Intelligently Evolving Gradient Descent

John Vahedi

Abstract

- Intelligent form of the gradient descent algorithm
- ”Semi-stochastic,“ informed by short-term history
- 3 evolutionary methods
- Compared to plain SGD
- Linear and non-linear examples
- For stable but complex neural & transformer architectures

1. Background
2. Algorithms
3. Method
4. Results & Analysis
5. Conclusion

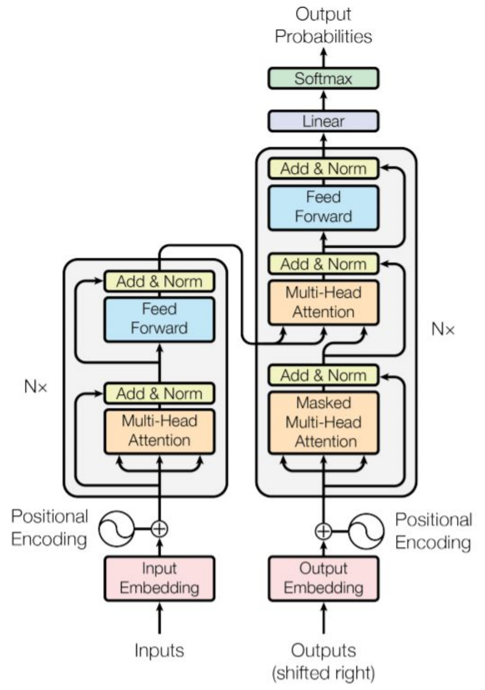


Figure 1: The Transformer - model architecture.

1.

Background

Introduction

Cost Function

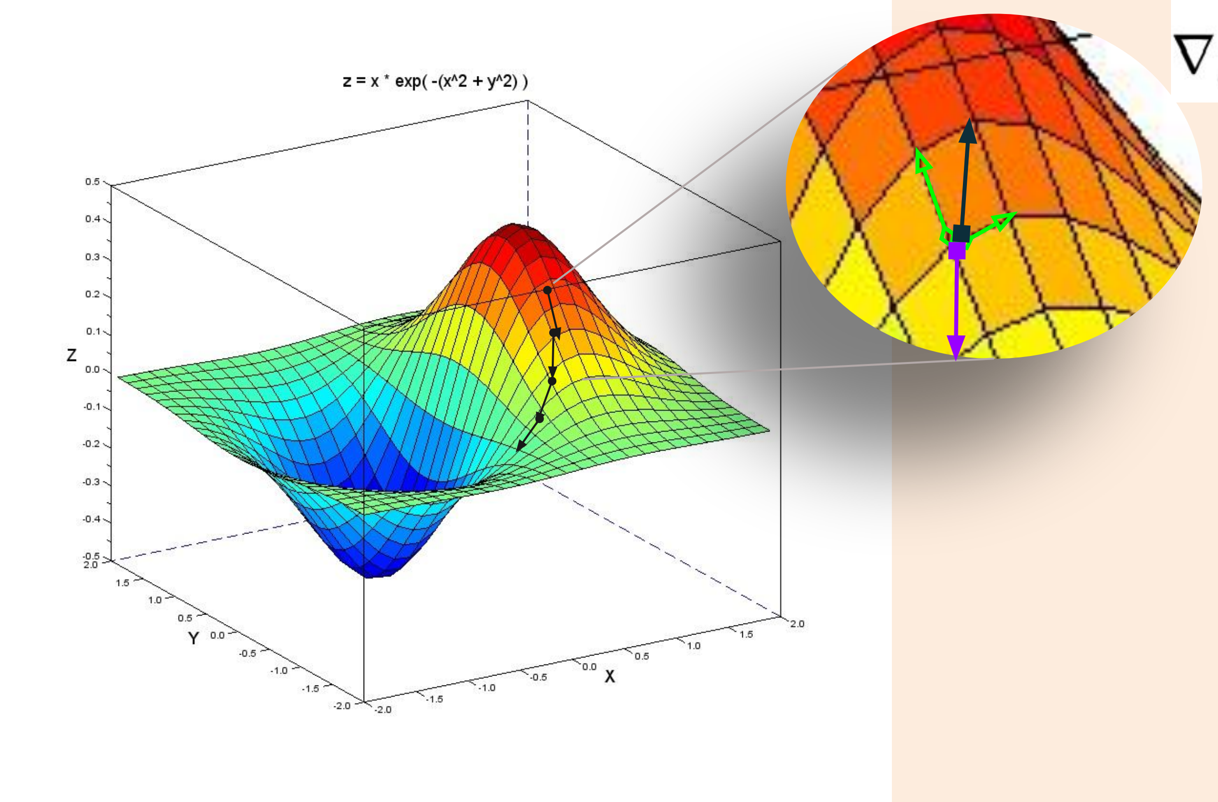
Inspired Example:

- ❖ Business
  - ➢ Reduce cost
  - ➢ Two dependent choices (*n*=2)
  - ➢ Current choices set

Note: Our business will be something different

And ‘n’ may be > 2

What is Gradient Descent?



