Hacker Role

Solving Large Scale LinearPrediction Problems Using Stochastic Gradient Descent Algorithms

A Comparison of SGD Variants:
Analysis of Convergence, Regularization, and Efficiency

Presented by
Punyo Sunya- 220101082
Talari Syama Sameeksha- 220101097
Vidhi Arora- 220101111
Dalisha Nagpal- 220108019

31st October, 2025

Problem Statement

Solving Large-Scale Linear Prediction Problems

Core Problem: The Need for Efficiency

Traditional Batch Learning is computationally prohibitive for large datasets (slow and memory intensive).

- **Limitation 1:** Requires computing the gradient over all *N* training samples in every step, making iterations very slow.
- **Limitation 2:** Not suitable in an online setting.

Solution

Stochastic Gradient Descent (SGD)

SGD algorithms offer a simple, efficient alternative for large-scale applications.

1. **Efficiency:** They update the weight vector using just one data point (or a small batch) at a time, making them fast.
2. **Implementation:** Simple to implement.
3. **Online Algorithm:** Learns as data streams in.
4. **Context:** This paper focuses on applying SGD to regularized linear prediction methods, including:

- Least Squares (for regression)
- Logistic Regression (for classification)
- Support Vector Machines (SVM) (for classification)

Core Algorithm - Regularized SGD Update Rule

Standard Stochastic Gradient Descent (Algorithm 2.1)

Goal: Minimize the regularized expected loss

$$Q_\lambda(w) = E_{X,Y}\phi\left(w^T X, Y\right) + \frac{\lambda}{2}\|w\|_2^2$$

Update Rule

$$w_t = w_{t-1} - \eta_t \times g_t$$

where the stochastic gradient is

$$g_t = \lambda w_{t-1} + \phi_1^{'}\left(w_{t-1}^T X_t, Y_t\right) X_t$$

Key Parameters:

- **Learning Rate:** η
  Step size. The paper argues for a small, constant η for fast convergence.
- **Regularization:** λ
  Penalty for large weights (L2).

Variance Reduction with Averaged SGD

Concept

$$\hat{v}_t = \frac{r_{t-1}}{r_t}\hat{v}_{t-1} + \frac{r_t - r_{t-1}}{r_t}\hat{w}_{t-1}$$

$$r_t = r_{t-1} + \eta_t - \frac{AM^2{\eta_t}^2}{2}$$

For constant learning rates β, the rule simplifies to the arithmetic mean:

$$\hat{v}_T = \frac{1}{T}\sum_{t=1}^{T}\hat{w}_{t-1}$$

The model is updated with $\hat{w}$, but the final predictor is the average $\hat{v}$

Benefit

Reduces variance, leading to smoother convergence and better asymptotic performance.

$\hat{v}_t$	Averaged Weight Vector at time t
$\hat{w}_{t-1}$	Previous Weight Vector
T	Total Number of Iterations/steps

Code Results

```
==================================================
EXPERIMENT 1: Baseline (Paper's Recommendation)
Small eta, No Regularization, Early Stopping (10 epochs)
==================================================
--- Standard SGD (SVM Loss) ---
  > Test Accuracy: 83.00%
  > Test Precision: 79.08%
  > Time Taken: 148.35 ms
  > Final |w| (L2-Norm): 1.6984


--- Averaged SGD (SVM Loss) ---
  > Test Accuracy: 82.00%
  > Test Precision: 78.29%
  > Time Taken: 346.42 ms
  > Final |w| (L2-Norm): 1.3045
```

```
==================================================
EXPERIMENT 2: Effect of Large Learning Rate (eta)
Large eta (0.5), No Regularization, 10 epochs
==================================================
--- Standard SGD (Large eta) ---
  > Test Accuracy: 77.67%
  > Test Precision: 72.12%
  > Time Taken: 186.98 ms
  > Final |w| (L2-Norm): 12.7399


--- Averaged SGD (Large eta) ---
  > Test Accuracy: 84.00%
  > Test Precision: 79.49%
  > Time Taken: 156.46 ms
  > Final |w| (L2-Norm): 10.2493
```

```
==================================================
EXPERIMENT 3: Regularization (lambda) vs. Early Stopping
==================================================
(Run A: Early Stopping - Same as baseline)
--- Standard SGD (Early Stop) ---
  > Test Accuracy: 82.67%
  > Test Precision: 78.95%
  > Time Taken: 210.72 ms
  > Final |w| (L2-Norm): 1.6878


(Run B: Explicit Regularization - More epochs)
--- Standard SGD (Explicit Lambda) ---
  > Test Accuracy: 84.00%
  > Test Precision: 80.26%
  > Time Taken: 964.73 ms
  > Final |w| (L2-Norm): 1.8280
```

Comparison 1: SGD vs. Averaged SGD (Variance Reduction)

Standard Stochastic Gradient Descent

After the final update, the weight vector has higher variance, especially with large learning rates.

Averaged Stochastic Gradient Descent

The average of all weight vectors computed during the run.

$$v_T = \frac{1}{T} \sum_{t=1}^{T} w_{t-1}$$

Averaging reduces variance and smooths the convergence curve.
However: For SGD with a small learning rate η , averaging is much less important and may be inferior to the non-averaged result.

