Optimization

Elena Congeduti, 18-11-2024

Announcements

Thursday, 21st of November Assignment 1 release

- 10 days to work at your submission (not graded) with deadline **Monday 2nd of December at 15:30**
- Guided peer-feedback lab **only if** you submit the deliverables in brightspace

The mean Assignment 2 grade of students who submitted Assignment 1 was significantly higher than that of students who didn’t

Lecture’s Agenda

- Bandits Review
- Neural Networks Optimization
- Learning Diagnostics
- Optimization Algorithms

Bandits Notebook Review

Learning and Optimization

One-armed bandit

Training set $y^{(1)}, y^{(2)}, y^{(3)}, \ldots \sim p_{data} = (1-p, p)$

$$1, 0, 0, 0, 0, 0, 0, 1, 1, 0$$

Learning objective: find $\hat{\lambda}$ such that $p_{model} = (1-\hat{\lambda}, \hat{\lambda}) \approx p_{data}$ (i.e. $\hat{\lambda} \approx p$)

Learning and Optimization

One-armed bandit

Training set $y^{(1)}, y^{(2)}, y^{(3)}, \ldots \sim p_{data} = (1-p, p)$

$$1, 0, 0, 0, 0, 0, 0, 1, 1, 0$$

Learning objective: find $\hat{\lambda}$ such that $p_{model} = (1-\hat{\lambda}, \hat{\lambda}) \approx p_{data}$ (i.e. $\hat{\lambda} \approx p$)

Optimization objective: find $\hat{\lambda}$ that minimizes the loss

$$L(\lambda) = \frac{1}{10}\sum_{i=1}^{10} -y^{(i)}\log(\lambda) - (1-y^{(i)})\log(1-\lambda) = -0.3\log(\lambda)) - 0.7\log(1-\lambda)$$

Training and Generalization error

One-armed bandit

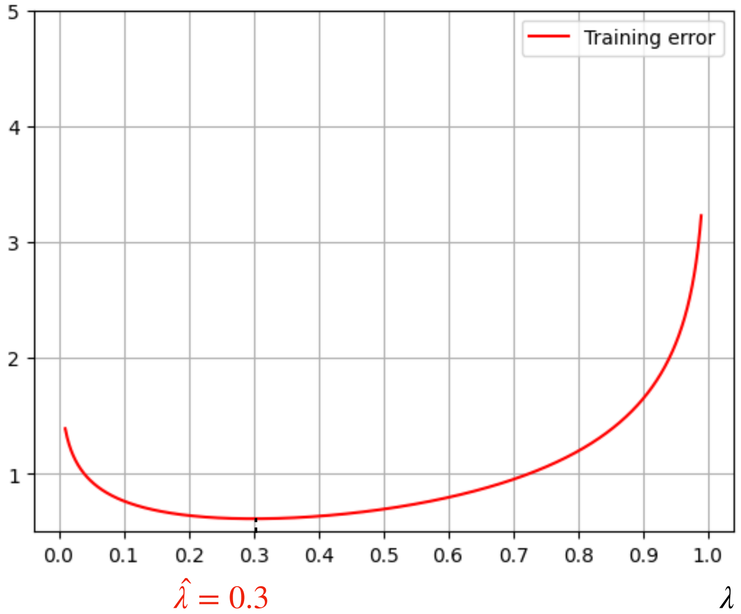
Training set $y^{(1)}, y^{(2)}, y^{(3)}, \ldots \sim p_{data} = (1-p, p)$

$$1, 0, 0, 0, 0, 0, 0, 1, 1, 0$$

Learning objective: find $\hat{\lambda}$ such that $p_{model} = (1-\hat{\lambda}, \hat{\lambda}) \approx p_{data}$ (i.e. $\hat{\lambda} \approx p$)

Optimization objective: find $\hat{\lambda}$ that minimizes the loss

Training loss $L(\lambda) = -0.3\log(\lambda)) - 0.7\log(1-\lambda)$



Training and Generalization error

One-armed bandit

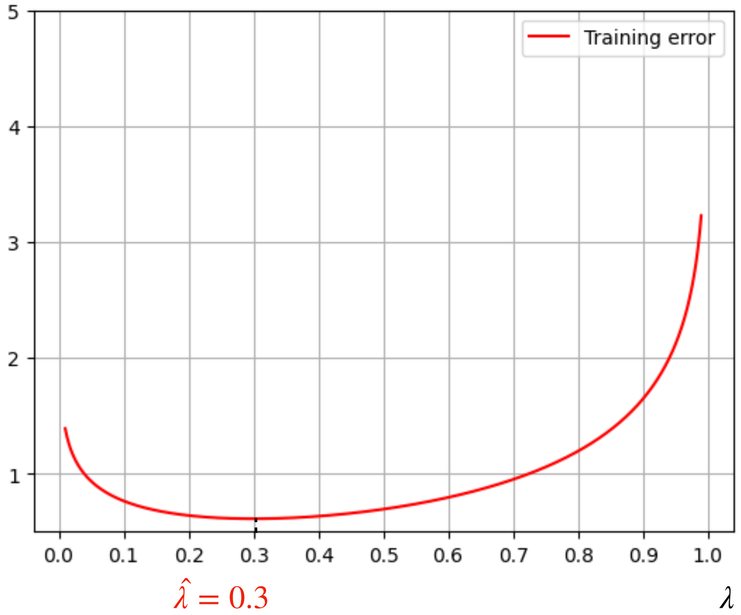
Training set $y^{(1)}, y^{(2)}, y^{(3)}, \ldots \sim p_{data} = (1-p, p)$

$1, 0, 0, 0, 0, 0, 0, 1, 1, 0$

Learning objective: find $\hat{\lambda}$ such that $p_{model} = (1-\hat{\lambda}, \hat{\lambda}) \approx p_{data}$ (i.e. $\hat{\lambda} \approx p$)

Optimization objective: find $\hat{\lambda}$ that minimizes the loss

Training loss $L(\lambda) = -0.3\log(\lambda)) - 0.7\log(1-\lambda)$



$L(\hat{\lambda}) = 0.27$ ⟶ Training error over observed data

Training and Generalization error

One-armed bandit

Training set $y^{(1)}, y^{(2)}, y^{(3)}, \ldots \sim p_{data} = (1-p, p)$

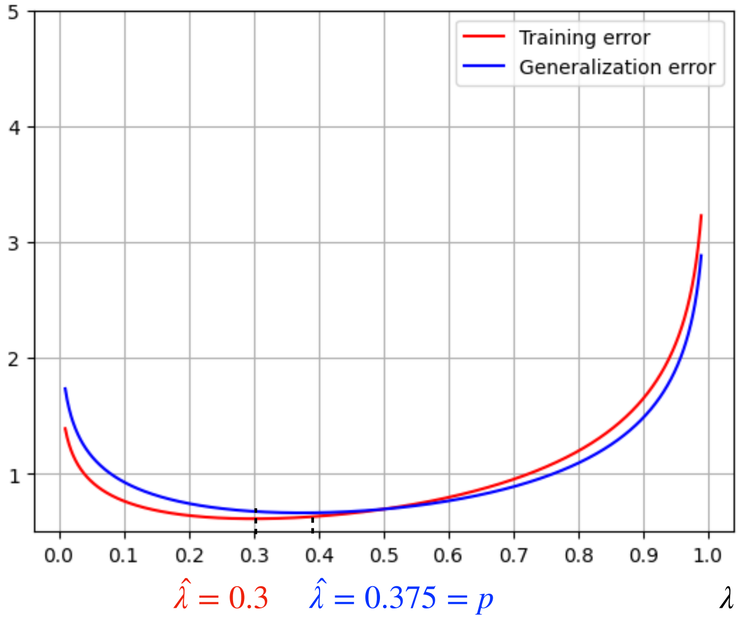
$1, 0, 0, 0, 0, 0, 0, 1, 1, 0$

Learning objective: find $\hat{\lambda}$ such that $p_{model} = (1-\hat{\lambda}, \hat{\lambda}) \approx p_{data}$ (i.e. $\hat{\lambda} \approx p$)

Generalization error $CE(p_{data}, p_{model}) = -p\log(\lambda) - (1-p)\log(1-\lambda)$

Optimization objective: find $\hat{\lambda}$ that minimizes the loss

Training loss $L(\lambda) = -0.3\log(\lambda)) - 0.7\log(1-\lambda)$



$L(\hat{\lambda}) = 0.27$ ⟶ Training error over observed data

$CE(\hat{\lambda}) = 0.29$ ⟶ Generalization error over unseen data

Training and Generalization error

One-armed bandit

Training set $y^{(1)}, y^{(2)}, y^{(3)}, \ldots \sim p_{data} = (1-p, p)$

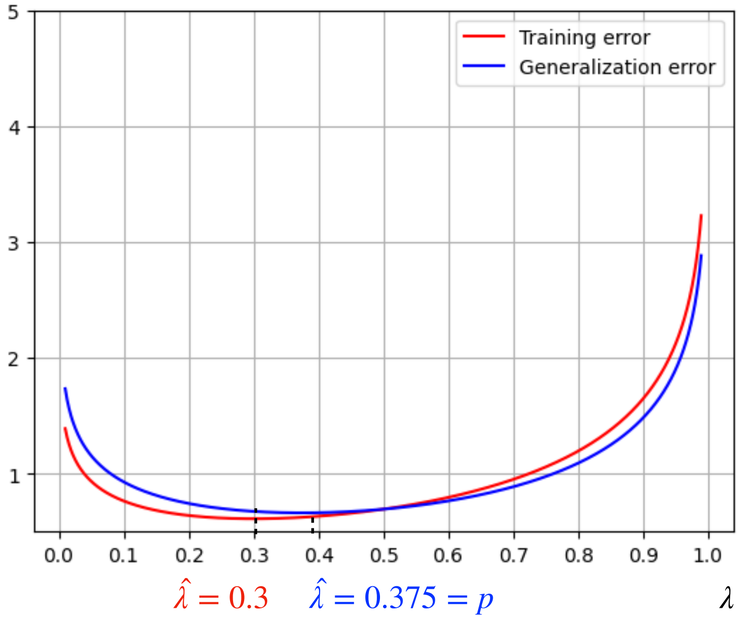
$1, 0, 0, 0, 0, 0, 0, 1, 1, 0$

Learning objective: find $\hat{\lambda}$ such that $p_{model} = (1-\hat{\lambda}, \hat{\lambda}) \approx p_{data}$ (i.e. $\hat{\lambda} \approx p$)

Generalization error $CE(p_{data}, p_{model}) = -p\log(\lambda) - (1-p)\log(1-\lambda)$

Optimization objective: find $\hat{\lambda}$ that minimizes the loss

Training loss $L(\lambda) = -0.3\log(\lambda)) - 0.7\log(1-\lambda)$



$L(\hat{\lambda}) = 0.27$ ⟶ Training error over observed data

$CE(\hat{\lambda}) = 0.29$ ⟶ Generalization error over unseen data

p_{data} is unknown, so we can't compute the generalization error (nor its derivatives)!

Training and Generalization error

One-armed bandit

Training set $y^{(1)}, y^{(2)}, y^{(3)}, \ldots \sim p_{data} = (1-p, p)$

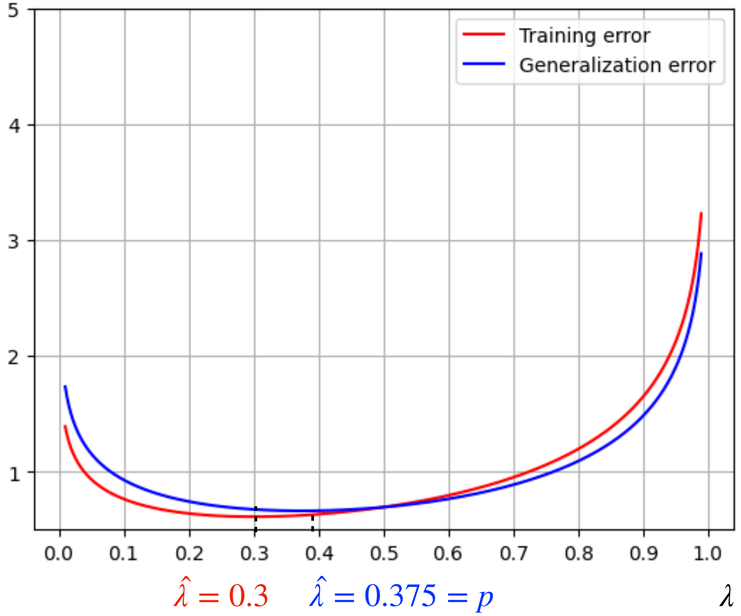
$$1, 0, 0, 0, 0, 0, 0, 1, 1, 0$$

Learning objective: find $\hat{\lambda}$ such that $p_{model} = (1-\hat{\lambda}, \hat{\lambda}) \approx p_{data}$ (i.e. $\hat{\lambda} \approx p$)

Generalization error $CE(p_{data}, p_{model}) = -p\log(\lambda) - (1-p)\log(1-\lambda)$

Optimization objective: find $\hat{\lambda}$ that minimizes the loss

Training loss $L(\lambda) = -0.3\log(\lambda)) - 0.7\log(1-\lambda)$



$L(\hat{\lambda}) = 0.27$ ⟶ Training error over observed data

$CE(\hat{\lambda}) = 0.29$ ⟶ Generalization error over unseen data

Train minimizing the training loss but we estimate the generalization error with an independent test set $y^{(j)} \sim p_{data}$

Test error $\frac{1}{n_{test}} \sum_{j=1}^{n_{test}} -y^{(j)}\log(\lambda) - (1-y^{(j)})\log(1-\lambda)$

Training and Generalization error

One-armed bandit

Training set $y^{(1)}, y^{(2)}, y^{(3)}, \ldots \sim p_{data} = (1-p, p)$

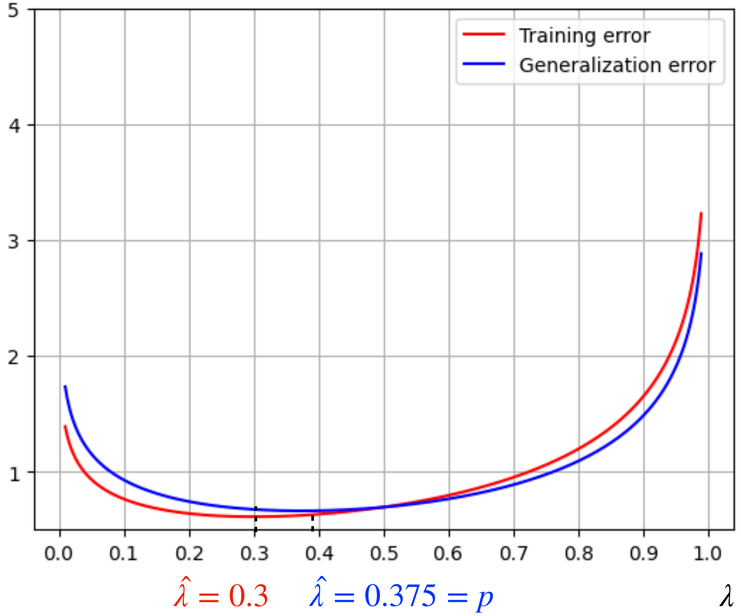
$1, 0, 0, 0, 0, 0, 0, 1, 1, 0$

Learning objective: find $\hat{\lambda}$ such that $p_{model} = (1-\hat{\lambda}, \hat{\lambda}) \approx p_{data}$ (i.e. $\hat{\lambda} \approx p$)

Generalization error $CE(p_{data}, p_{model}) = -p\log(\lambda) - (1-p)\log(1-\lambda)$

Optimization objective: find $\hat{\lambda}$ that minimizes the loss

Training loss $L(\lambda) = -0.3\log(\lambda)) - 0.7\log(1-\lambda)$



$L(\hat{\lambda}) = 0.27$ ⟶ Training error over observed data

$CE(\hat{\lambda}) = 0.29$ ⟶ Generalization error over unseen data

Problem: the two objectives might be not aligned!
(Overfitting)

Neural Network Optimization

Quiz: true or false?

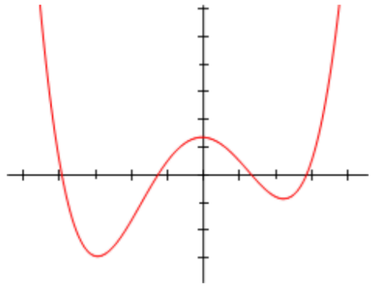
Loss functions are generally convex and have therefore only a single (global) minimum

Neural Network Optimization

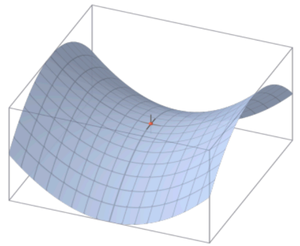
Quiz: true or false?

