Problems With SGDRegressor+Ema For Resource Usage Predictions

**1. Assume Linearity Between Features and Target**

Linear models make a strong assumption: the relationship between input features (e.g., CPU usage, request rate) and the output (e.g., resource delta) is linear.

Real-world systems are often:

* Nonlinear
* Threshold-based (e.g., latency spikes only after 80% CPU)
* Multi-phase (e.g., different behavior under low/high load)

**2. Struggle with Interactions**

Linear models can’t natively model feature interactions unless we manually add interaction terms.

Example:

* We may need terms like CPU\_Usage × RequestRate or CPU\_Utilization² to capture important behaviors.

**3. Poor Fit on Discontinuous Patterns**

If the target variable changes sharply (e.g., latency jumps when memory crosses a limit), linear models can't capture that step.

**4. Sensitive to Feature Scaling**

Linear models produce:

* Slow convergence
* Poor generalization

…if the input features aren’t properly scaled or normalized.

**5. Outliers Can Skew Learning**

Even with robust regression settings, linear models are still influenced by:

* Sudden spikes (e.g., crash recovery CPU spikes)
* Outlier resource usage under abnormal load

**6. One-Slope-For-All Limit**

They learn one set of weights globally. That means:

* One trend line has to explain the whole dataset
* Can't adapt locally for different traffic modes (e.g., idle vs peak)

## **Example in Kubernetes Context**

Suppose CPU usage stays flat at 200m under 300 rps, but jumps sharply to 400m at 600 rps.

A linear model would try to draw a single line between those two points, underestimating low-load usage and overestimating high-load usage.

| **Weakness** | **Result** |
| --- | --- |
| Linear assumption | Poor fit on real-world workloads |
| No interactions | Can't capture compound effects |
| Global model | No local adaptation |
| No thresholds | Misses step changes |
| Sensitive to scale | Needs preprocessing |