Comparison 2: Implicit Regularization via Early Stopping

The Problem with Unregularized Batch Learning:

Prone to overfitting on the training data.

SGD's Advantage:

- Running SGD for only a small number of iterations (T) acts as an implicit regularization mechanism.
- It prevents the model from minimizing the empirical loss too perfectly.

Observation: Optimal performance is often reached in the early stages of training (e.g., ~10 iterations over the data), demonstrating extreme efficiency.

Comparison 3: SGD (Hinge Loss) vs. Perceptron

Stochastic Gradient Descent (SGD) with Hinge Loss (SVM)

By using a small, constant learning rate, SGD introduces less variance in the weight updates, leading to a more stable and superior estimator compared to the Perceptron.

The Perceptron Algorithm

A classic online algorithm that can be theoretically viewed as a special case of SGD with the SVM Hinge Loss when the learning rate is effectively very large $\eta \to \infty$

Because it implicitly uses a large learning rate, the non-averaged Perceptron has a relatively large variance in its updates, making it less stable

The Result: SGD with a carefully chosen small learning rate and the Hinge Loss is theoretically and experimentally superior to the standard Perceptron method, achieving better statistical consistency (lower true risk).

Key Findings & Statistical Implications

- **SGD's Statistical Superiority:** The method of minimizing the regularized loss function (e.g., SVM Hinge Loss) using SGD is both theoretically and experimentally superior to classic online algorithms like the Perceptron. This superiority stems from achieving better statistical consistency (a lower true risk).
- **Impact on Practice:** The combination of fast, theoretically-backed convergence and implicit regularization makes SGD the most suitable and efficient choice for training linear models in today's high-dimensional, large-scale data environments.
- **Optimal Learning Rate & Variance Control:**
  - The convergence analysis shows that the optimal rate can be attained using a simple, constant learning rate η.
  - Averaging primarily serves as a variance reduction mechanism. It is less important when a small, stable learning rate (η) is used for SGD, but is still useful if a high η is necessary.

Experimental Logs - Parameters and Results

Experiment	η (eta)	λ (lambda)	Epochs	Averaging (Yes/No)	Test Accuracy	Precision	Time (ms)	$\|\|w\|\|_2$	Key Observations
1. Baseline	Small	0	10	No	83.00%	79.08%	148.35	1.6984	Strong implicit reg
				Yes	82.00%	78.29%	346.42	1.3045	Slightly lower norm
2. Large η	0.5	0	10	No	77.67%	72.12%	186.98	12.5399	Diverges
				Yes	84.00%	79.49%	156.46	10.2493	Averaging saves it
3A. Early Stop	Small	0	10	No	82.67%	78.95%	210.72	1.6878	Fast + good gen
3B. Explicit λ	Small	>0	More	No	84.00%	80.26%	964.73	1.8280	Slower, similar perf

Experiment Result

Theoretical Claim (Paper)	Parameter / Setting	Observed Result
SGD is efficient: Few passes over data suffice	10 epochs only	All runs < 1 sec (148–965 ms)
Early stopping = implicit regularization	10 epochs, no explicit λ	$\|\|w\|\|_2 \approx 1.7$ (small)
Small constant learning rate (η) works well	Small η (we took 0.002)	Accuracy: 83% (Std), 82% (Avg)
Large η → Standard SGD diverges	η = 0.5	Acc drops to 77.67%, $\|\|w\|\|_2 = 12.74$
Averaging fixes large η instability	η = 0.5 + averaging	Acc recovers to 84%, $\|\|w\|\|_2 = 10.25$

Experiment Result

Theoretical Claim (Paper)	Parameter / Setting	Observed Result
Averaging gives stable convergence	Averaged SGD	Lower variance in $\|w\|_2$, better recovery
Explicit λ needs more epochs/time	Explicit λ + more epochs	Accuracy : 84% but 5 times slower
Early stop ≈ Explicit reg in performance	Regularization (λ) vs. Early Stopping	Accuracy : 82.7% (Early stopping) vs 84.0% (Regularization), similar $\|w\|_2$
No explicit λ needed	No λ, 10 epochs	Good generalization (83%)
Averaging improves over standard SGD	Compare Std vs Avg	Average SGD is more stabile in unstable cases

Conclusion: Why SGD is the Ideal Large-Scale Algorithm

- **Takeaway:** For large-scale linear prediction problems, Standard SGD with a small, constant learning rate offers the optimal balance of efficiency, simplicity and generalization performance.
- **Simplicity and Implementability:** The core SGD update rule is extremely simple, involving only basic vector arithmetic (a single gradient calculation per step), making it straightforward to implement, deploy and scale in real-world systems.
- **Extremely Efficienct:** SGD achieves convergence to a desired accuracy $\mathcal{E}$ in $O\left(\varepsilon^{-2}\right)$ steps. This rate is competitive and often translates to reaching optimal performance quickly, typically within a handful of passes over the dataset.
- **Implicit Regularization (Early Stopping):** Running SGD for only a small number of iterations provides an implicit regularization effect. This method of Early Stopping is easier to implement and often more robust than manually tuning the explicit regularization parameter λ.

Thank You

References

- Solving Large Scale Linear Prediction Problems Using Stochastic Gradient Descent Algorithms
- Implemented Code