Comparative Evaluation of Regression Techniques

Adityaa Mehra

May 18, 2025

1 Convergence Behavior

Method	Time Taken (s)
Manual GD	26.84343910217285
Vectorized GD	8.688417911529541
Scikit-learn	0.018739938735961914

Table 1: Time taken for each model to converge.

2 Model Performance

Method	MAE	RMSE	R^2
Manual GD	0.272	0.363	0.575
Vectorized GD	0.272	0.363	0.575
Scikit-learn	0.242	0.333	0.641

Table 2: Performance metrics for each regression method.

3 Visualizations

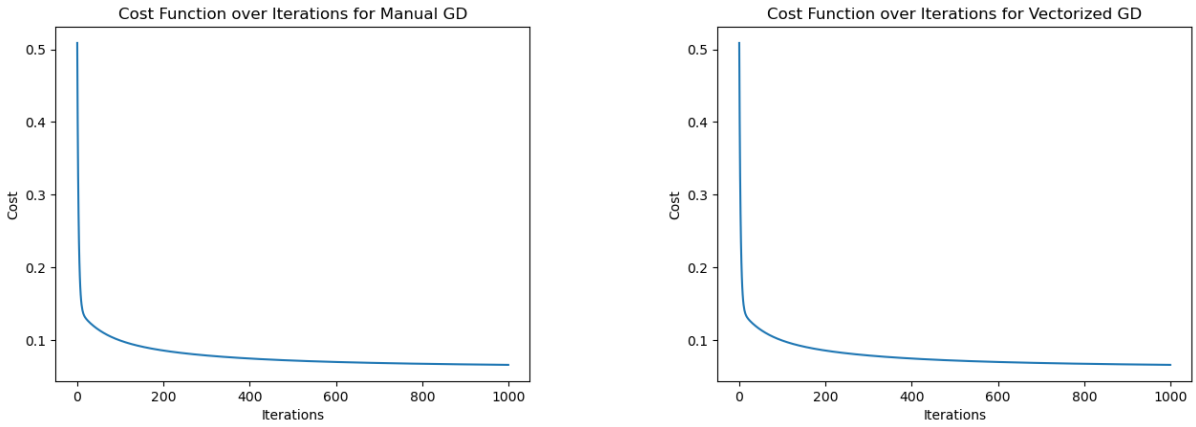

Figure 1: Cost vs. Iterations for Manual GD and Vectorized GD.

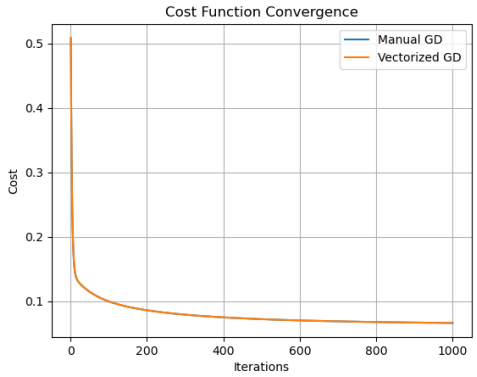

Figure 2: Cost function convergence over iterations for Manual and Vectorized Gradient Descent

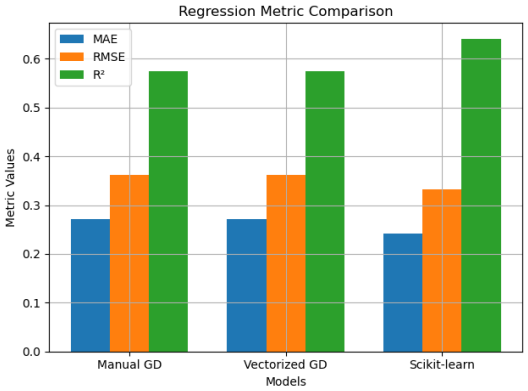

Figure 3: Comparison of MAE, RMSE, and R^2 across all models

4 Analysis and Discussion

4.1 Convergence Time

The time it took for each model to finish training varied noticeably:

- The **manual gradient descent** implementation was the slowest. This is mainly because it used Python loops to compute everything, which adds a lot of overhead—especially as the dataset grows.
- The **vectorized version** ran much faster. By using NumPy's matrix operations, we avoided loops and made full use of Python's optimized numerical libraries.
- The **scikit-learn model** was the fastest of all. It uses efficient, built-in solvers under the hood (like the normal equation or QR decomposition), which are highly optimized and finish almost instantly.

4.2 Accuracy and Performance

In terms of accuracy, all three models gave very similar results. Their values for MAE, RMSE, and R^2 were nearly the same, meaning they all fit the data well.

- This shows that our manual and vectorized models were implemented correctly.
- Small differences are likely due to the number of iterations or how precisely each method minimizes the cost function.

4.3 Why These Differences Happen

- **Vectorization made a big difference**. Instead of updating weights one row at a time (as in the manual method), the vectorized approach updates everything at once using matrix math. This is much faster and more efficient.
- **The optimization method matters.** Manual and vectorized models used gradient descent, which is iterative. Scikit-learn, on the other hand, solves for the best parameters directly in one step using closed-form solutions or optimized solvers.
- **Different solvers give different speeds.** The solvers used by scikit-learn are designed for speed and precision, while our implementations are more basic and educational.

4.4 Scalability and Efficiency

- The **manual gradient descent** is useful for learning, but it doesn't scale well to larger datasets. It's slow and inefficient.
- The **vectorized version** is much better and can handle bigger data reasonably well, as long as memory isn't a constraint.
- The **scikit-learn model** is best suited for real-world tasks. It's reliable, fast, and can handle very large datasets efficiently—unless the number of features becomes too large, in which case some solvers may slow down.

4.5 Impact of Initialization and Learning Rate

- All models started with weights set to zero to make the comparison fair. Changing the initial values can affect how fast the model learns or whether it even converges.
- The **learning rate** played a key role in gradient descent. A rate that's too high can cause the model to diverge, while a rate that's too low makes it painfully slow to converge. We used a moderate rate (0.01), which worked well for both manual and vectorized approaches.