~~Loss functions are generally convex and have therefore only a single (global) minimum~~

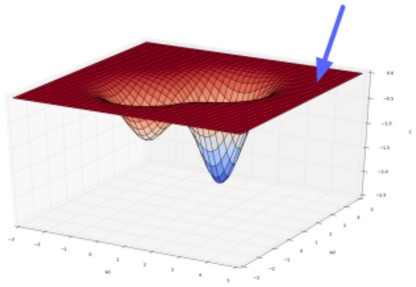
Loss functions are generally non-convex and may have a large number of local minima, saddle points and flat regions*



local minimum



saddle point



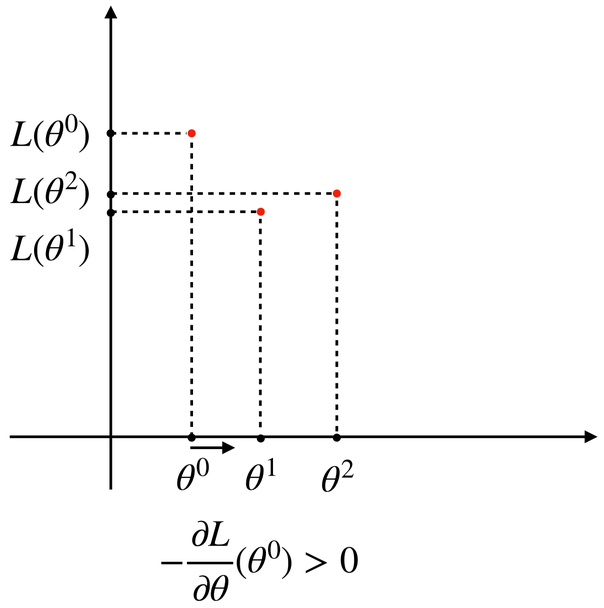
flat region

*What is an example of a convex loss function? And loss function with flat region?

Neural Network Optimization

Quiz: true or false?

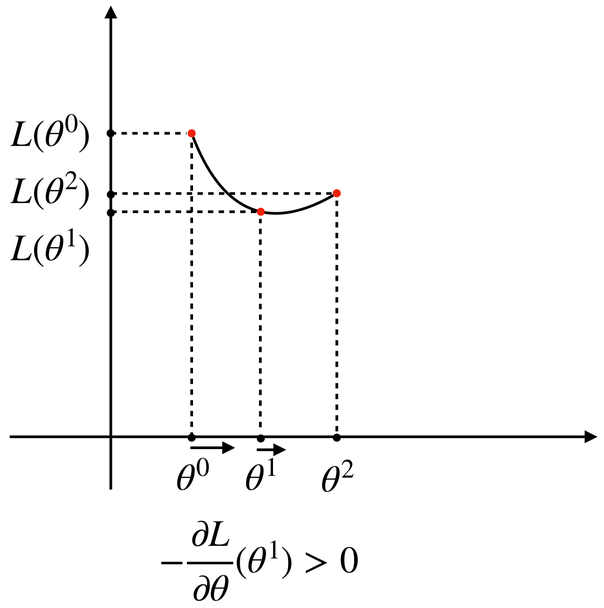
During gradient descent, it is possible to observe an increase in the training loss after a single update



Neural Network Optimization

Quiz: true or false?

During gradient descent, it is possible to observe an increase in the training loss after a single update

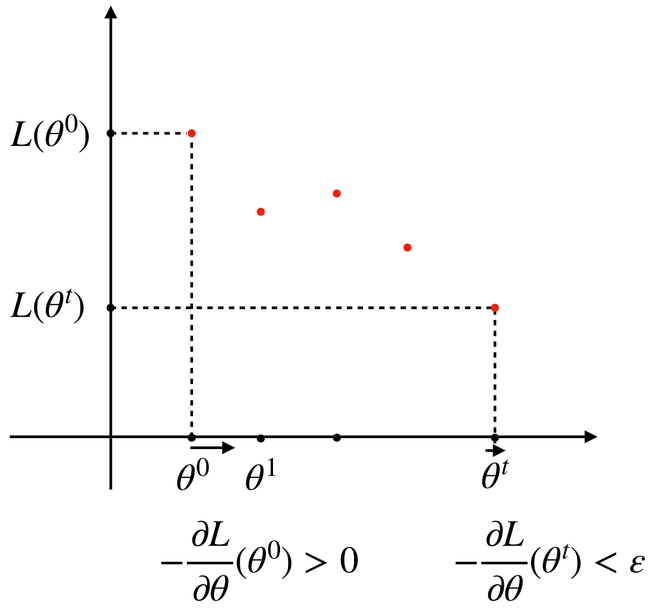


One possible scenario but there are more…

Neural Network Optimization

Quiz: true or false?

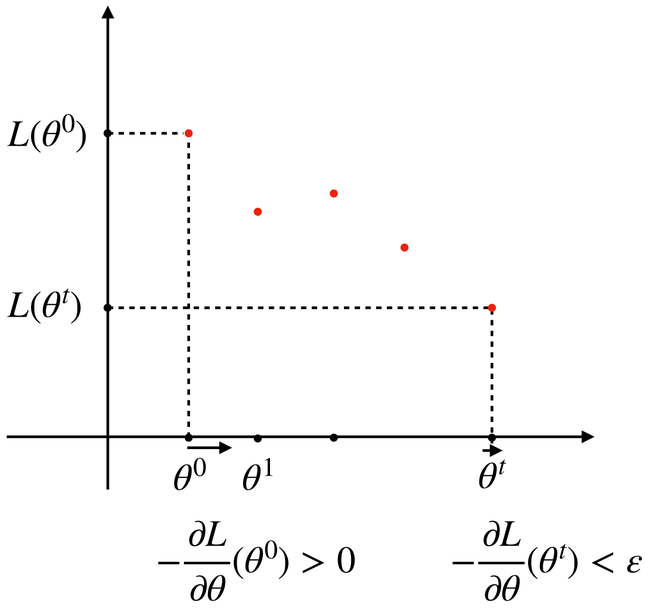
After some iteration of gradient descent, the gradient is very small that implies we are sufficiently close to the global minimum



Neural Network Optimization

Quiz: true or false?

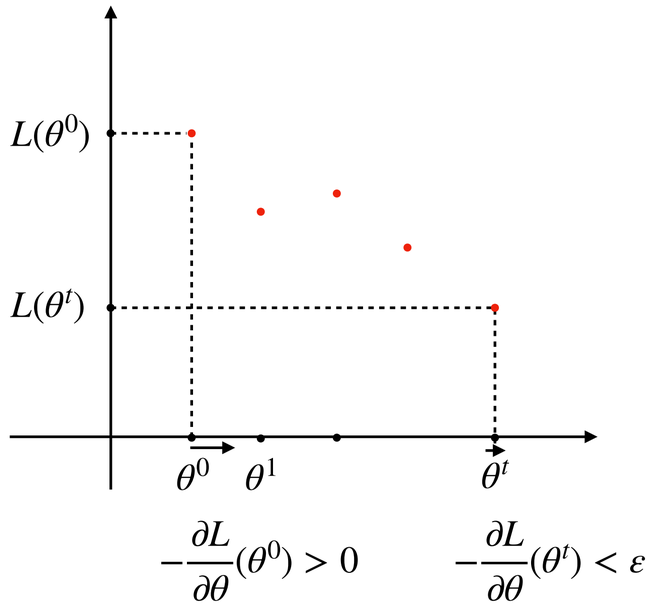
After some iteration of gradient descent, the gradient is very small that implies we are sufficiently close to the global minimum



Neural Network Optimization

Quiz: true or false?

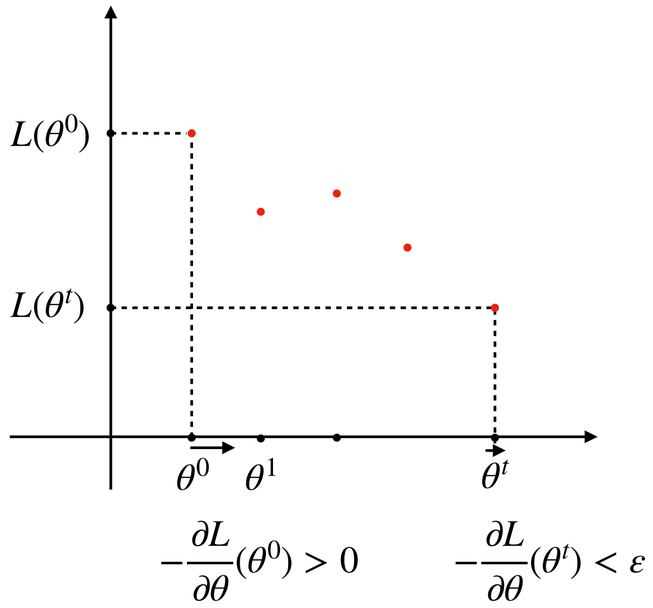
After some iteration of gradient descent, the gradient is very small that implies we are sufficiently close to a local minimum



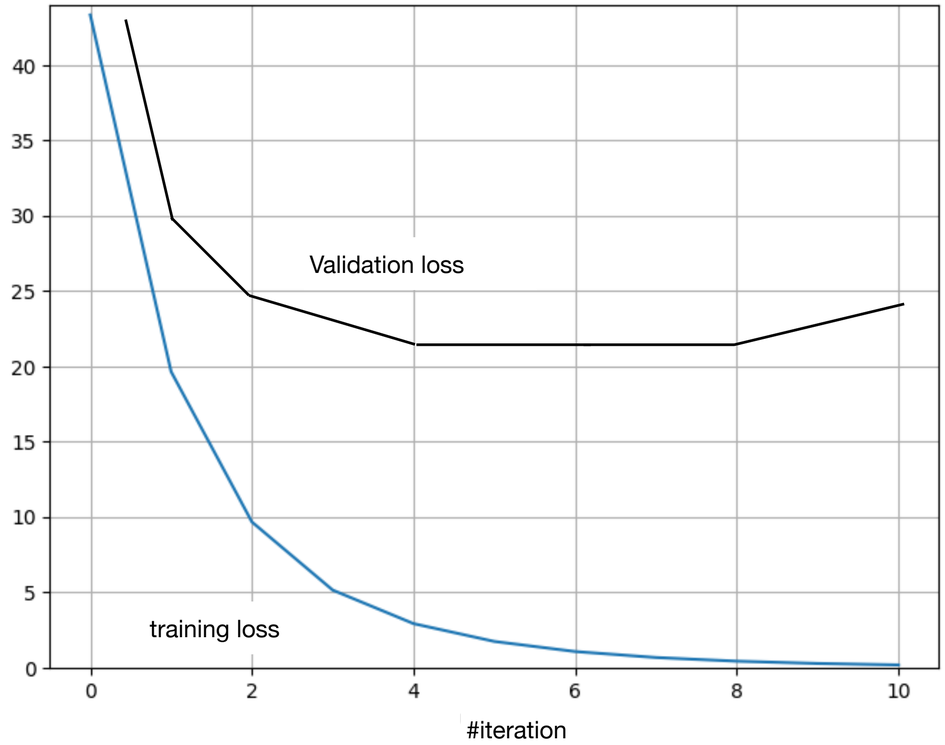
Neural Network Optimization

Quiz: true or false?

After some iteration of gradient descent, the gradient is very small that implies we are sufficiently close to a local minimum

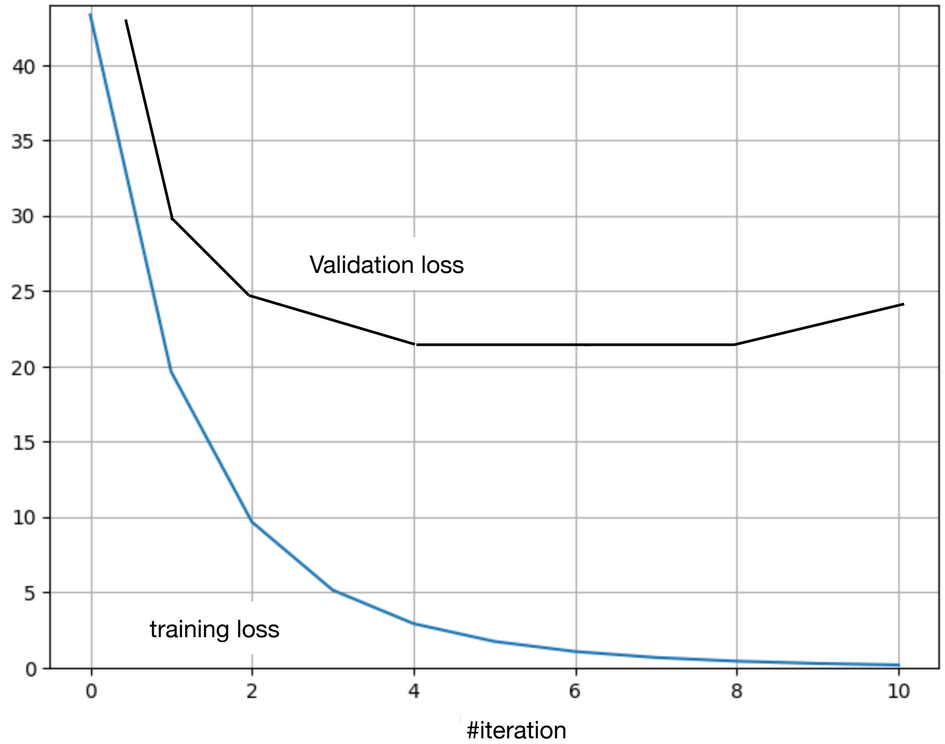


Training Curves



- Training loss and validation loss computed over iterations of gradient descent or epochs

Training Curves

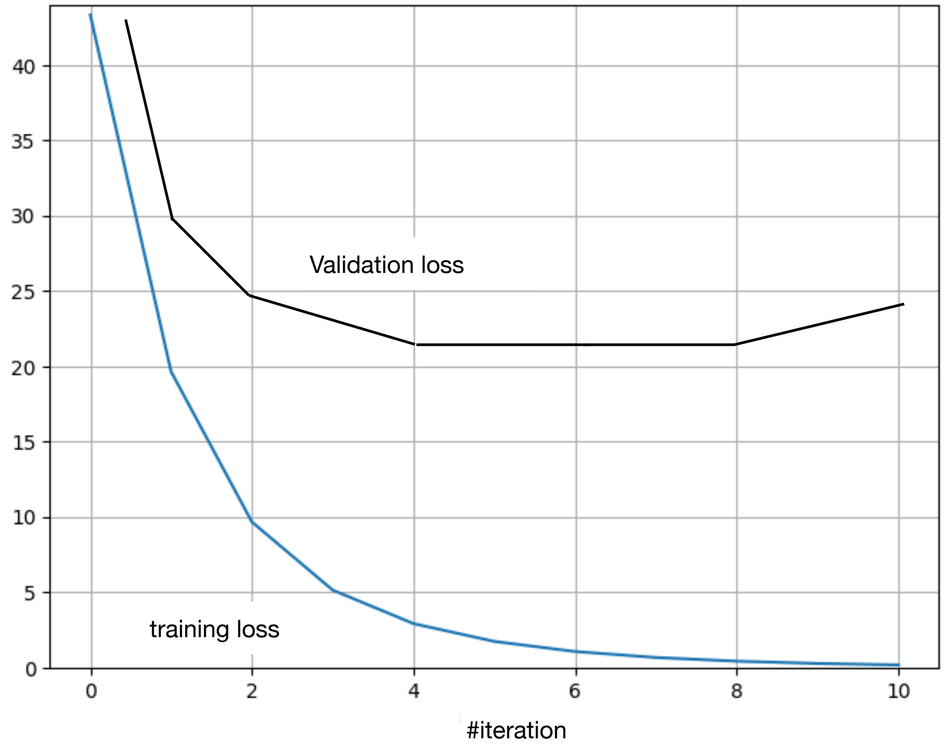


- Training loss and validation loss computed over iterations of gradient descent or epochs

↓

one complete pass of the entire training set

Training Curves

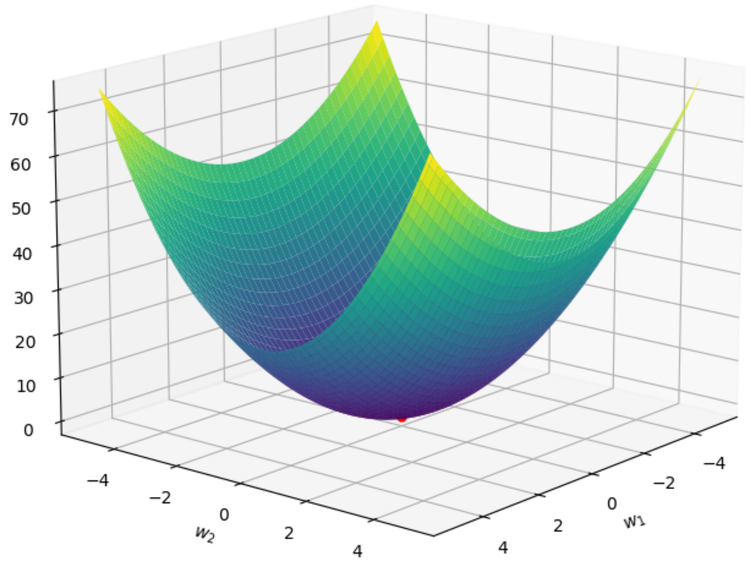


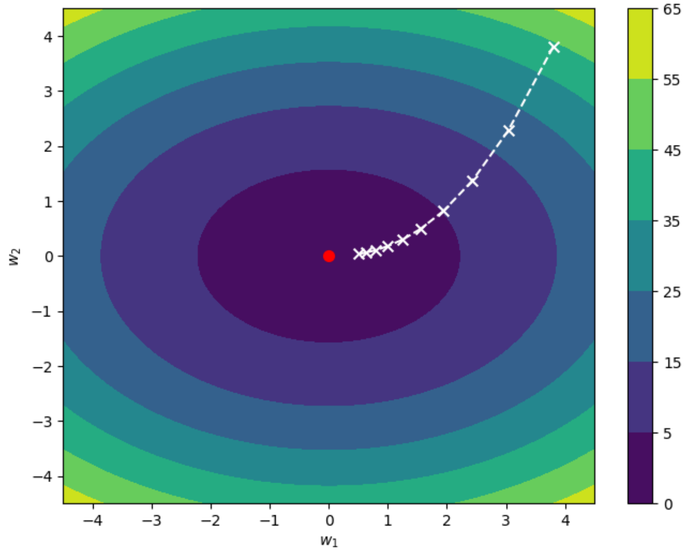
- Training loss and validation loss computed over iterations of gradient descent or epochs
- Optimization diagnostics: hard to tell from the training curves whether gradient descent has converged but they can reveal major problems.