$\nabla f(\omega)$

Gradient Descent Step

Mathematically:

Step Size

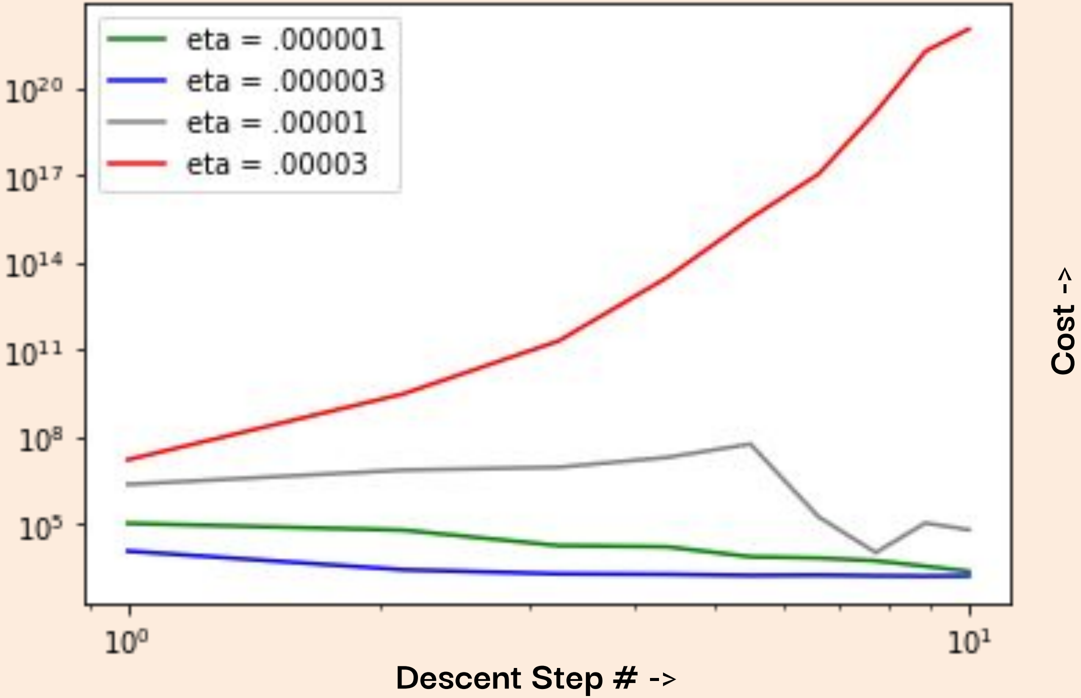
$$w_{i+1} = w_i - \eta_i \cdot \nabla_w f(w_i) + \epsilon, \qquad \eta_i = \frac{\eta_0}{i^p}, \qquad p \in [0, 1]$$

New position

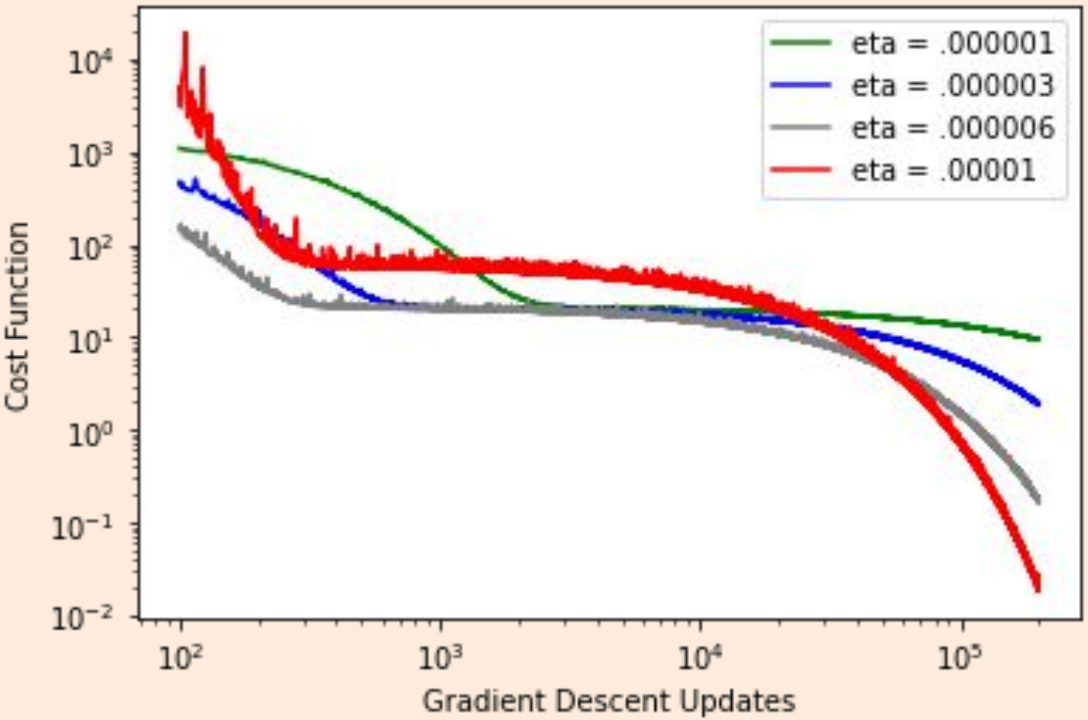
Current position

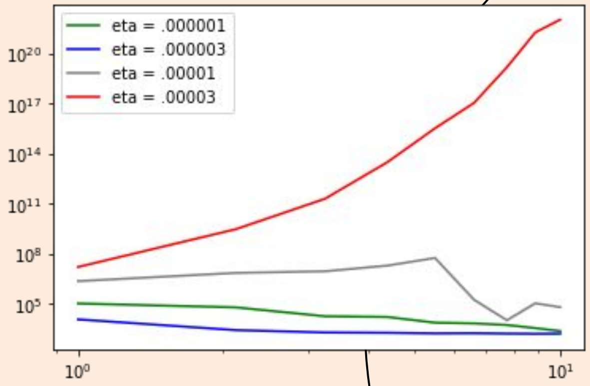
Gradient Direction

Step Size (3 Faux Pas)

