISTIC REGRESSION RECAP

- **Predicts probability:**

$$h_\theta(x) = \frac{1}{1 + e^{-\theta^T x}}$$

- **Cost function (log-loss):**

$$J(\theta) = -\frac{1}{m} \sum_{i=1}^{m} \left[y^{(i)} \log h_\theta(x^{(i)}) + (1 - y^{(i)}) \log(1 - h_\theta(x^{(i)})) \right]$$

- **Goal:** Minimize $J(\theta)$

GRADIENT DESCENT (GD)

- **Iterative update:**

$$\theta := \theta - \alpha \nabla J(\theta)$$

- **Gradient:**

$$\nabla J(\theta) = \frac{1}{m} X^T (h_\theta(X) - y)$$

Key points:

- Needs learning rate (α\alphaα)
- May require many iterations
- Works well for large datasets

NEWTON'S METHOD

Uses second-order approximation:

$$\theta := \theta - H^{-1}\nabla J(\theta)$$

Where Hessian is:

$$H = \frac{1}{m}X^T RX, \quad R = diag\left(h_\theta(x^{(i)})(1 - h_\theta(x^{(i)}))\right)$$

Key points:

- No learning rate needed
- Converges in fewer steps
- Hessian computation is expensive in high dimensions.

CONVERGENCE COMPARISON

Aspect	Gradient Descent	Newton’s Method
Iterations	Many	Few
Step size	Needs tuning	Automatic
Computation per step	Cheap (gradient only)	Expensive (Hessian + inverse)
Stability	May oscillate/diverge	Stable if Hessian invertible
Large datasets	Scales well	Not practical

EXAMPLE BEHAVIOR

- Gradient Descent: slow, may take 100+ iterations.
- Newton’s Method: fast, often <10 iterations.
- Trade-off: per-iteration cost vs total iterations.

CONCLUSION

- Gradient Descent: best for large-scale, high-dimensional problems
- Newton's Method: very fast but costly to compute
- Hybrid approaches (e.g., BFGS): balance speed & efficiency

THANK YOU

FOR YOUR SUPPORT