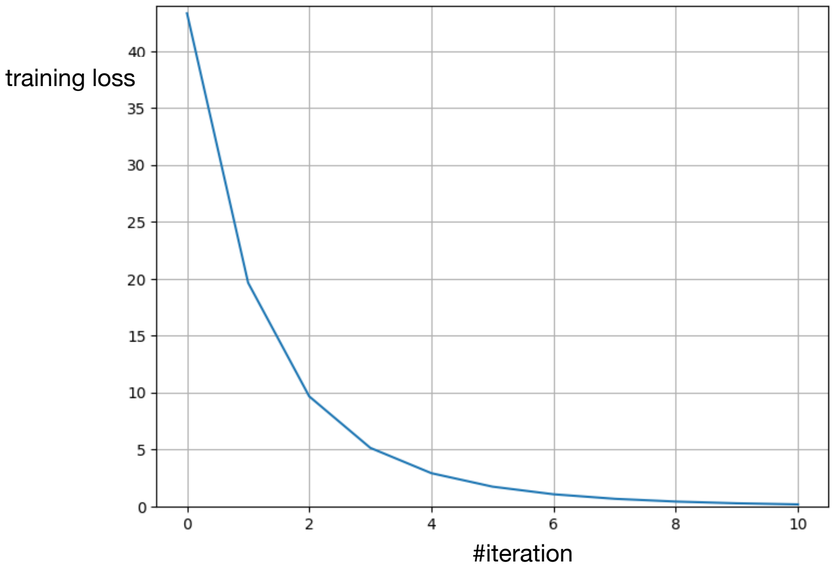
Learning Rate and Training Loss

Weights update: $w \longleftarrow w - \alpha \frac{\partial L}{\partial w}$

L





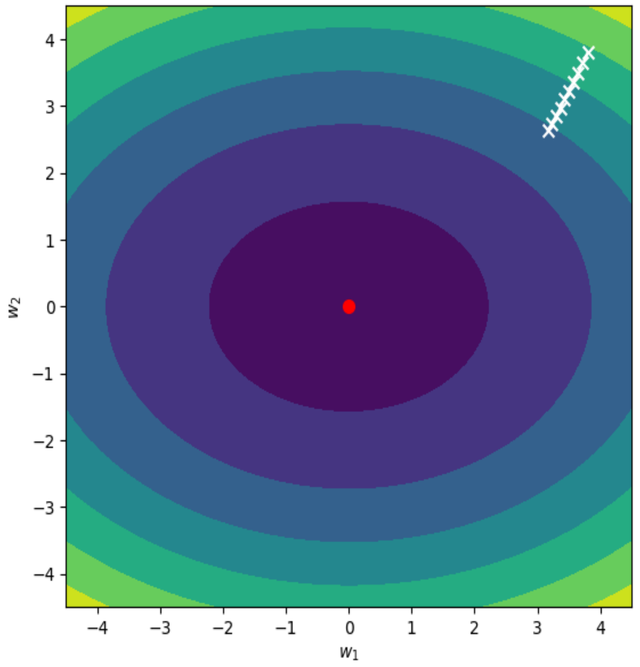


$\alpha = 0.1$

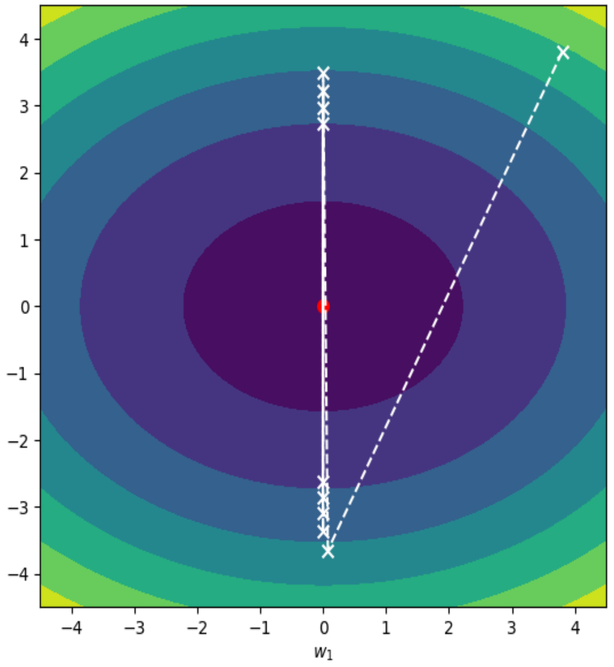
The learning rate α is arguably the most important hyperparameter to tune

Learning Rate Diagnostics

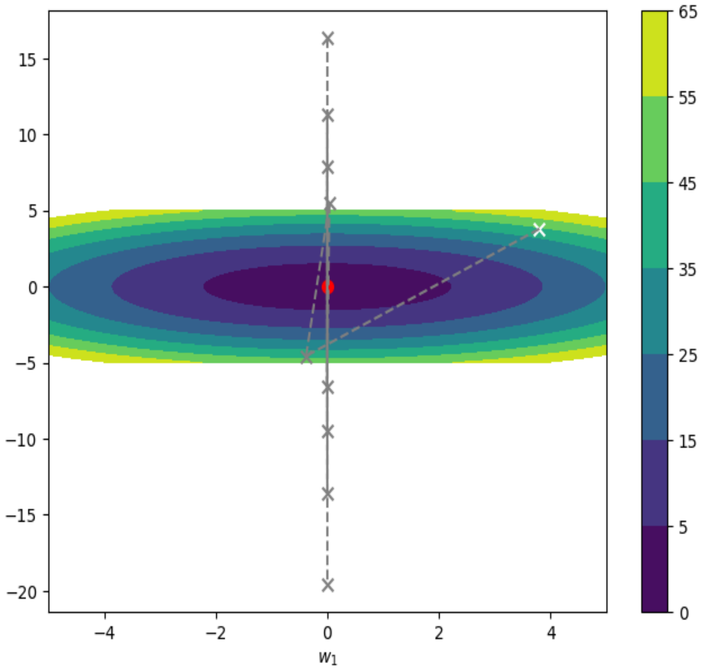
Weights update: $w \longleftarrow w - \alpha \frac{\partial L}{\partial w}$



Slow progress



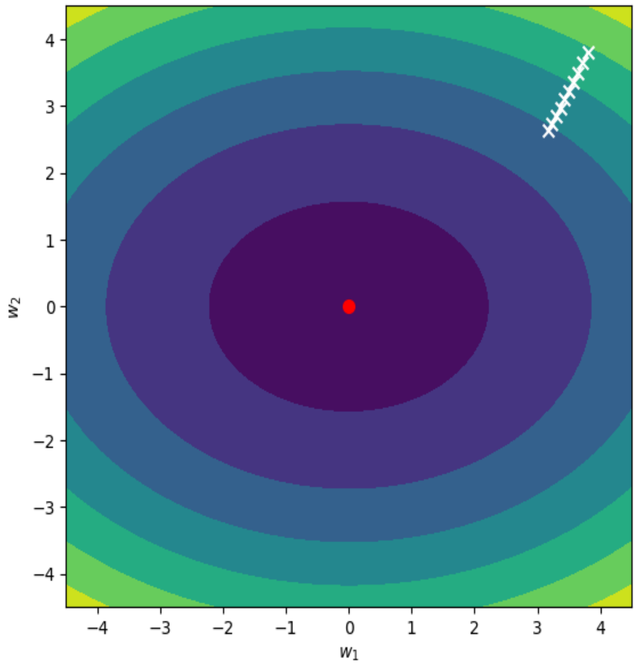
Oscillations



Instability

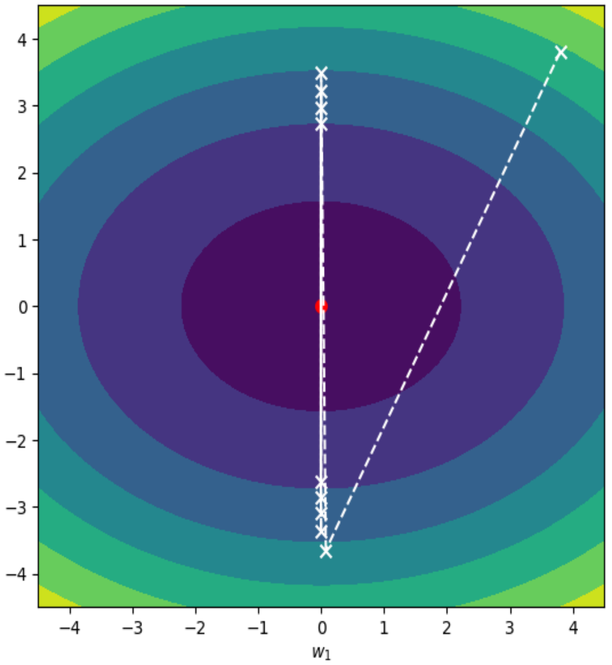
Learning Rate Diagnostics

Weights update: $w \longleftarrow w - \alpha \frac{\partial L}{\partial w}$



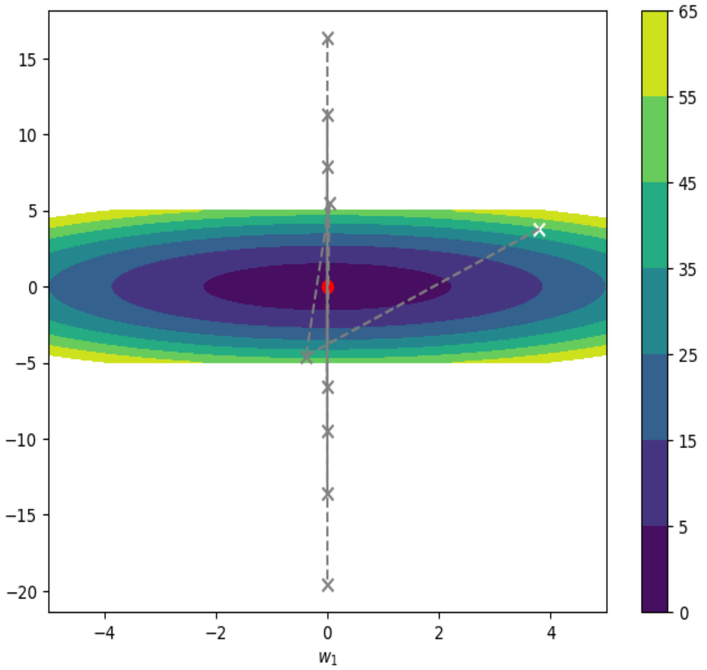
Slow progress

$\alpha = 0.01$ too small



Oscillations

$\alpha = 0.48$ too large

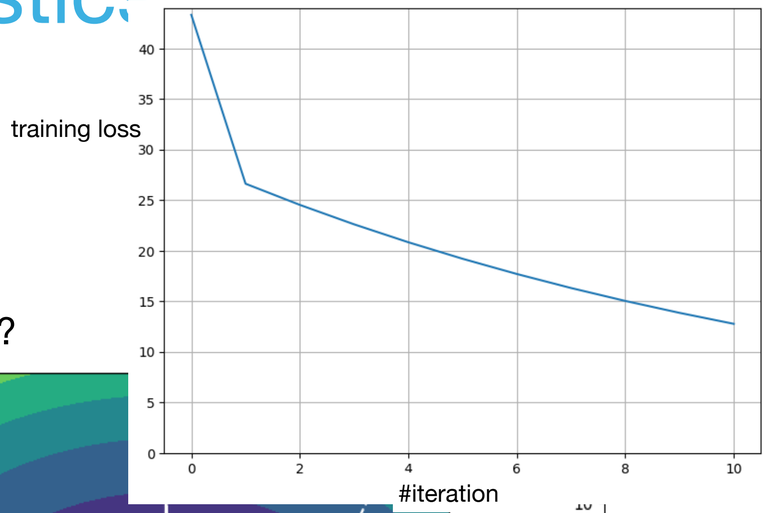


Instability

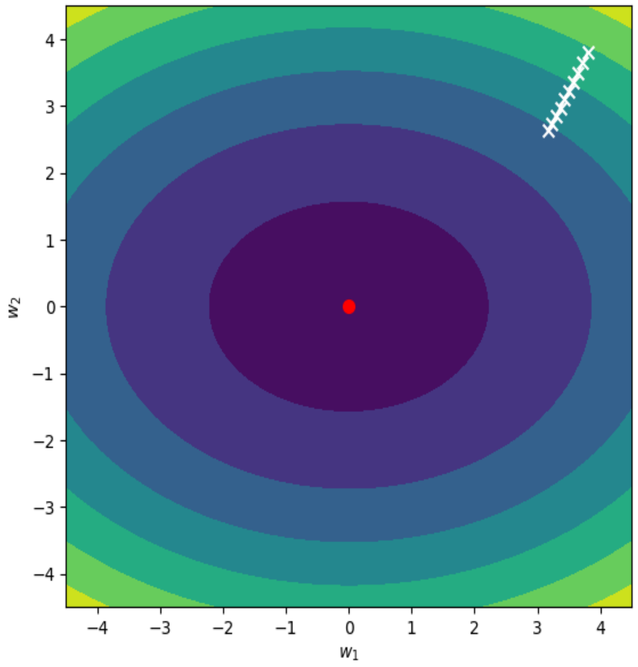
$\alpha = 0.51$ much too large

Learning Rate Diagnostics

Weights update: $w \longleftarrow w - \alpha \frac{\partial L}{\partial w}$

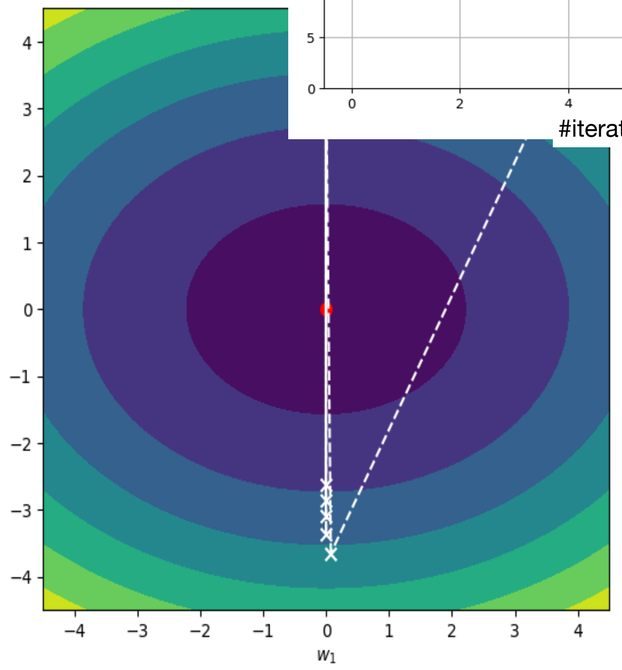


Which scenario does the training curve represent?