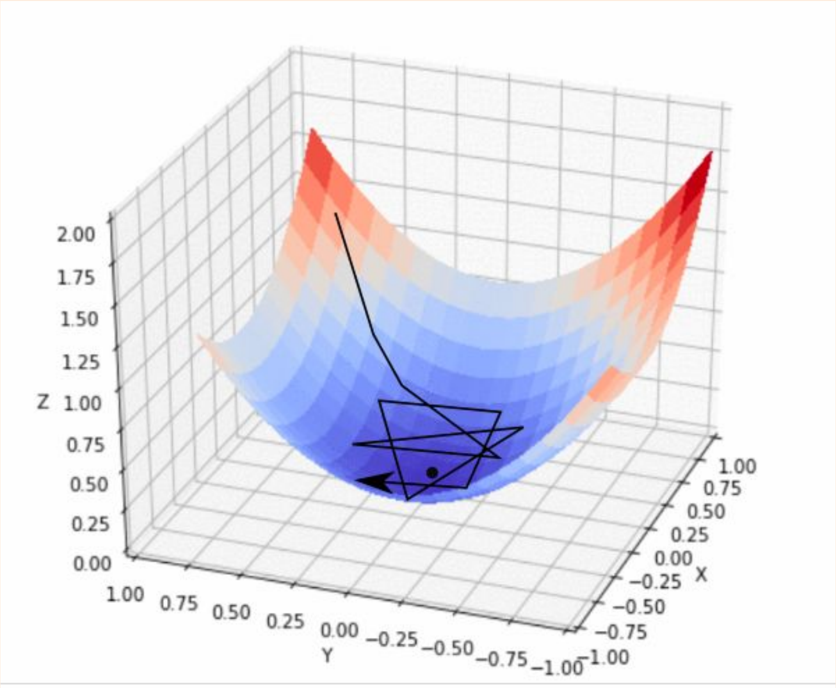


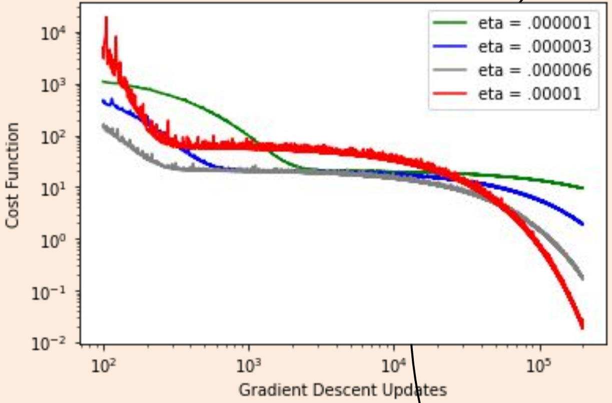
Step Size (3 Faux Pas)

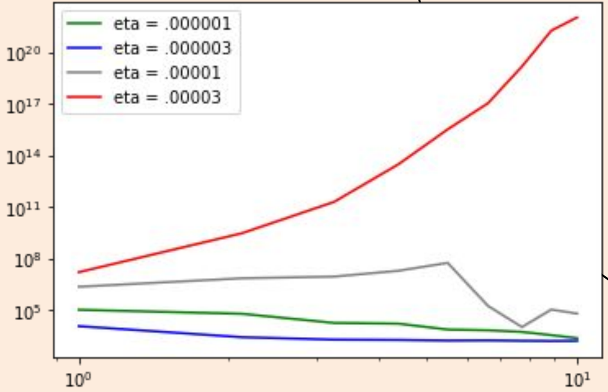




Step Size (3 Faux Pas)







Why Stochastic?

Definition:

- "Randomness"
- Avoiding some directions in Gradient randomly
  - Picks **subset** of gradient

Reason/strategy:

- Redundant: Data or Features
- Avoid Local Minima
- Memory Advantage

Regardless:

- Eventually reach all data (Downside Low, Upside High)

2.

Algorithms

Historical Gradient Descent

Momentum

Remembers old motion; piecemeals out same motion later

Adaptive

Adjust step size based on gradient or historical frequency

Noise

Explicit noise to added gradient; helps initialization

Implicit

Prescient methods; Add complexity but more stable

Too much stochastic....?

