

# SGD with Large Step Sizes Learns Sparse Features

## Seminar Optimization

Popović Milutin

Supervisor: Radu Ioan Bot

31. October 2023

- Objective is to minimize functions of the form

$$f(x) = \frac{1}{n} \sum_{i=1}^n f_i(x)$$

- Training Data:

$$\{(x_1, y_1), \dots, (x_n, y_n)\} \in \mathbb{R}^d \times \mathcal{Y}$$

- In large-scale ML: large dimension  $d$   
and large number of training data  $n$ .

- Training Data:

$$\{(x_1, y_1), \dots, (x_n, y_n)\} \in \mathbb{R}^d \times \mathcal{Y}$$

- In large-scale ML: large dimension  $d$   
and large number of training data  $n$ .

Classical examples of fitting the data via minimizing:

- Least Squares

$$\frac{1}{n} \|Ax - b\|_2^2 = \frac{1}{n} \sum_{i=1}^n (a_i^T x - b_i)^2$$

- Support Vector Machine (SVM):

$$\frac{1}{2} \|x\|_2^2 + \frac{C}{n} \sum_i^n \max(0, 1 - y_i(x^T a_i + b))$$

- Deep Neural Nets

$$\frac{1}{n} \sum_i^n \text{loss}(y_i, DNN(x; a_i))$$

Classical examples of fitting the data via minimizing:

- Least Squares

$$\frac{1}{n} \|Ax - b\|_2^2 = \frac{1}{n} \sum_{i=1}^n (a_i^T x - b_i)^2$$

- Support Vector Machine (SVM):

$$\frac{1}{2} \|x\|_2^2 + \frac{C}{n} \sum_i^n \max(0, 1 - y_i(x^T a_i + b))$$

- Deep Neural Nets

$$\frac{1}{n} \sum_i^n \text{loss}(y_i, \text{DNN}(x; a_i))$$

Classical examples of fitting the data via minimizing:

- Least Squares

$$\frac{1}{n} \|Ax - b\|_2^2 = \frac{1}{n} \sum_{i=1}^n (a_i^T x - b_i)^2$$

- Support Vector Machine (SVM):

$$\frac{1}{2} \|x\|_2^2 + \frac{C}{n} \sum_i^n \max(0, 1 - y_i(x^T a_i + b))$$

- Deep Neural Nets

$$\frac{1}{n} \sum_i^n \text{loss}(y_i, \text{DNN}(x; a_i))$$

Classical examples of fitting the data via minimizing:

- Least Squares

$$\frac{1}{n} \|Ax - b\|_2^2 = \frac{1}{n} \sum_{i=1}^n (a_i^T x - b_i)^2$$

- Support Vector Machine (SVM):

$$\frac{1}{2} \|x\|_2^2 + \frac{C}{n} \sum_i^n \max(0, 1 - y_i(x^T a_i + b))$$

- Deep Neural Nets

$$\frac{1}{n} \sum_i^n \text{loss}(y_i, DNN(x; a_i))$$



- Common pattern:

$$f(x) = \frac{1}{n} \sum_{i=1}^n f_i(x)$$

- GD would compute the gradient of every  $f_i(x)$  to update the next iterate
- SGD picks a pseudorandom  $i(r) \in \{1, 2, \dots, n\}$
- then uses only  $\nabla f_{i(r)}(x_k)$  as its descent direction

$$x^{k+1} = x^k - t_k \nabla f_{i(r)}(x^k)$$

- Key property :  $\mathbb{E}[\nabla f_{i(r)}(x)] = \nabla f(x)$
- $\nabla f_{i(r)}(x)$  is an unbiased estimator !

- GD would compute the gradient of every  $f_i(x)$  to update the next iterate
- SGD picks a pseudorandom  $i(r) \in \{1, 2, \dots, n\}$
- then uses only  $\nabla f_{i(r)}(x_k)$  as its descent direction

$$x^{k+1} = x^k - t_k \nabla f_{i(r)}(x^k)$$

- Key property :  $\mathbb{E}[\nabla f_{i(r)}(x)] = \nabla f(x)$
- $\nabla f_{i(r)}(x)$  is an unbiased estimator !

- GD would compute the gradient of every  $f_i(x)$  to update the next iterate
- SGD picks a pseudorandom  $i(r) \in \{1, 2, \dots, n\}$
- then uses only  $\nabla f_{i(r)}(x_k)$  as its descent direction

$$x^{k+1} = x^k - t_k \nabla f_{i(r)}(x^k)$$

- Key property :  $\mathbb{E}[\nabla f_{i(r)}(x)] = \nabla f(x)$
- $\nabla f_{i(r)}(x)$  is an unbiased estimator !

- GD would compute the gradient of every  $f_i(x)$  to update the next iterate
- SGD picks a pseudorandom  $i(r) \in \{1, 2, \dots, n\}$
- then uses only  $\nabla f_{i(r)}(x_k)$  as its descent direction

$$x^{k+1} = x^k - t_k \nabla f_{i(r)}(x^k)$$

- Key property :  $\mathbb{E}[\nabla f_{i(r)}(x)] = \nabla f(x)$
- $\nabla f_{i(r)}(x)$  is an unbiased estimator !

- GD would compute the gradient of every  $f_i(x)$  to update the next iterate
- SGD picks a pseudorandom  $i(r) \in \{1, 2, \dots, n\}$
- then uses only  $\nabla f_{i(r)}(x_k)$  as its descent direction

$$x^{k+1} = x^k - t_k \nabla f_{i(r)}(x^k)$$

- Key property :  $\mathbb{E}[\nabla f_{i(r)}(x)] = \nabla f(x)$
- $\nabla f_{i(r)}(x)$  is an unbiased estimator !

- large step sizes -> *loss stabilization*
- the longer the larger step size is used the better the sparse representation

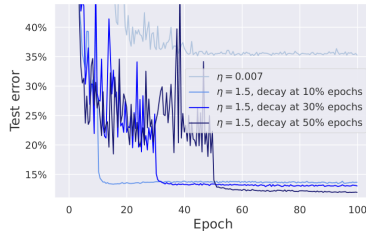
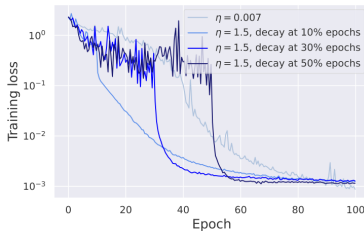


Figure: ResNet-18 (Residual Network with 18 layers) trained on CIFAR-10 (60k 32x32 images)

[1]

- large step sizes -> *loss stabilization*
- the longer the larger step size is used the better the sparse representation

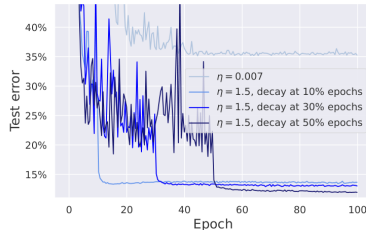
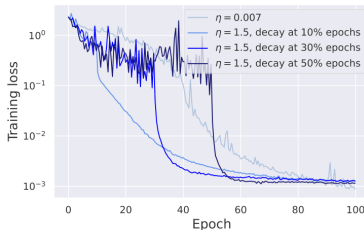


Figure: ResNet-18 (Residual Network with 18 layers) trained on CIFAR-10 (60k 32x32 images)

[1]



- [1] Maksym Andriushchenko et al. *SGD with Large Step Sizes Learns Sparse Features*. 2023. arXiv: 2210.05337 [cs.LG].

*To be continued...*

*Thank You!*