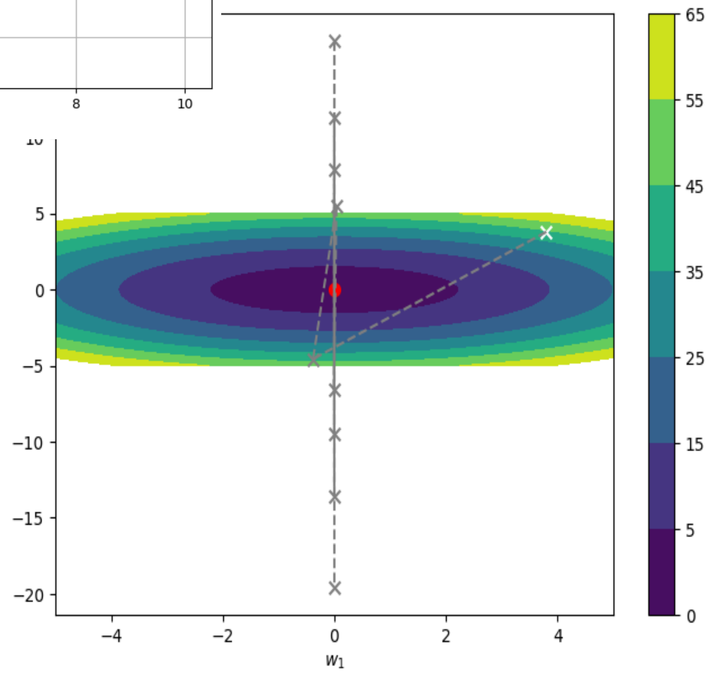
Slow progress

$\alpha = 0.01$ too small



Oscillations

$\alpha = 0.48$ too large



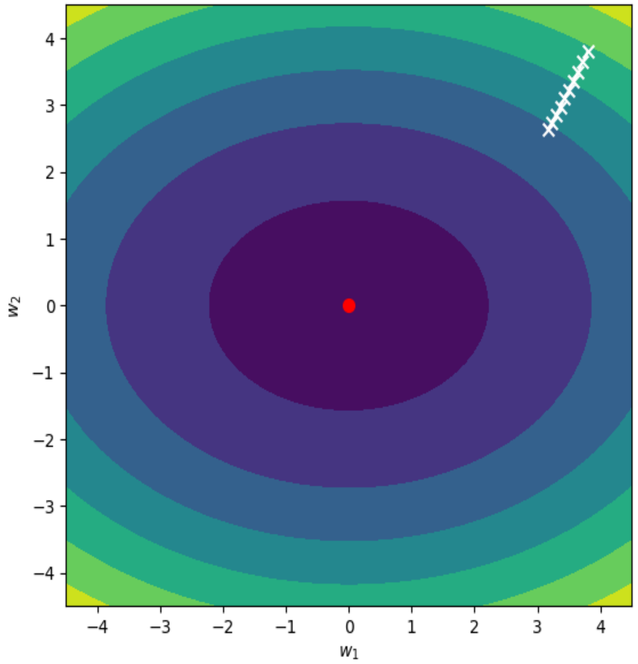
Instability

$\alpha = 0.51$ much too large

Learning Rate Diagnostics

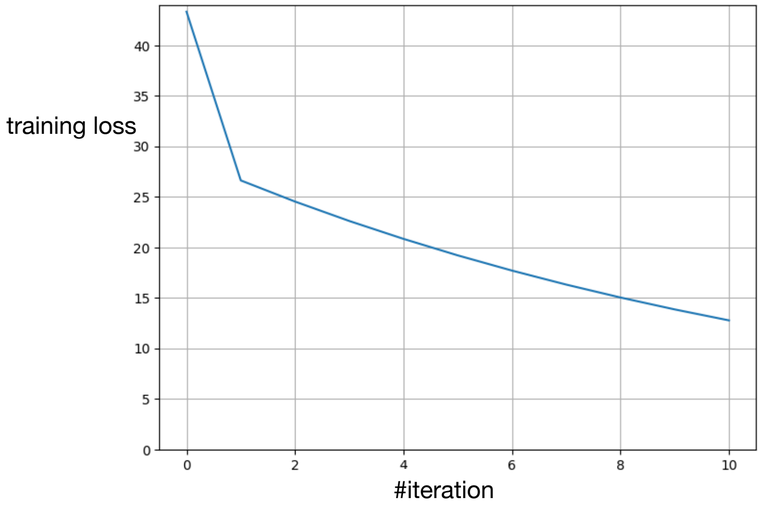
Weights update: $w \longleftarrow w - \alpha \frac{\partial L}{\partial w}$

Which scenario does the training curve represent?



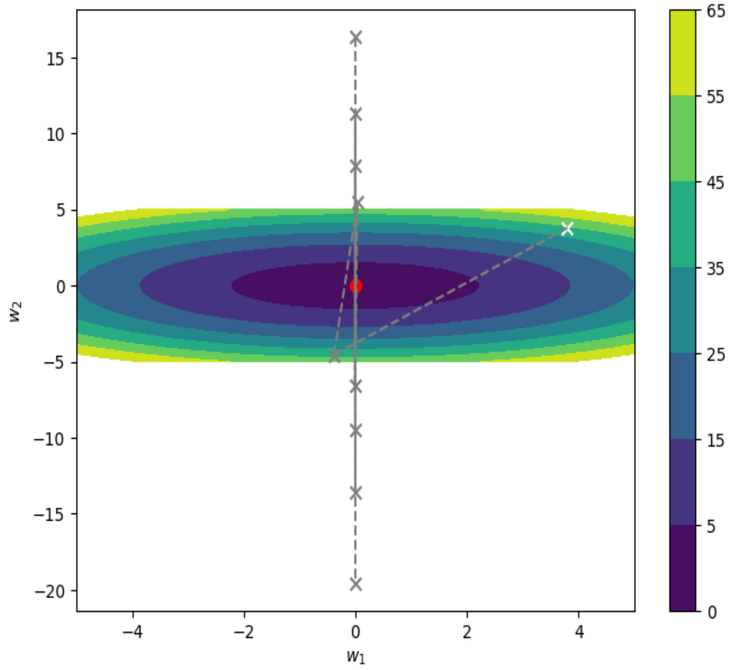
Slow progress

$\alpha = 0.01$ too small



Oscillations

$\alpha = 0.48$ too large



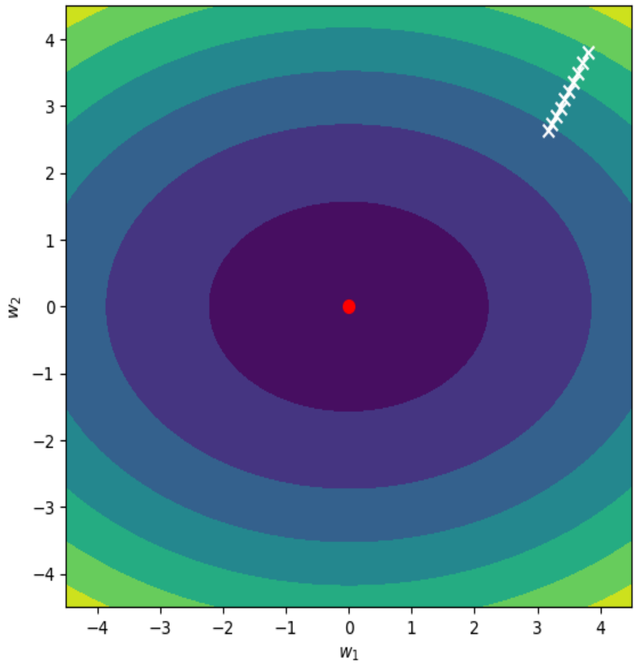
Instability

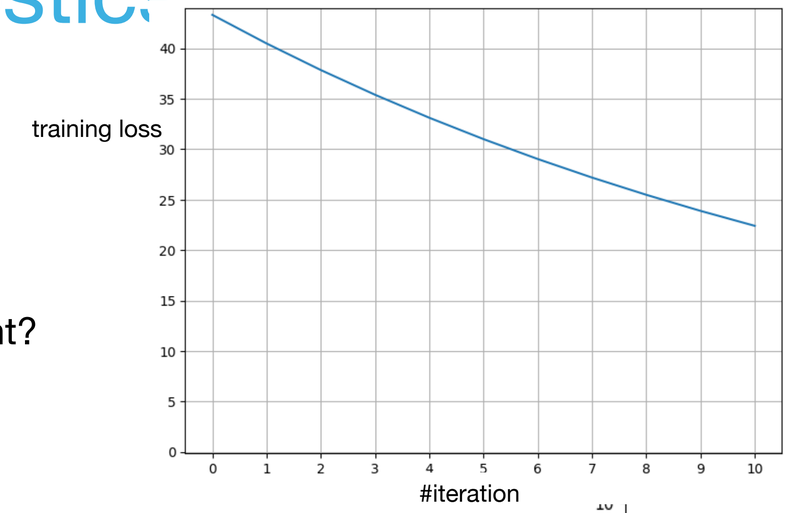
$\alpha = 0.51$ much too large

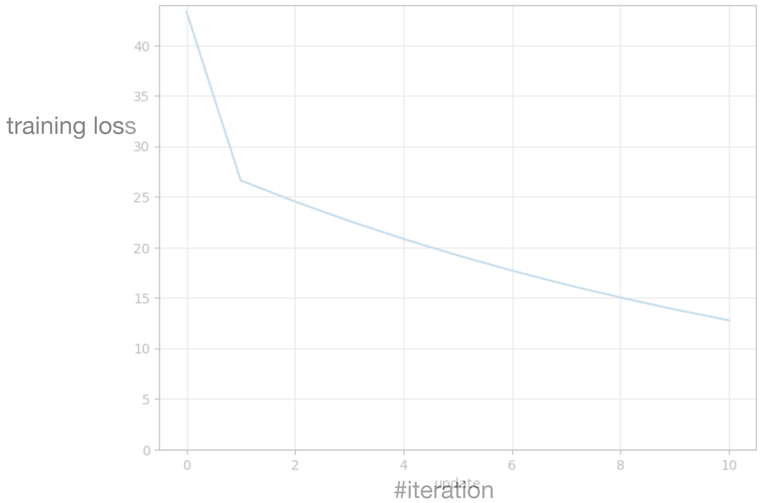
Learning Rate Diagnostics

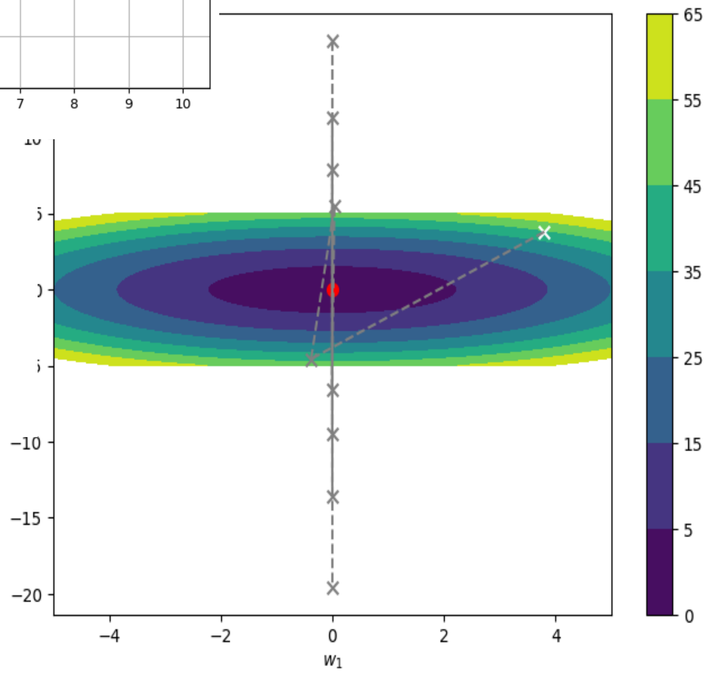
Weights update: $w \longleftarrow w - \alpha \frac{\partial L}{\partial w}$

Which scenario does the training curve represent?









Slow progress

$\alpha = 0.01$ too small

Oscillations

$\alpha = 0.48$ too large

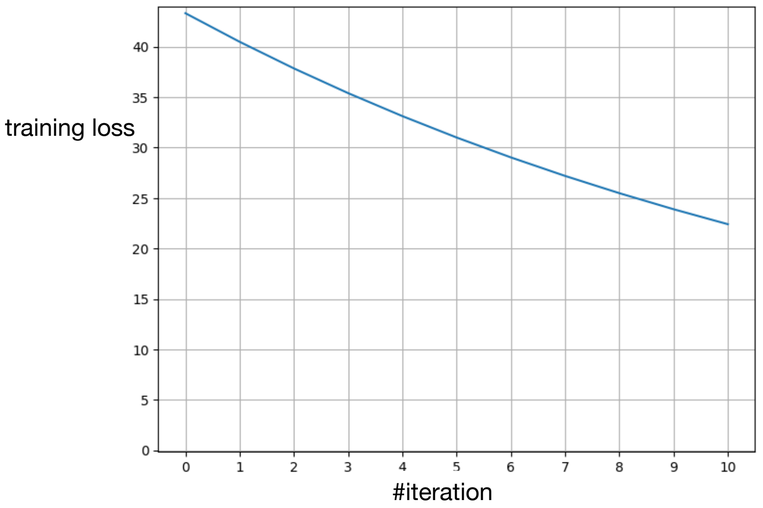
Instability

$\alpha = 0.51$ much too large

Learning Rate Diagnostics

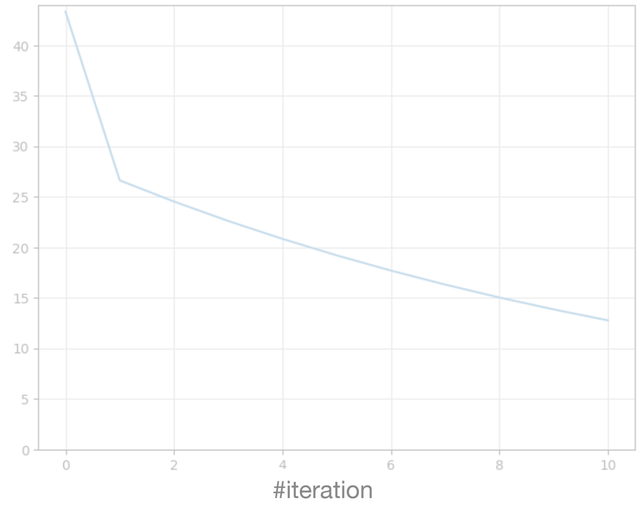
Weights update: $w \longleftarrow w - \alpha \frac{\partial L}{\partial w}$

Which scenario does the training curve represent?