- Stochasticity has Benefits
  - Discussed
- Total Stochastic Choice
  - Leaves information on the table

Novel Gradient Descent

Semi-Stochastic Gradient Descent (SSGD)

- *Remembers last step*
- *Gradient as fitness*
- *Percent fittest, rolls over*

Note: Requires sorting

Novel Gradient Descent

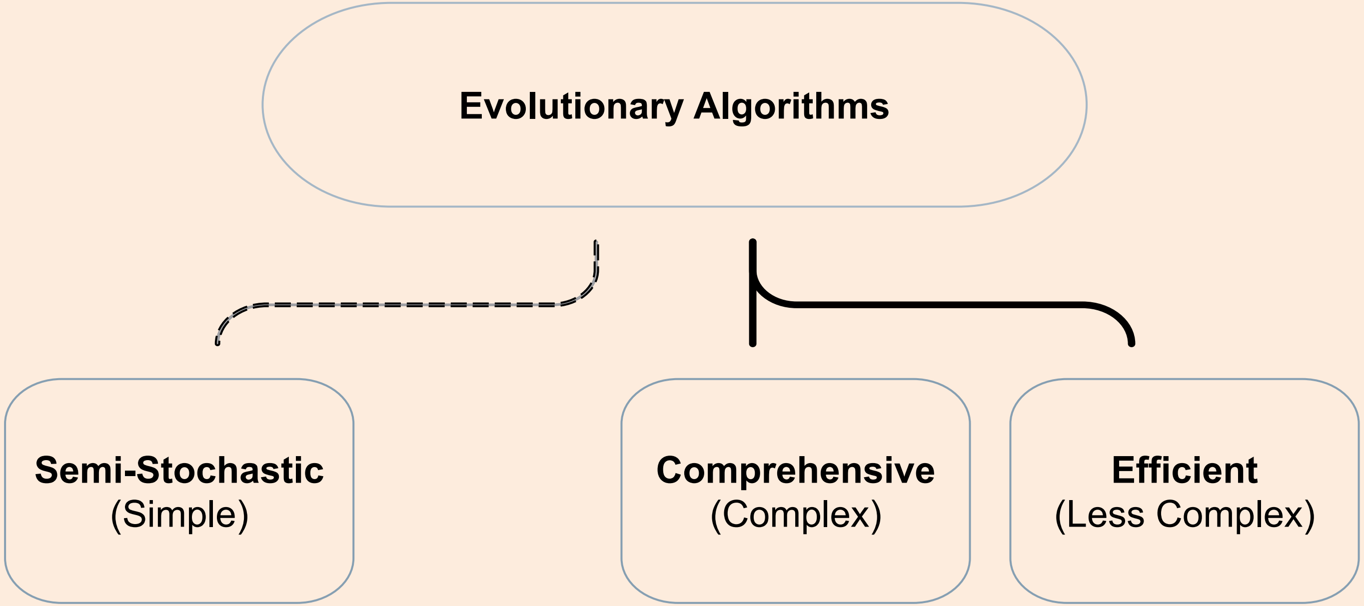
Algorithm 1 SSGD

Import data
Initialize subset of data, size K - call this subset L_k
Initialize a at initial guess a_0
for $i = 1, 2, \ldots, N$ **do**
    Set $\nabla_a \Phi$ = zeros(input space dimensionality)
    for $point = 1, 2, \ldots, K$ **do**
        Assess gradient at point L[point] and add to $\nabla_a \Phi$
    end for
    Update a=a$-(\eta/i^p)*(\nabla_a \Phi/K)$
    Selection on L_k, return $L_{k,selected}$ holding highest-contributing points
    Repopulate $L_{k,selected}$ with random points from dataset, set repopulated array equal to L_k
end for

“Those who cannot remember the past are condemned to repeat it.”

- George Santayana

Evolutionary Gradient Descent



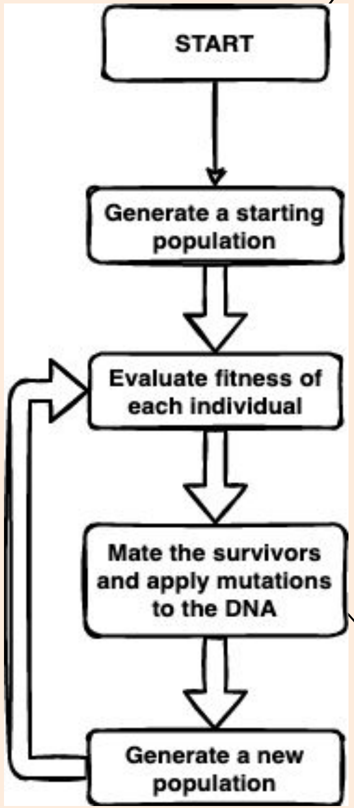
What is Evolutionary Algorithm?

➔ “Population” of solutions running in parallel

➔ Inspired by biological systems

Relevant mechanisms:

- *Fitness*
- *Breeding*
  - *Recombine*
  - *Mutation*



Evolutionary Gradient Descent

A. Initialize <u>**P**</u> random <u>subsets</u>
B. Calculate accumulated individual **Gradient**
C. Update **Parameters** individually and **Error**
D. Select *fittest* percentage for **Breeding**

Fitness

3 Factors:

$$\boxed{fitness = |\nabla_w \Phi^*| * \frac{1}{factor_{age}} * \frac{1}{(1 + |error|)}}$$

$$factor_{age} = \begin{cases} 1 & , age < C \\ (age - C) + 1, & otherwise \end{cases}$$

Evolutionary Gradient Descent

2. *Comprehensive Stochastic Gradient Descent (CEvSGD)*
   - ➔ Tracks individual’s parameters and stored
     - ◆ Returns parameter with lowest error