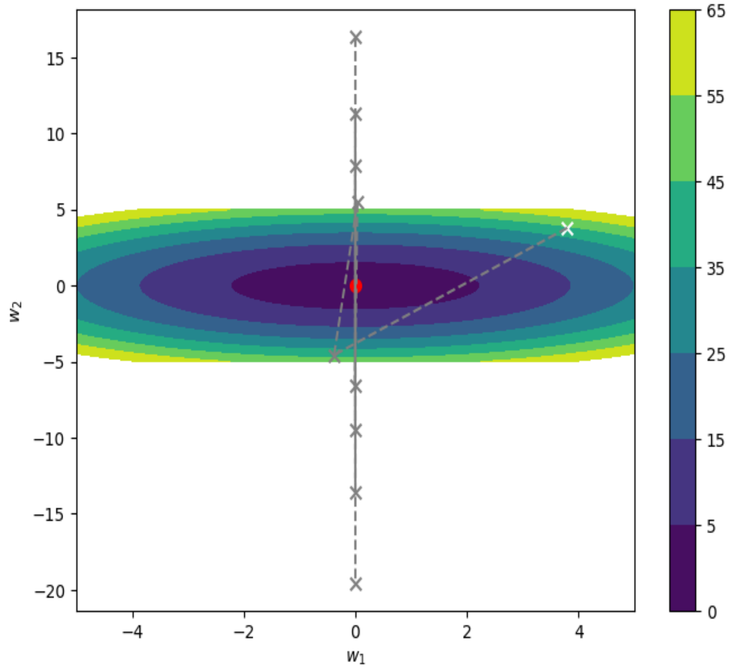
Slow progress

$\alpha = 0.01$ too small



Oscillations

$\alpha = 0.48$ too large



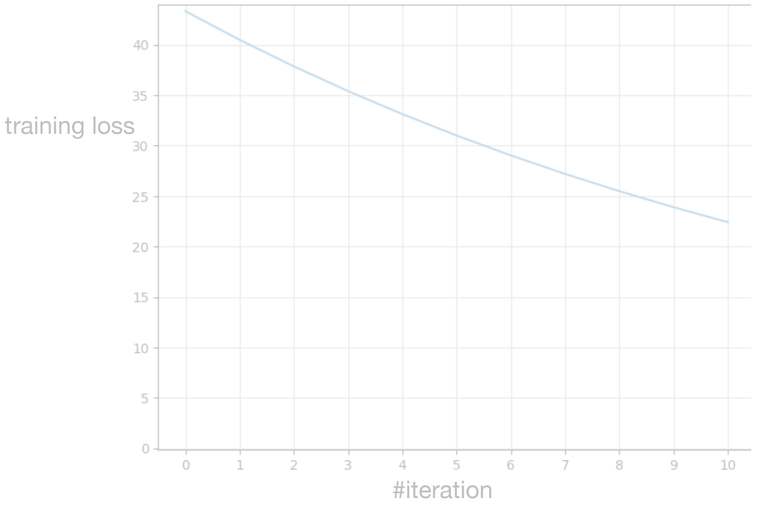
Instability

$\alpha = 0.51$ much too large

Learning Rate Diagnostics

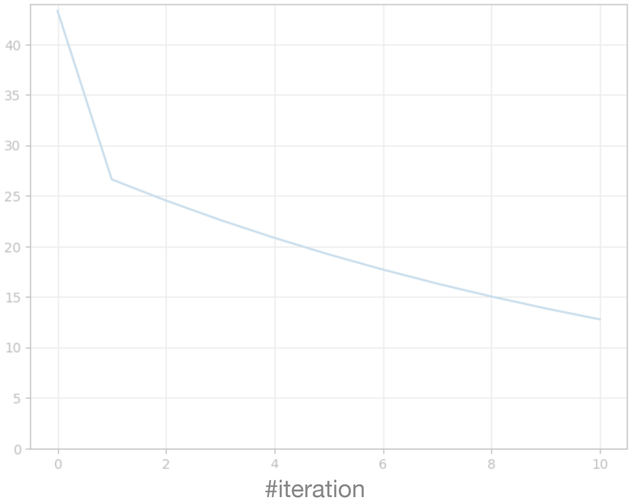
Weights update: $w \longleftarrow w - \alpha \frac{\partial L}{\partial w}$

Which scenario does the training curve represent?



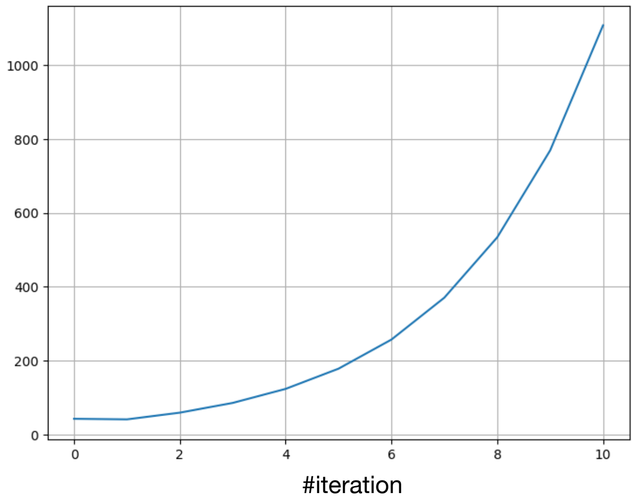
Slow progress

$\alpha = 0.01$ too small



Oscillations

$\alpha = 0.48$ too large



Instability

$\alpha = 0.51$ much too large

SGD Learning Rate

Weights update: $w \longleftarrow w - \alpha \frac{1}{m} \sum_{i=1}^{m} \frac{\partial l^{(i)}}{\partial w}$ → Approximation of $\frac{\partial L}{\partial w}$

- Randomly sample m training examples
- Compute an approximation of the actual gradient $\frac{\partial L}{\partial w}$
- Individual stochastic gradients are noisy

Stochastic Gradient Descent

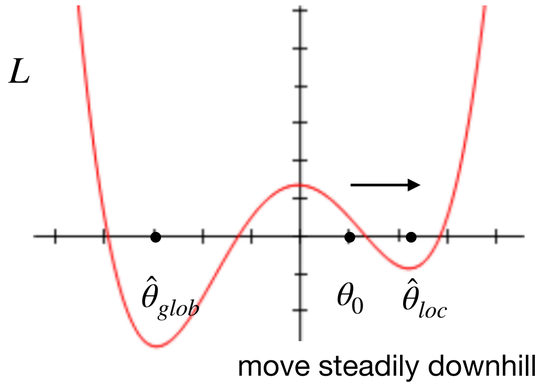
Quiz: true or false?

Introducing some noise to the gradient via random sampling mini-batches reduces the impact of the initial weights on the final point of gradient descent algorithm.

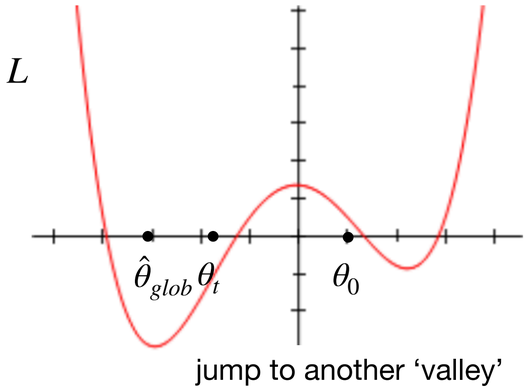
Stochastic Gradient Descent

Quiz: true or false?

Introducing some noise to the gradient via random sampling mini-batches reduces the impact of the initial weights on the final point of gradient descent algorithm.



full-batch/deterministic GD



mini-batch/stochastic GD

Batch Size

Quiz: true or false?

The mini-batch size is a hyperparameter:

- For large batches, GD performs more weight updates per second because each one requires less computation
- For small batches, GD converges in fewer weight updates because each stochastic gradient is less noisy

Batch Size

Quiz: true or false?

The mini-batch size is a hyperparameter:

- For large batches, GD performs more weight updates per second because each one requires less computation
- For small batches, GD converges in fewer weight updates because each stochastic gradient is less noisy

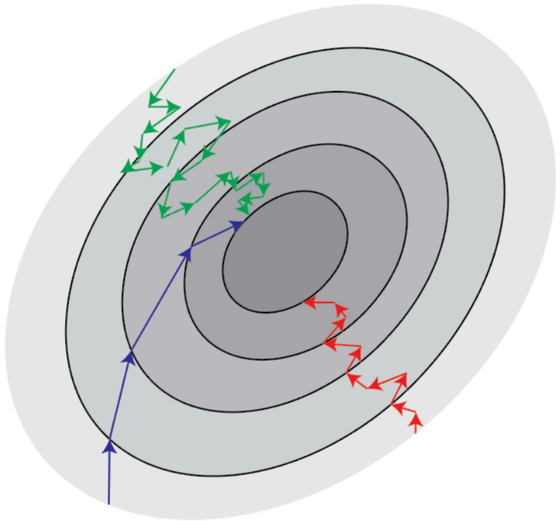
Batch Size

Quiz: true or false?

The mini-batch size is a hyperparameter:

- For small batches, GD performs more weight updates per second because each one requires less computation
- For large batches, GD converges in fewer weight updates because each stochastic gradient is less noisy

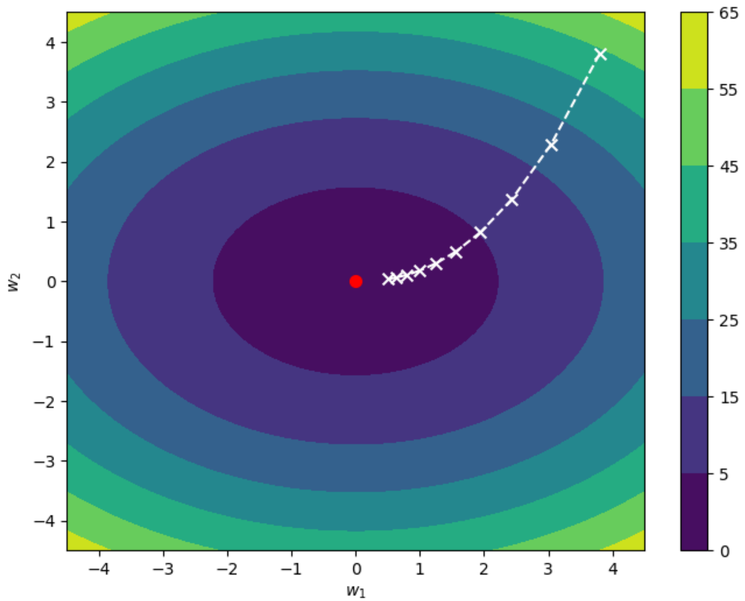
GD	-	Batch size	
full batch		n	[training set size]
mini-batch		1< m < n	[typically a power of 2 between 8 and 512]
stochastic		1	



SGD Learning Rate

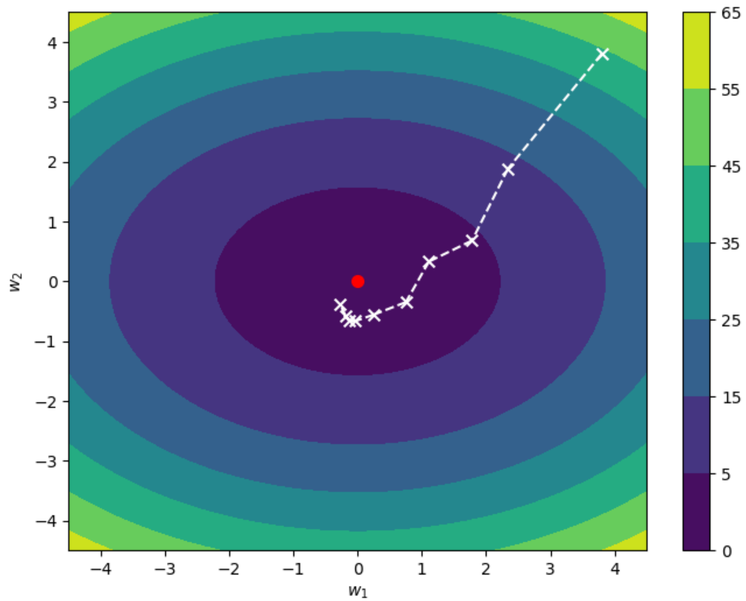
Weights update: $w \longleftarrow w - \alpha \frac{1}{m} \sum_{i=1}^{m} \frac{\partial l^{(i)}}{\partial w}$

Deterministic GD



point steadily downhill

Stochastic GD



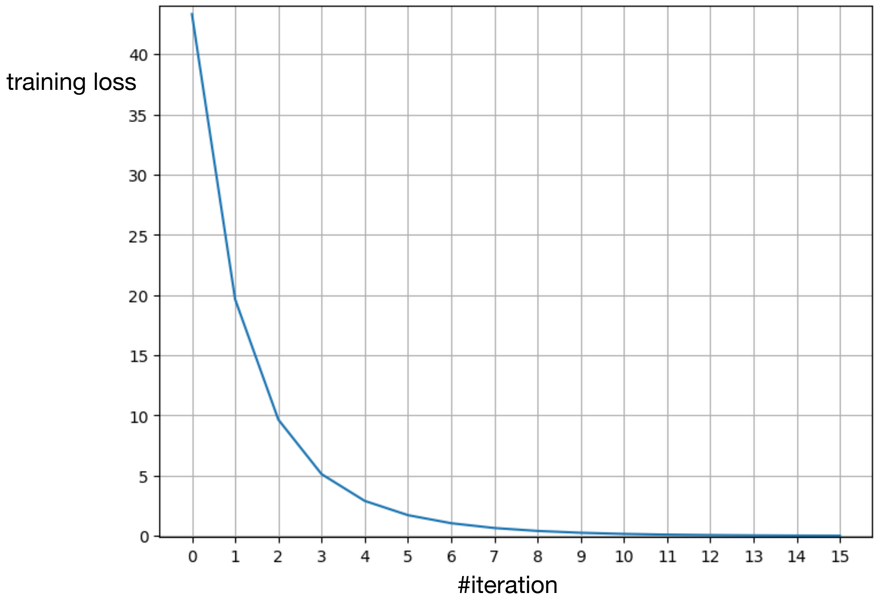
on average point downhill

$\alpha = 0.1$

SGD Learning Rate

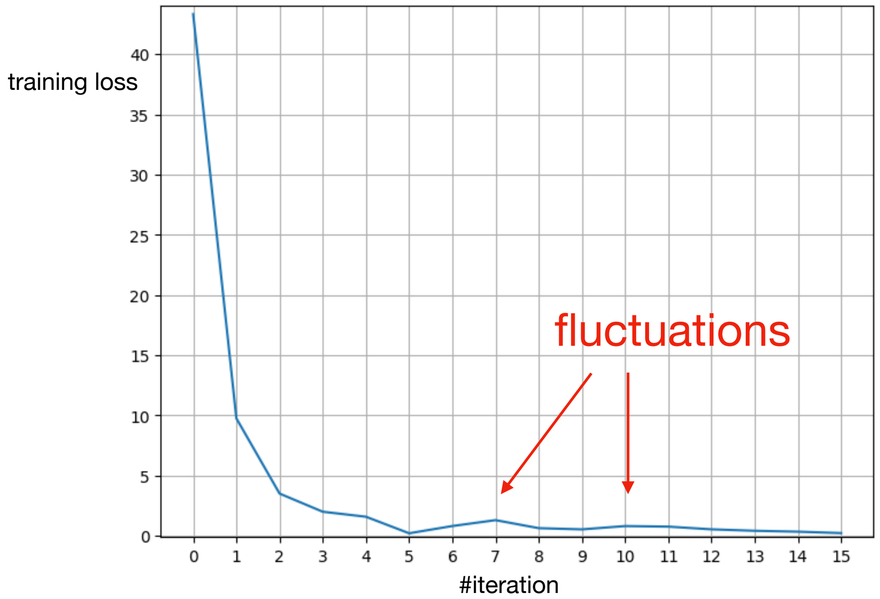
Weights update: $w \longleftarrow w - \alpha \frac{1}{m} \sum_{i=1}^{m} \frac{\partial l^{(i)}}{\partial w}$

Deterministic GD



point steadily downhill

Stochastic GD



on average point downhill

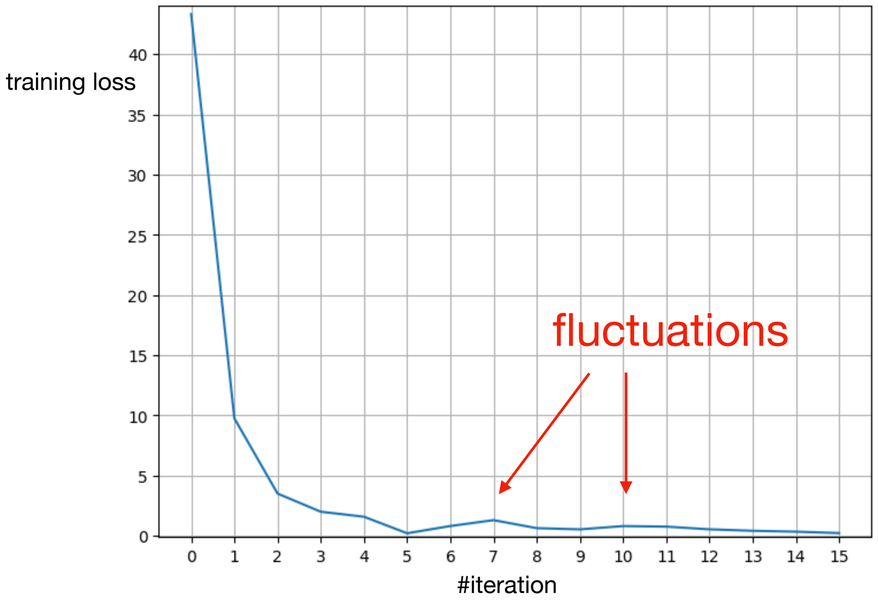
$\alpha = 0.1$

SGD Learning Rate

Weights update: $w \longleftarrow w - \alpha \frac{1}{m} \sum_{i=1}^{m} \frac{\partial l^{(i)}}{\partial w}$