3. *Efficient Stochastic Gradient Descent (EEvSGD)*
   - ➔ Does not track individual's parameters
     - ◆ Global parameter updated with averaged gradient

Evolutionary Gradient Descent

Algorithm 2 EvSGD Variant 1: CEvSGD (Comprehensive EvSGD)

Import data
Initialize P subsets of data, each of size K - call this array L
Initialize array A of size P, entries equal to initial guess a_0
for $i = 1, 2, \ldots, N$ **do**
    for $member = 1, 2, \ldots, P$ **do**
        Set $L_k = L[\text{member}]$
        Set $\nabla_a \Phi = zeros(\text{input space dimensionality})$
        for $point = 1, 2, \ldots, K$ **do**
            Assess gradient at point $L_k[\text{point}]$and add to $\nabla_a \Phi[\text{point}]$
        end for
        $a = A[\text{member}]$
        Update $a{=}a{-}(\eta/i^p)*(\nabla_a \Phi[\text{member}]/K)$ and correspondingly update A
    end for
    Selection on L, return $L_{selected}$ which holds top performing subsets
    Run breeding on $L_{selected}$, return L_{new}
    Update A, with new members of the population initialized at $A[\text{parent}]$
end for

Algorithm 3 EvSGD Variant 2: EEvSGD (Efficient EvSGD)

Import data
Initialize an array L of P subsets of data, each of size K - L corresponds to our population
Initialize a equal to initial guess a_0
Initialize an array $\nabla_a \Phi$ of size P, each entry is $zeros(P)$
for $i = 1, 2, \ldots, N$ **do**
    for $member = 1, 2, \ldots, P$ **do**
        Set $L_k = L[\text{member}]$
        Set $\nabla_a \Phi[\text{member}] = zeros(P)$
        for $point = 1, 2, \ldots, K$ **do**
            Assess gradient at $L_k[\text{point}]$ and add to $\nabla_a \Phi[\text{member}]$
        end for
    end for
    Selection on L, return $L_{selected}$, which holds top-performing subsets
    $\nabla_{a,avg} \Phi{=}average(\nabla_a \Phi)$
    Update $a{=}a{-}(\eta/i^p)*(\nabla_{a,avg} \Phi/K)$
    Run breeding on $L_{selected}$, return L_{new}
end for

Evolutionary Gradient Descent

1. Semi-Stochastic Gradient Descent (SSGD)
2. Comprehensive Stochastic Gradient Descent (CEvSGD)
3. Efficient Stochastic Gradient Descent (EEvSGD)

Other “similar” algos

- “Evolutionary stochastic gradient descent for optimization of deep neural networks” by Xiaodong Cui et al.

- Momentum based SGDs

3.

Method

Imported Packages

MatPlotLib

Generate graphs of plots

Numpy

Used for parallel computation, and random value generator

Regression Problem

$$F(a,x) = a_1 \cdot f_1(x) + a_2 \cdot f_2(x) + \ldots a_N \cdot f_N(x) = \sum_{n=1}^{N} a_n \cdot f_n(x) \qquad (3)$$

$$\nabla_a F(a,x) = [f_1(x), f_2(x), \ldots, f_N(x)]^T \qquad (4)$$

$$\Phi(a) = \frac{1}{M} \sum_{j=1}^{M} \phi_j(a). \qquad \phi_j(a) = (y_j - F(a,x_j))^2.$$

$$\nabla_a \Phi(a) = \frac{1}{M} \sum_{j=1}^{M} \nabla_a \phi_j(a) = \frac{1}{M} \sum_{j=1}^{M} 2 \cdot (y_j - F(a,x_j)) \cdot \nabla_a F(a,x_j) \qquad (8)$$

Regression Problem

$$\Phi^*(a) = \frac{1}{\sum_{j=1}^{M} w_j} \sum_{j=1}^{M} w_j \phi_j(a) = \frac{1}{K} \sum_{j=1}^{M} w_j \phi_j(a)$$

$$\nabla_a \Phi(a) \approx \nabla_a \Phi^*(a).$$

$$\nabla_a \Phi(a) = \nabla_a \Phi^*(a) + err, \qquad err \sim N(0, \sigma) \Rightarrow$$

$$E[\nabla_a \Phi(a)] = E[\nabla_a \Phi^*(a)]$$

$$a_{i+1} = a_i - \eta_i \cdot \nabla_a \Phi^*(a_i) \tag{10}$$

Created Packages

Utils

EA_Utils

Generator

Dependencies

none

Parameter Tuning

η (eta)

Initial step size

p

Power of series used in division to slow down step sizes over iteration

other

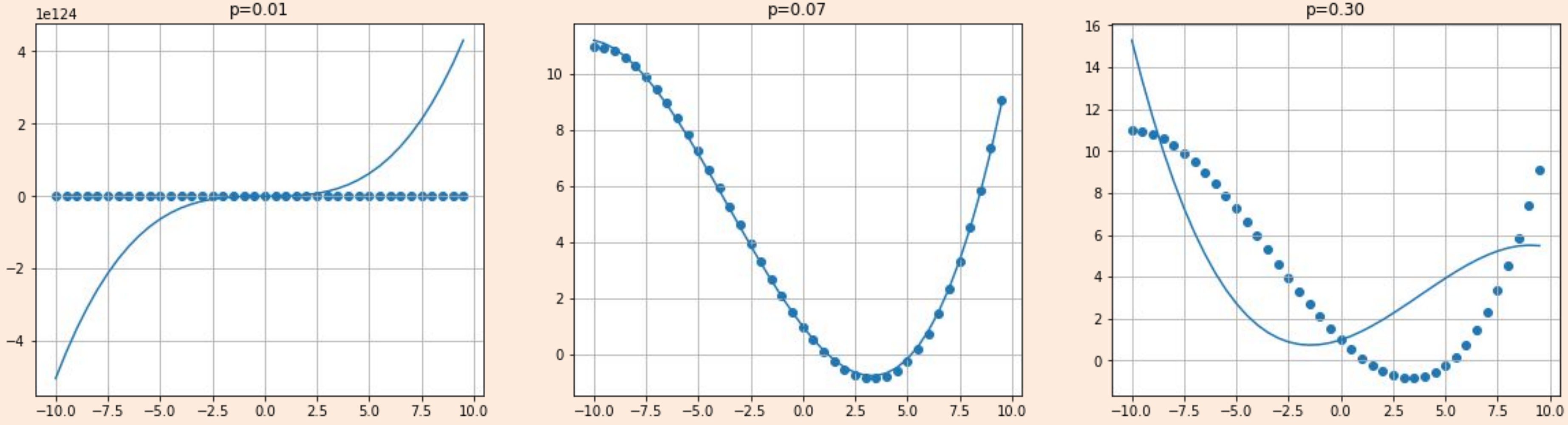
%Retained, %Mutation, %Inheritance, Age Cutoff

Parameter Tuning

p

Power of series used in division to slow down step sizes over iteration

Effect of different p values for a fixed η=0.000001 on SSGD



Parameter Tuning

Algorithm	Optimal p	Optimal η_0
1: SSGD	0.07	9×10^{-6}
2: EEvSGD	0	1.5×10^{-7}
3: CEvSGD	0.07	8.1×10^{-6}

Search done: **p** in [0,.5] and **eta** in [1 x 10^-7, 5 x 10^-5]

Normalization

Normalized Discovered Version:

$$y = a_0^* + a_1^*(X)_s + a_2^*(X^2)_s + a_3^*(X^3)_s$$

$$= a_0^* + a_1^*\frac{(X-\mu_1)}{\sigma_1} + a_2^*\frac{(X^2-\mu_2)}{\sigma_2} + a_3^*\frac{(X^3-\mu_3)}{\sigma_3}$$

$$= (a_0^* - \sum_{i=0}^{3}\frac{a_i^*\mu_i}{\sigma_i}) + \frac{a_1^*}{\sigma_1}X + \frac{a_2^*}{\sigma_2}X^2 + \frac{a_3^*}{\sigma_3}X^3$$

$$a_0 = a_0^* - \sum_{i=0}^{3}\frac{a_i^*\mu_i}{\sigma_i}, \qquad a_i = \frac{a_i^*}{\sigma_i}$$

- Helps Stability

4.

Results & Analysis

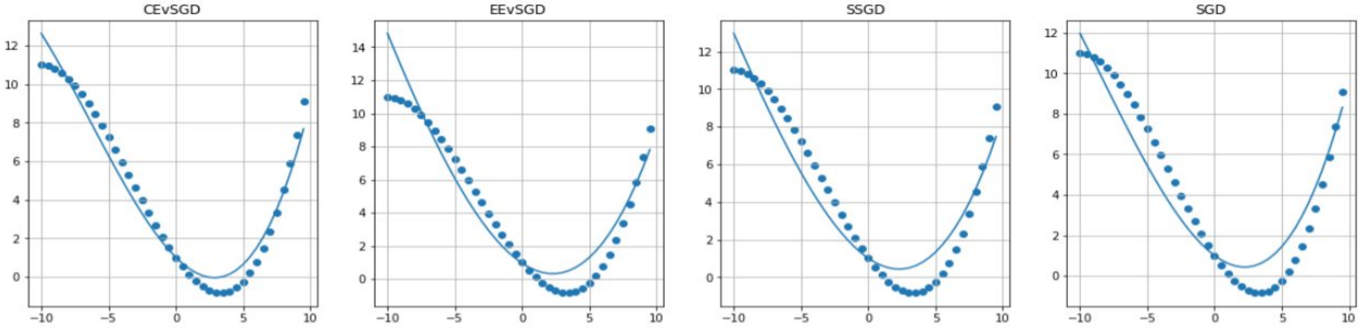


Figure 4: Models regressed over dataset at 30k steps (above)