- **On average** parameters are moving in the direction to decrease the loss
- The gradients do not vanish even when arriving at a minimum

Stochastic GD



on average point downhill

SGD Learning Rate

Weights update: $w \longleftarrow w - \alpha \frac{1}{m} \sum_{i=1}^{m} \frac{\partial l^{(i)}}{\partial w}$

- To mitigate fluctuations: gradually decrease the learning rate α_k over iteration (linear or exponential decay)

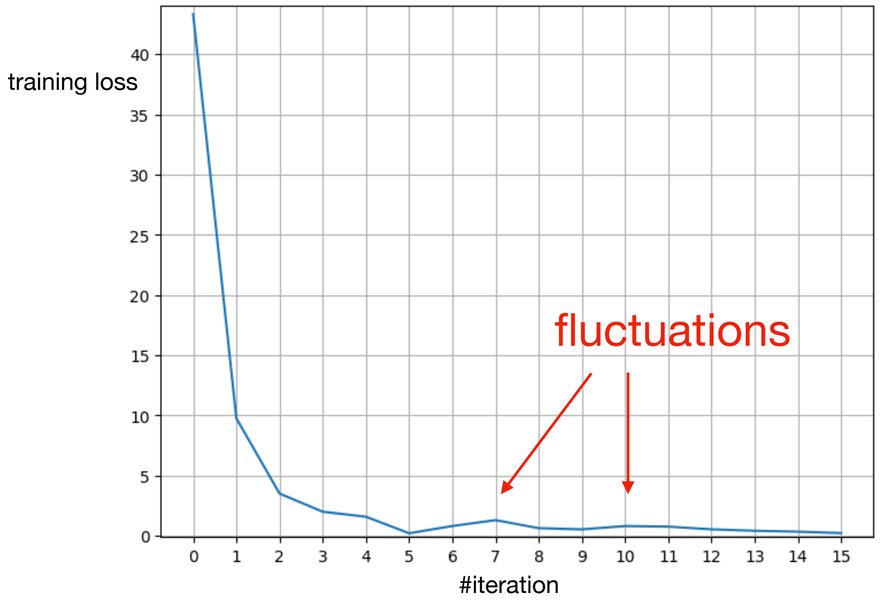
$$a_k = \left(1 - \frac{k}{\tau}\right)\alpha_0 + \frac{k}{\tau}\alpha_T \qquad \text{for } k \leq \tau$$

$$a_k = a_0 e^{-k/\tau} \qquad \text{for } k \geq \tau$$

- In early training stages, use the stochasticity to explore parameter space (fluctuations are fine!)

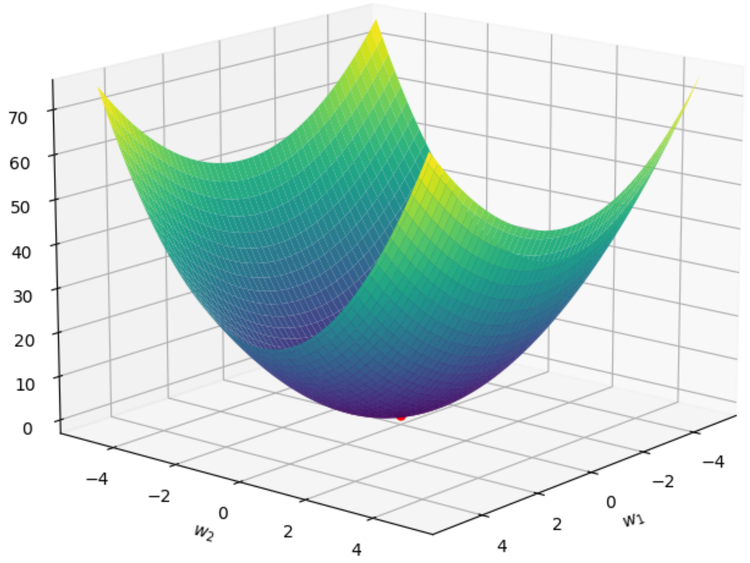
- Later stages, remove the fluctuation effects only due to noisy gradients

Stochastic GD

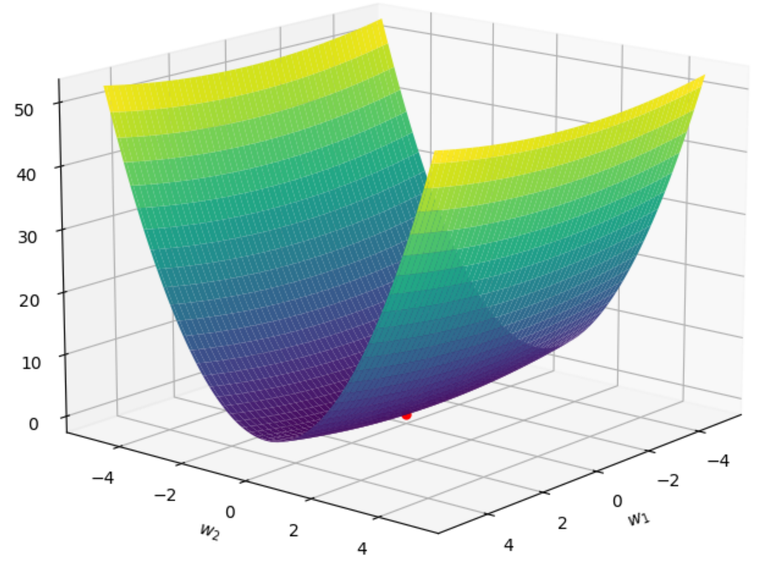


on average point downhill

Curvature directions

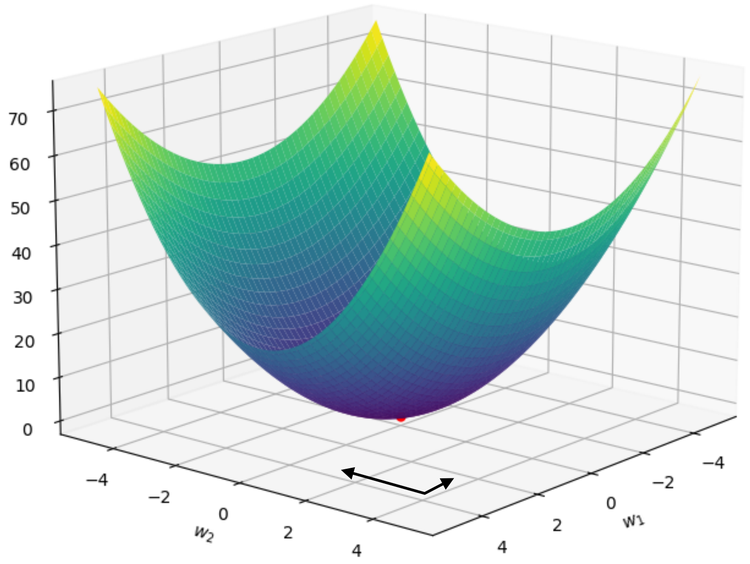


$L(w_1, w_2) = w_1^2 + 2w_2^2$



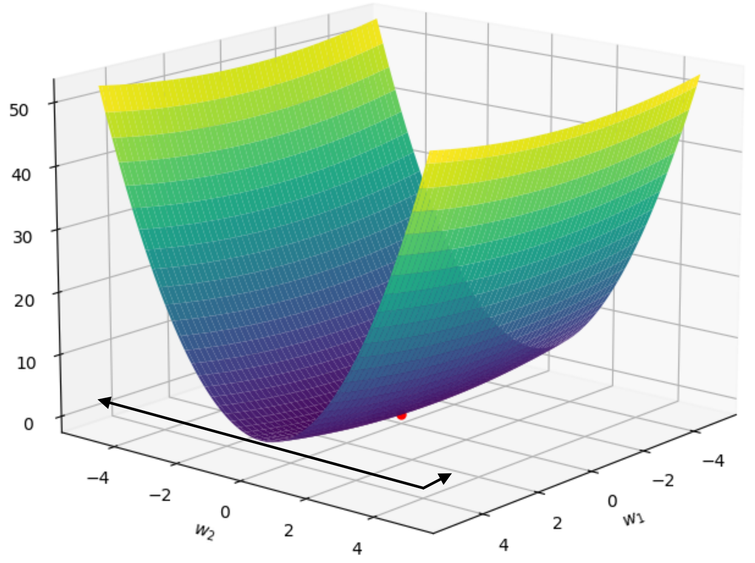
$L(w_1, w_2) = 0.1w_1^2 + 2w_2^2$

Curvature directions



$$L(w_1, w_2) = w_1^2 + 2w_2^2$$

similar curvature in both direction



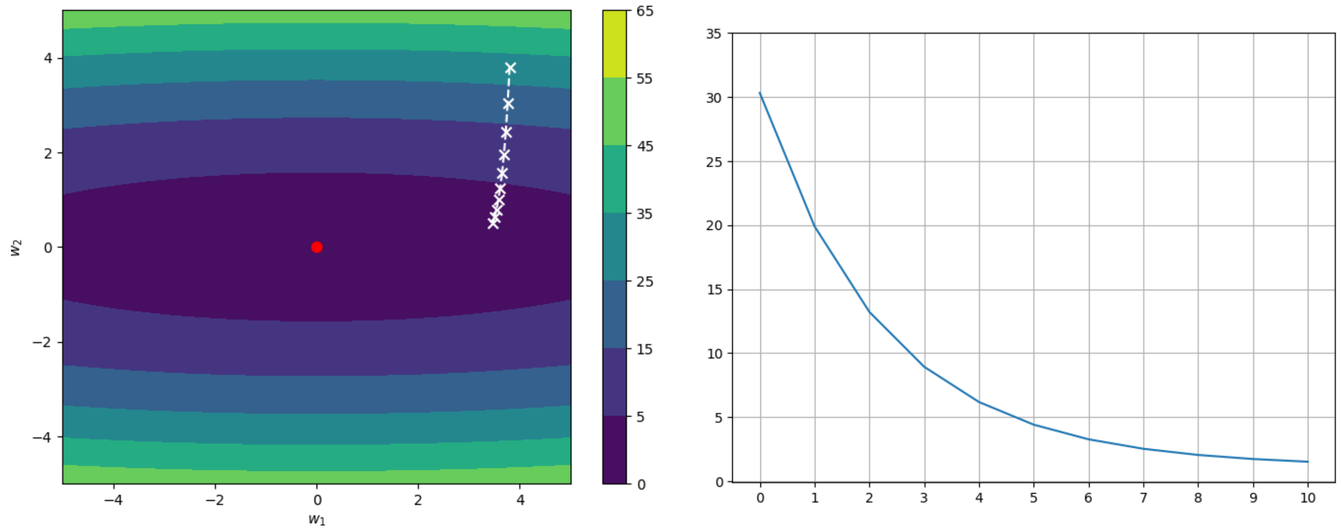
$$L(w_1, w_2) = 0.1w_1^2 + 2w_2^2$$

$$\left|\frac{\partial L}{\partial w_1}\right| << \left|\frac{\partial L}{\partial w_2}\right|$$

much steeper in one direction

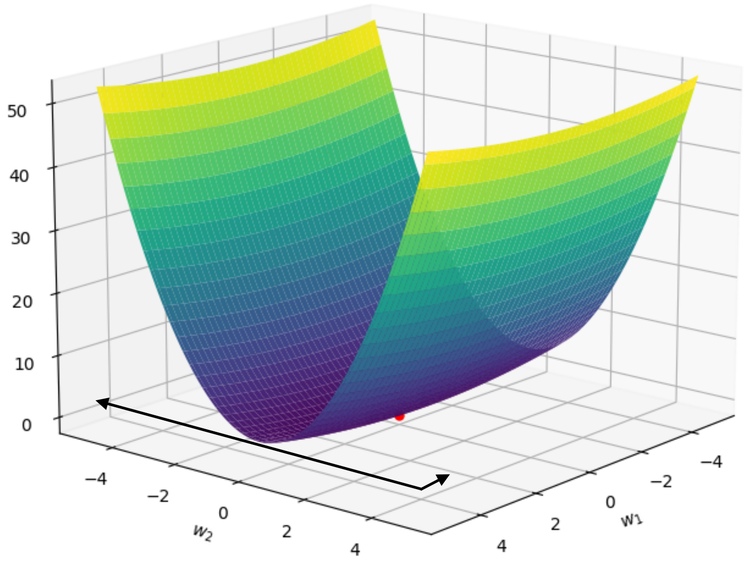
Curvature directions

Same learning rate in both directions: $\alpha = 0.01$



Slow enough for w_2

Too slow for w_1



$$L(w_1, w_2) = 0.1w_1^2 + 2w_2^2$$

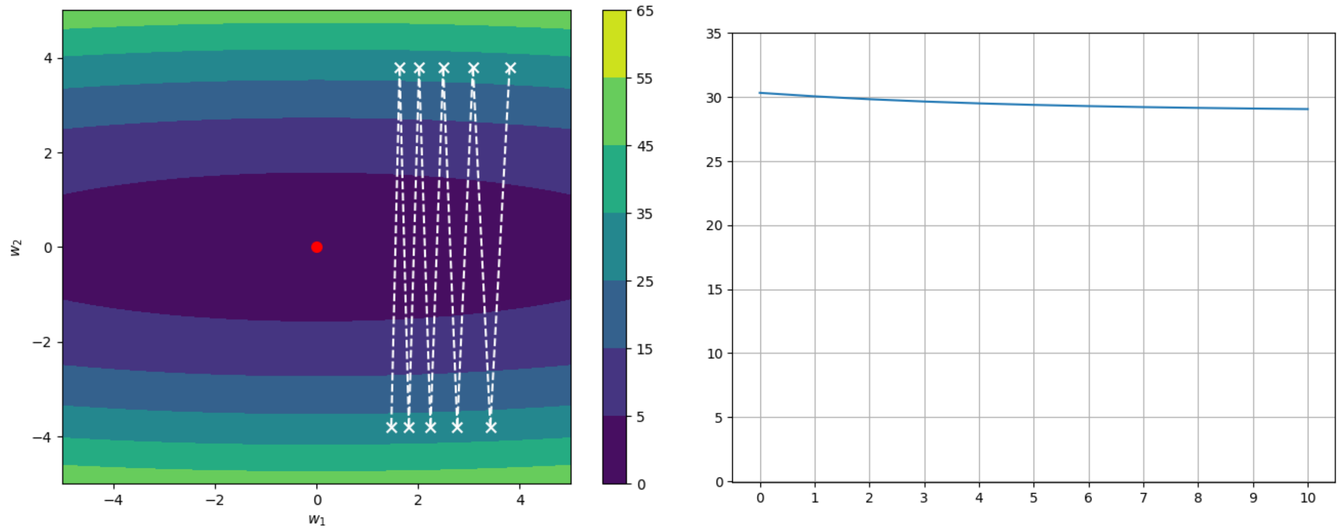
$$\left|\frac{\partial L}{\partial w_1}\right| << \left|\frac{\partial L}{\partial w_2}\right|$$

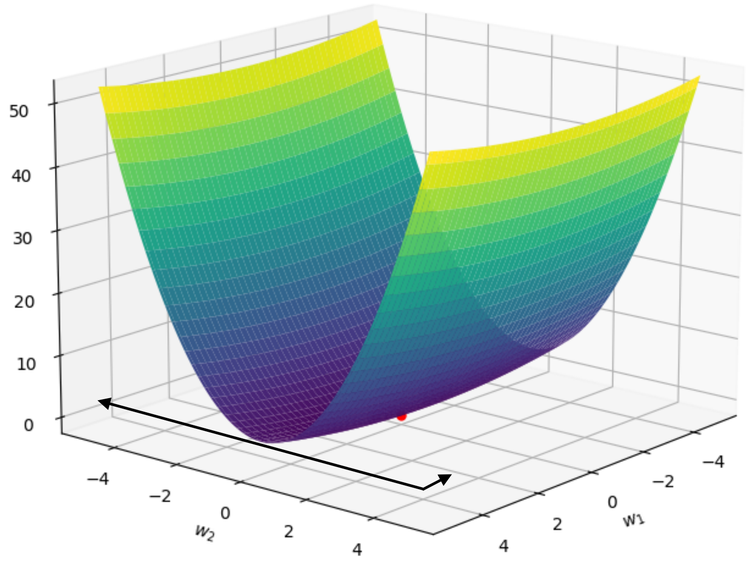
much steeper in one direction

Very slow convergence!