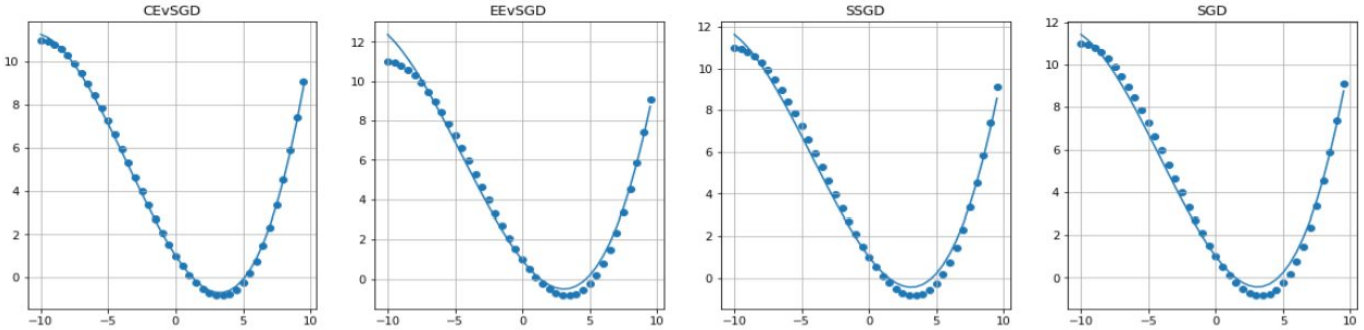
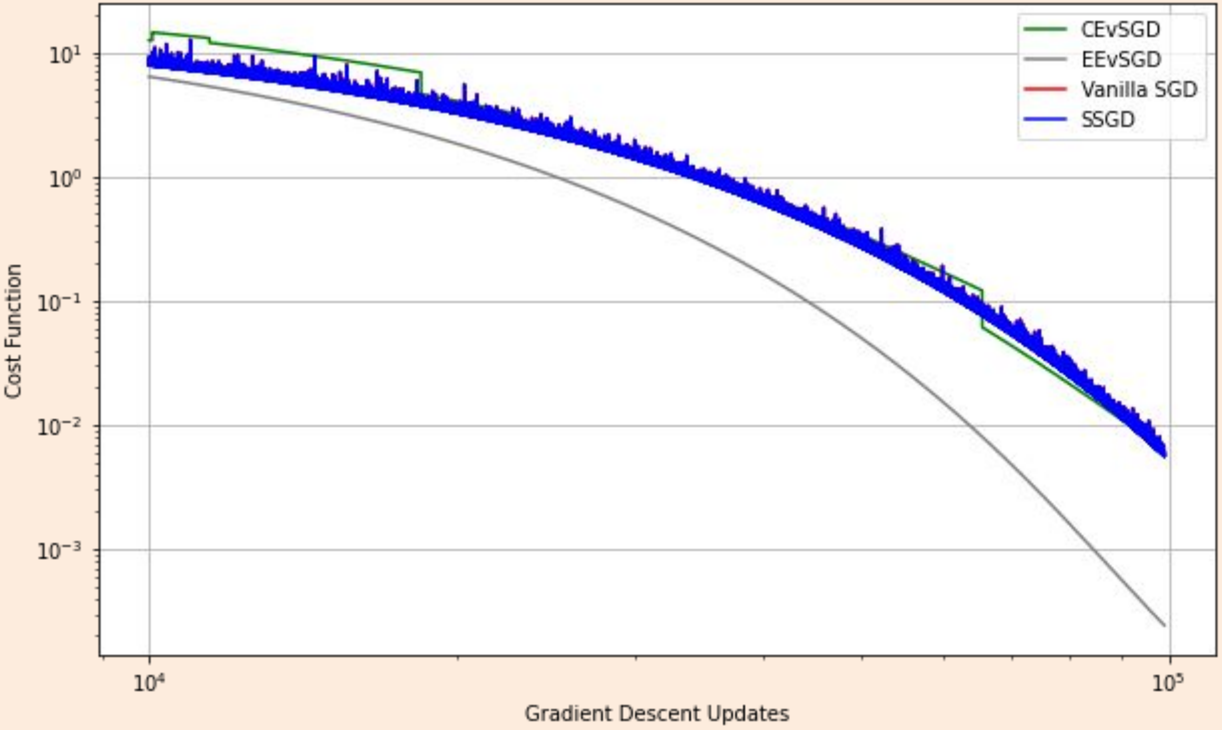


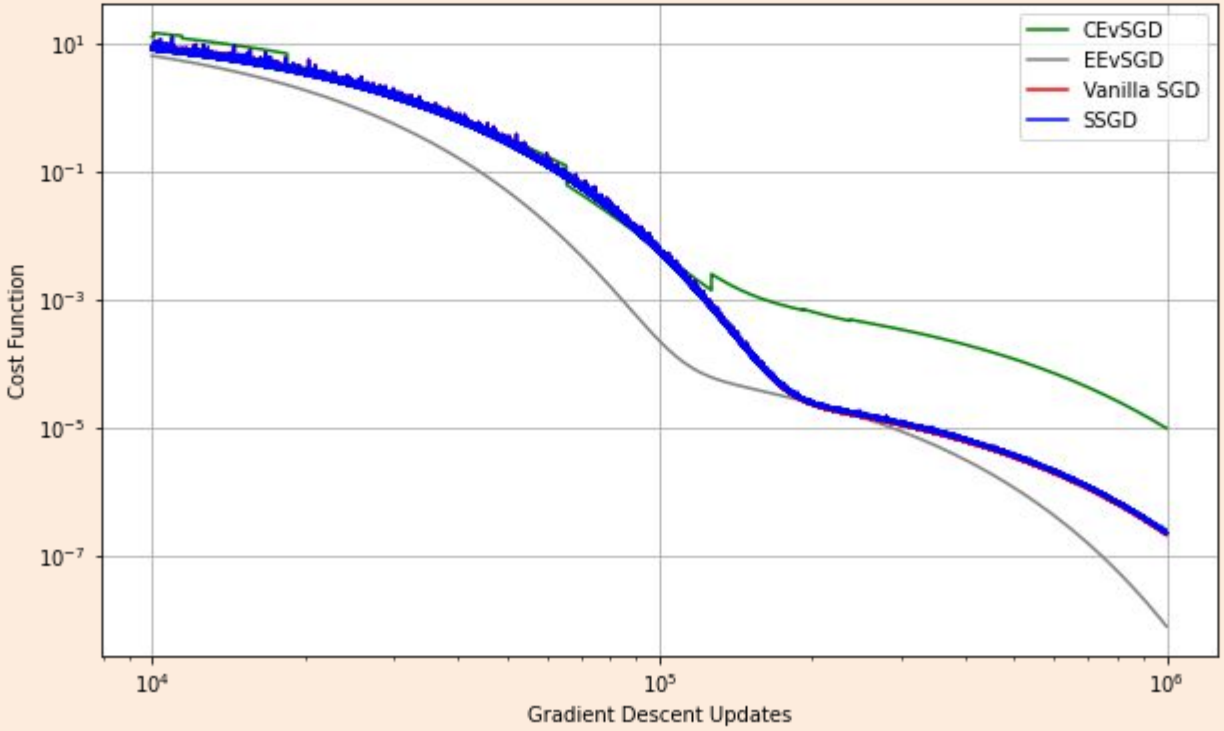
Figure 5: Models regressed over dataset at 60k steps (above)