Curvature directions

Same learning rate in both directions: $\alpha = 0.5$





$$L(w_1, w_2) = 0.1w_1^2 + 2w_2^2$$

$$\left|\frac{\partial L}{\partial w_1}\right| << \left|\frac{\partial L}{\partial w_2}\right|$$

much steeper in one direction

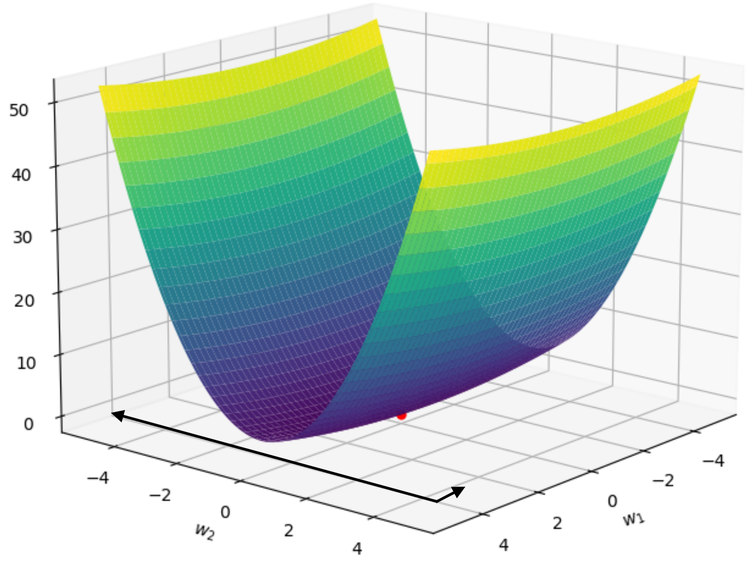
Fast enough for w_1

Too fast for w_2 —> oscillations

Oscillations prevent or slow down convergence!

Normalizing gradients

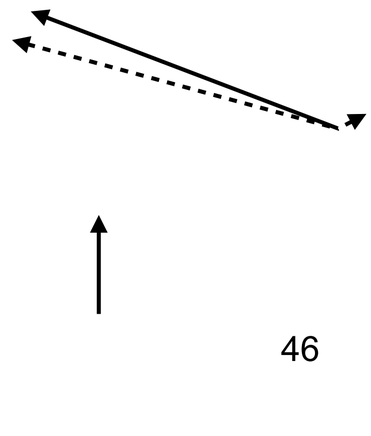
- Large update steps in steeper directions and small in flat directions (oscillation or slow convergence)
- Normalizing the gradient magnitude and retaining only the direction
- Move a fixed distance along each axis at each step
- The learning rate determines how downhill in all the directions



$$L(w_1, w_2) = 0.1w_1^2 + 2w_2^2$$

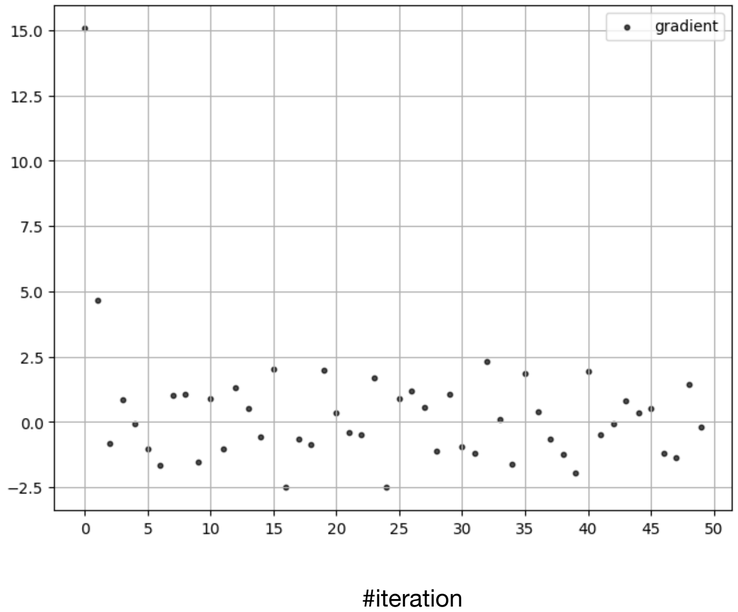
$$\nabla L = \left(\frac{\partial L}{\partial w_1}, \frac{\partial L}{\partial w_2} \right)$$

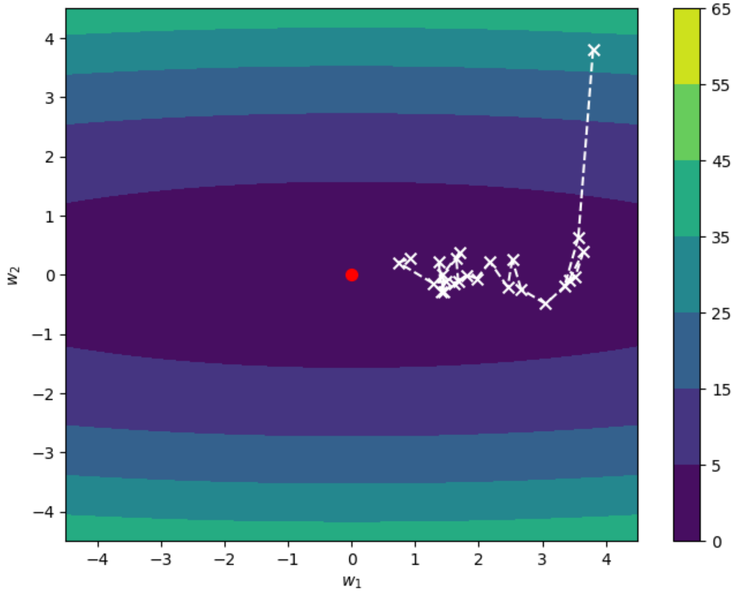
$$\tilde{\nabla} L = \left(\frac{\frac{\partial L}{\partial w_1}}{\left| \frac{\partial L}{\partial w_1} \right|}, \frac{\frac{\partial L}{\partial w_2}}{\left| \frac{\partial L}{\partial w_2} \right|} \right)$$



Variance of Stochastic Gradient

$$S = \frac{1}{m}\sum_{i=1}^{m}\frac{\partial l^{(i)}}{\partial w_2}$$





$$S = S^0, S^1, S^2, \ldots, S^{50}$$

Momentum

v velocity

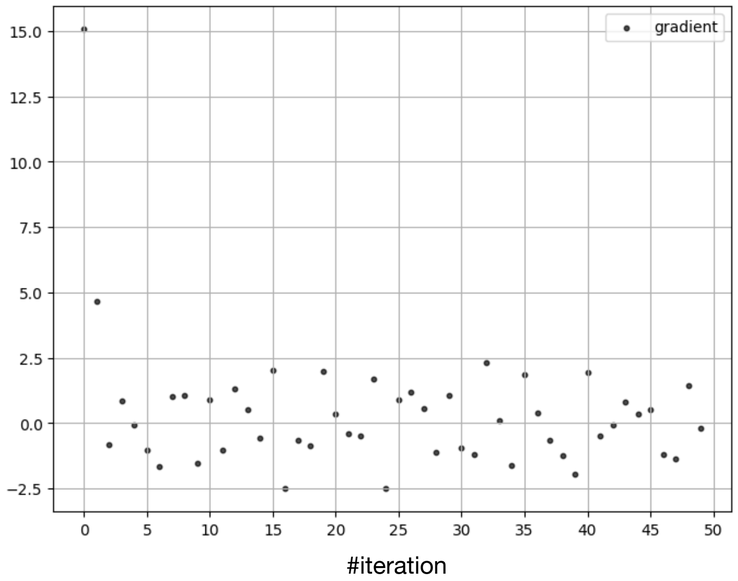
$$v \longleftarrow \beta v + (1-\beta)\,\frac{1}{m}\left(\sum_{i=1}^{m}\frac{\partial l^{(i)}}{\partial w}\right) \qquad \beta \in (0,1)$$

Momentum

v velocity

$$v \longleftarrow \beta v + (1-\beta)\frac{1}{m}\left(\sum_{i=1}^{m}\frac{\partial l^{(i)}}{\partial w}\right) \qquad \beta \in (0,1)$$

$$S = \frac{1}{m}\sum_{i=1}^{m}\frac{\partial l^{(i)}}{\partial w_2}$$



$\beta = 0.9,\ v^{-1} = 0$

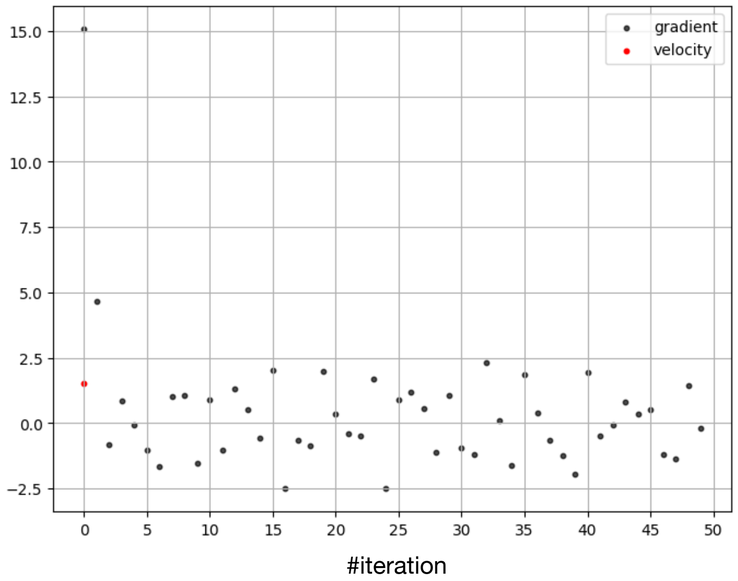
$S = S^0, S^1, S^2, \ldots, S^{50}$

Momentum

v velocity

$$v \longleftarrow \beta v + (1-\beta)\frac{1}{m}\left(\sum_{i=1}^{m}\frac{\partial l^{(i)}}{\partial w}\right) \qquad \beta \in (0,1)$$

$$S = \frac{1}{m}\sum_{i=1}^{m}\frac{\partial l^{(i)}}{\partial w_2}$$



$$\beta = 0.9,\ v^{-1} = 0$$

$$v^0 = \beta \times 0 + (1-\beta)S^0 = (1-\beta)S^0$$

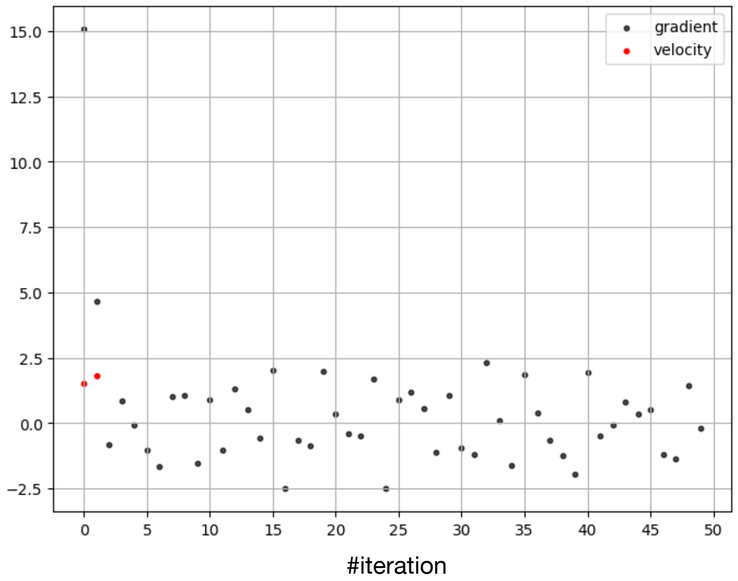
$$S = S^0, S^1, S^2, \ldots, S^{50}$$

Momentum

v velocity

$$v \longleftarrow \beta v + (1-\beta)\frac{1}{m}\left(\sum_{i=1}^{m}\frac{\partial l^{(i)}}{\partial w}\right) \qquad \beta \in (0,1)$$

$$S = \frac{1}{m}\sum_{i=1}^{m}\frac{\partial l^{(i)}}{\partial w_2}$$



$$\beta = 0.9,\ v^{-1} = 0$$

$$v^0 = (1-\beta)S^0$$
$$v^1 = \beta v^0 + (1-\beta)S^1$$

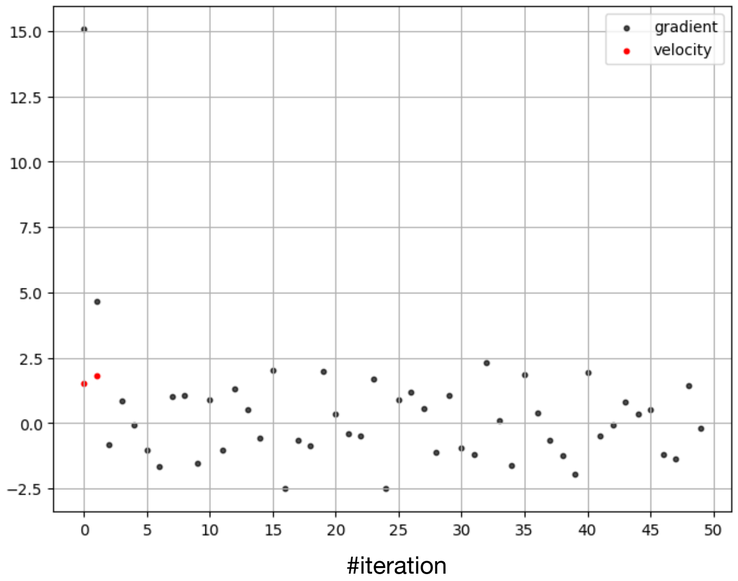
$$S = S^0, S^1, S^2, \ldots, S^{50}$$

Momentum

v velocity

$$v \longleftarrow \beta v + (1-\beta)\frac{1}{m}\left(\sum_{i=1}^{m}\frac{\partial l^{(i)}}{\partial w}\right) \qquad \beta \in (0,1)$$

$$S = \frac{1}{m}\sum_{i=1}^{m}\frac{\partial l^{(i)}}{\partial w_2}$$



$$S = S^0, S^1, S^2, \ldots, S^{50}$$

$$\beta = 0.9,\ v^{-1} = 0$$

$$v^0 = (1-\beta)S^0$$

$$v^0 = (1-\beta)S^0$$

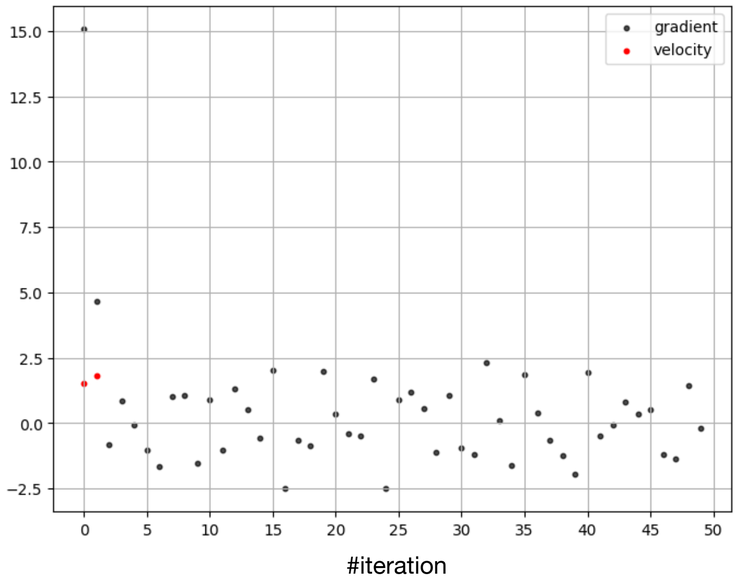
$$v^1 = \beta v^0 + (1-\beta)S^1 = \beta(1-\beta)S^0 + (1-\beta)S^1$$