5 Avg. Runs



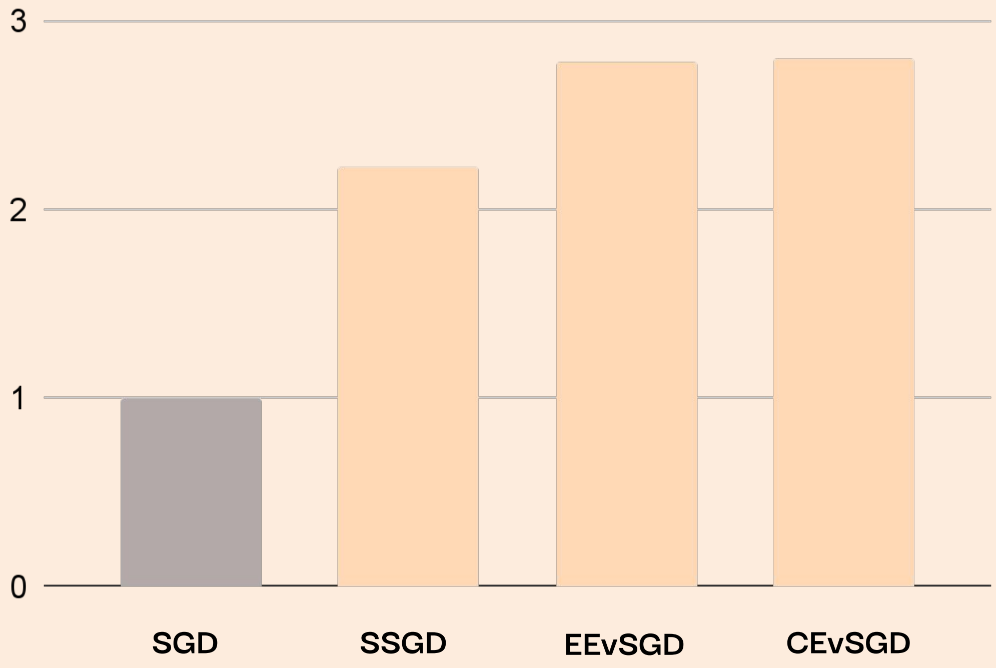
CEvSGD
EEvSGD
Vanilla SGD
SSGD
10^{1}
10^{0}
10^{-1}
10^{-2}
10^{-3}
Cost Function
10^{4}
10^{5}
Gradient Descent Updates

5 Avg. Runs



- Higher Accuracy

Time Complexity (n runs)



~100x Accuracy

High Accuracy:long VS fast:less accuracy

Stability

- CEvSGD ~ 50%
- EEvSGD ~ 80%
- SGD/ SSGD > 95%

5.

Conclusion & Future

Issues

- Normalization
- Stability Considerations
-

Success

- CEvSGD most robust
  - Even if unstable
  - Restart
- EEvSGD most efficient and accurate
  - 100x accurate
- SSGD performs better per iteration

Speedups

- Use different norm
- Parallelization
- Find and remove redundant calculations
- Better sort or remove if possible

Future

- Non-Linear
- Parallelization
- More types of Parameter Tuning
- Fitness Criteria

References

[1] William M Spears. "Crossover or mutation?" In: *Foundations of genetic algorithms*. Vol. 2. Elsevier, 1993, pp. 221–237.

[2] David E Goldberg. "Genetic and evolutionary algorithms come of age". In: *Communications of the ACM* 37.3 (1994), pp. 113–120.

[3] Panos Toulis and Edoardo M Airoldi. "Implicit stochastic gradient descent for principled estimation with large datasets". In: *ArXiv e-prints* (2014).

[4] Xiaodong Cui et al. "Evolutionary stochastic gradient descent for optimization of deep neural networks". In: *arXiv preprint arXiv:1810.06773* (2018).

Thank you.