Momentum

v velocity

$$v \longleftarrow \beta v + (1-\beta)\frac{1}{m}\left(\sum_{i=1}^{m}\frac{\partial l^{(i)}}{\partial w}\right) \qquad \beta \in (0,1)$$

$$S = \frac{1}{m}\sum_{i=1}^{m}\frac{\partial l^{(i)}}{\partial w_2}$$



$$S = S^0, S^1, S^2, \ldots, S^{50}$$

$$\beta = 0.9,\ v^{-1} = 0$$

$$v^0 = (1-\beta)S^0$$

$$v^1 = (1-\beta)(\beta S^0 + S^1)$$

0.9 1

past gradients have lower weight

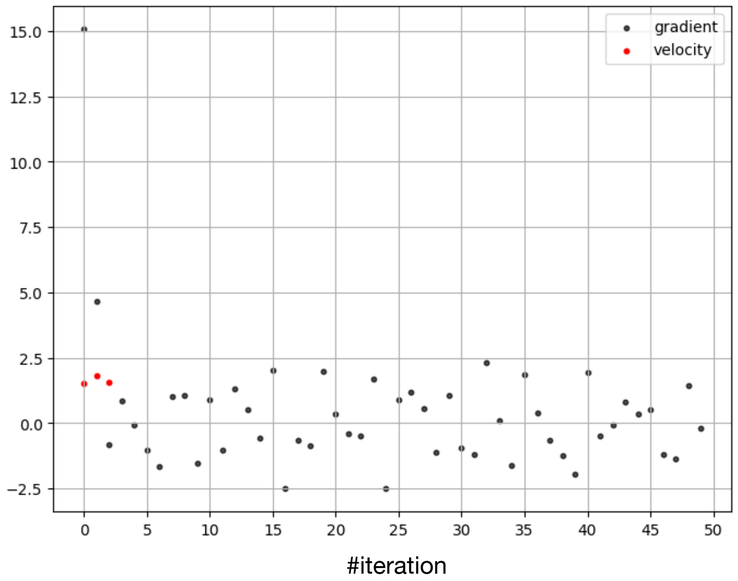
not exactly an average…

Momentum

v velocity

$$v \longleftarrow \beta v + (1-\beta)\frac{1}{m}\left(\sum_{i=1}^{m}\frac{\partial l^{(i)}}{\partial w}\right) \qquad \beta \in (0,1)$$

$$S = \frac{1}{m}\sum_{i=1}^{m}\frac{\partial l^{(i)}}{\partial w_2}$$



$$S = S^0, S^1, S^2, \ldots, S^{50}$$

$$\beta = 0.9,\ v^{-1} = 0$$

$$v^0 = (1-\beta)S^0$$
$$v^1 = (1-\beta)(\beta S^0 + S^1)$$
$$v^2 = \beta v^1 + (1-\beta)S^2$$

0.9 0.1

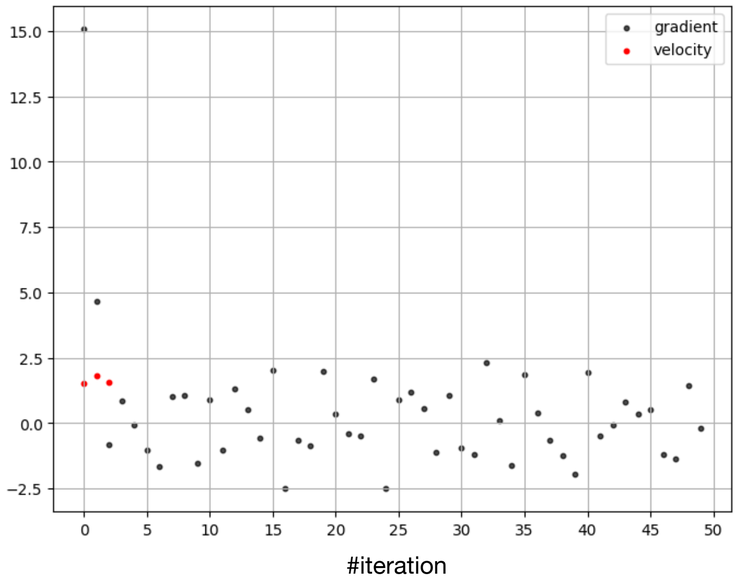
average between past gradient statistics and present gradient

Momentum

v velocity

$$v \longleftarrow \beta v + (1-\beta)\frac{1}{m}\left(\sum_{i=1}^{m}\frac{\partial l^{(i)}}{\partial w}\right) \qquad \beta \in (0,1)$$

$$S = \frac{1}{m}\sum_{i=1}^{m}\frac{\partial l^{(i)}}{\partial w_2}$$



$$S = S^0, S^1, S^2, \ldots, S^{50}$$

$$\beta = 0.9,\ v^{-1} = 0$$

$$v^0 = (1-\beta)S^0$$
$$v^1 = (1-\beta)(\beta S^0 + S^1)$$
$$v^2 = (1-\beta)(\beta^2 S^0 + \beta S^1 + S^2)$$

0.81 0.9 1

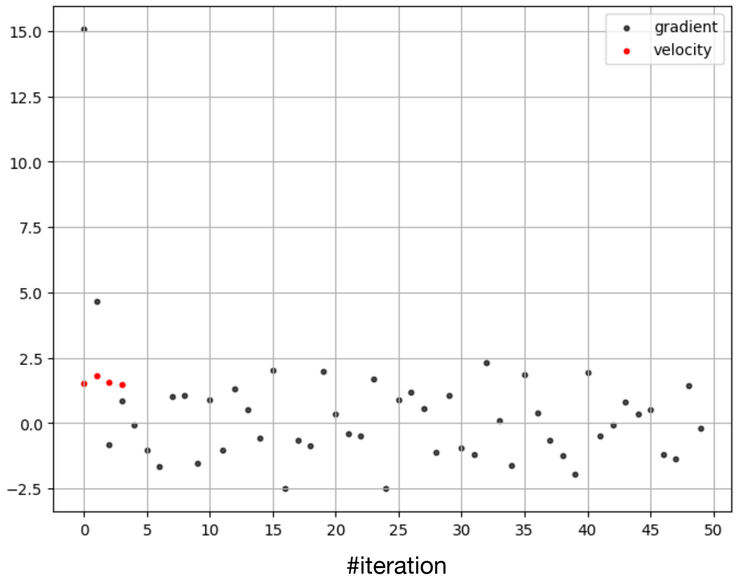
exponential weighted moving average

Momentum

v velocity

$$v \longleftarrow \beta v + (1-\beta)\frac{1}{m}\left(\sum_{i=1}^{m}\frac{\partial l^{(i)}}{\partial w}\right) \qquad \beta \in (0,1)$$

$$S = \frac{1}{m}\sum_{i=1}^{m}\frac{\partial l^{(i)}}{\partial w_2}$$



$$S = S^0, S^1, S^2, \ldots, S^{50}$$

$$\beta = 0.9,\ v^{-1} = 0$$

$$v^0 = (1-\beta)S^0$$
$$v^1 = (1-\beta)\beta\, S^0 + (1-\beta)\, S^1$$
$$v^2 = (1-\beta)(\beta^2 S^0 + \beta S^1 + S^2)$$

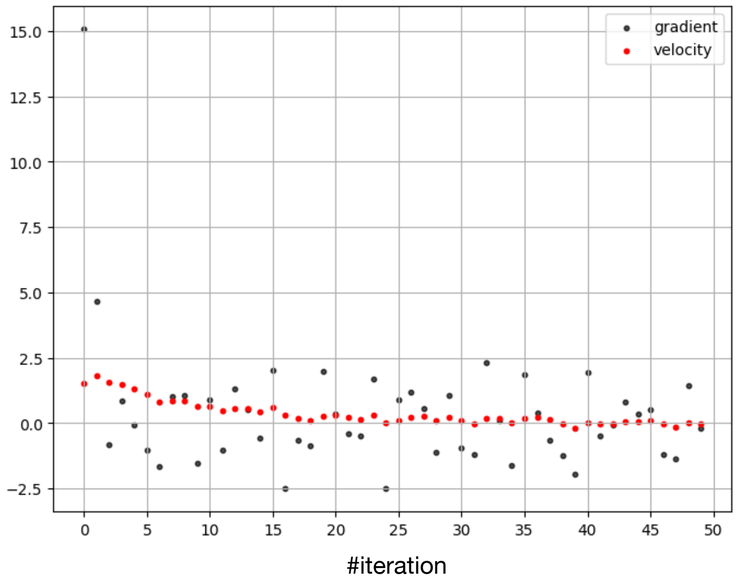
…

Momentum

v velocity

$$v \longleftarrow \beta v + (1-\beta)\frac{1}{m}\left(\sum_{i=1}^{m}\frac{\partial l^{(i)}}{\partial w}\right) \qquad \beta \in (0,1)$$

$$S = \frac{1}{m}\sum_{i=1}^{m}\frac{\partial l^{(i)}}{\partial w_2}$$



$$S = S^0, S^1, S^2, \ldots, S^{50}$$

$$\beta = 0.9,\ v^{-1} = 0$$

$$v^0 = (1-\beta)S^0$$
$$v^1 = (1-\beta)\beta\, S^0 + (1-\beta)\, S^1$$
$$v^2 = (1-\beta)(\beta^2 S^0 + \beta S^1 + S^2)$$

$\ldots$

$$v^k = (1-\beta)\sum_{t=0}^{k}\beta^{k-t}S^t$$

For many iteration (in the limit), it is an average!

Momentum

v velocity

$$v \longleftarrow \beta v + (1-\beta)\frac{1}{m}\left(\sum_{i=1}^{m}\frac{\partial l^{(i)}}{\partial w}\right) \qquad \beta \in (0,1)$$

$$w \longleftarrow w - \alpha\, v$$

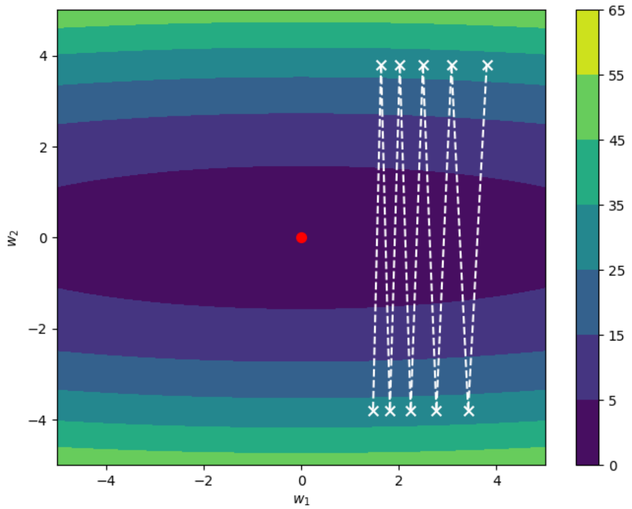
- The update step depends on an (exponential weighted) average of past gradients
- The more “aligned” the past gradient statistic and the new gradient are, the bigger is the update step

$$v^k = \beta v^{k-1} + (1-\beta)S^k$$

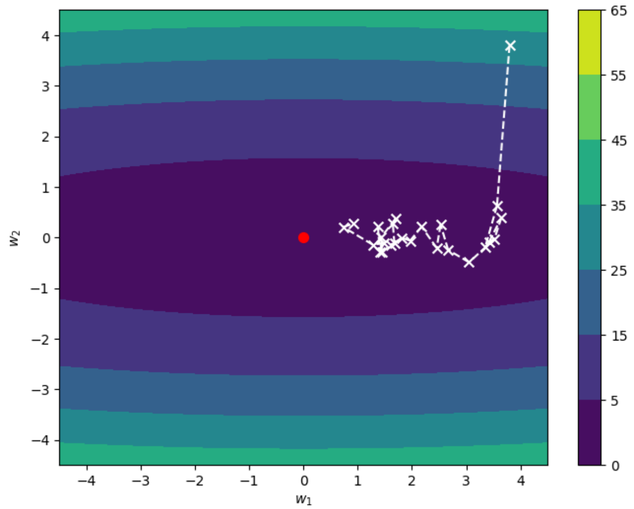
- Momentum smooths the average of noisy gradients and reduce variance

Adaptive moment estimation (Adam)

Combine the momentum with gradient normalization in SGD to prevent oscillations due to badly conditioned curvature and smooth stochastic gradients



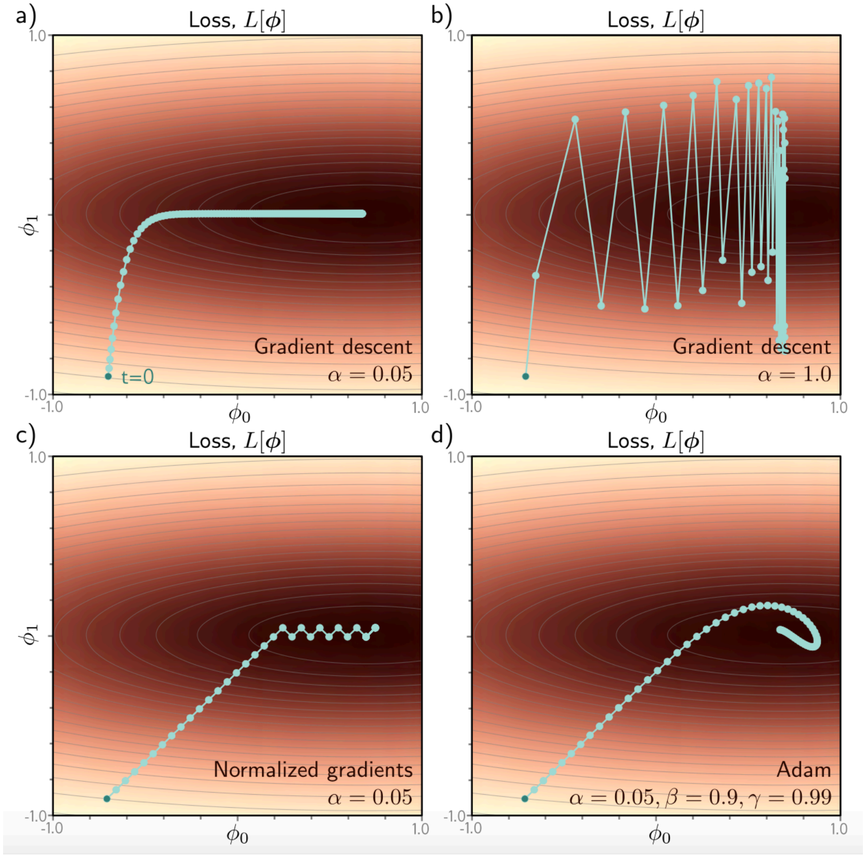
Badly conditioned curvature



Variance of stochastic gradient

Adaptive moment estimation (Adam)

Combine the momentum with gradient normalization in SGD to prevent oscillations due to badly conditioned curvature and smooth stochastic gradients



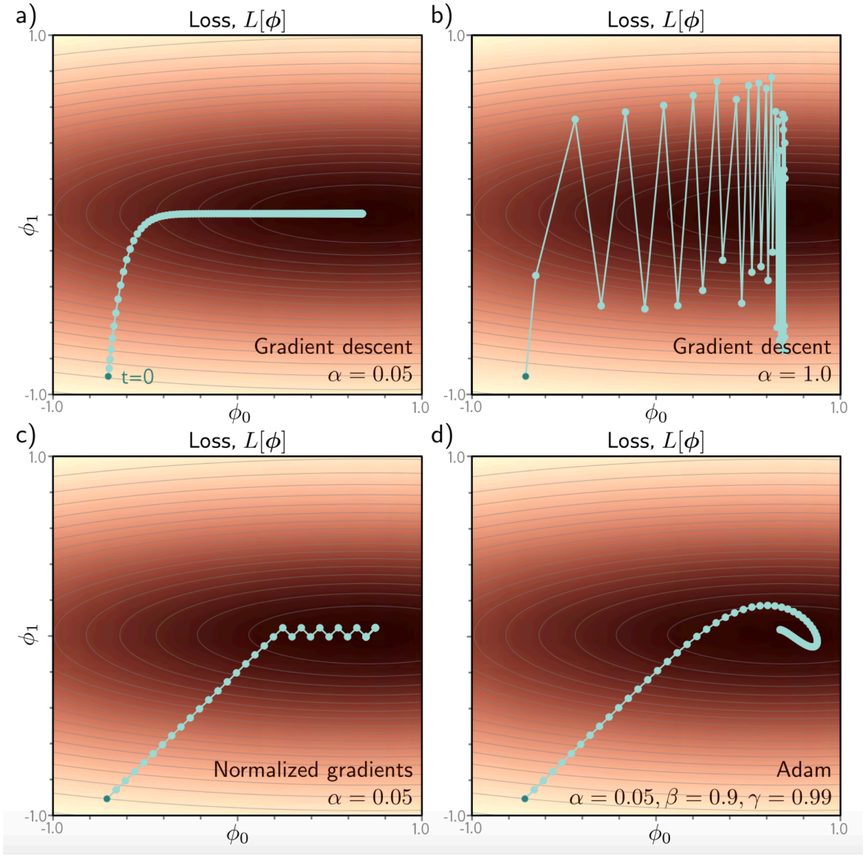
a) ?

b) ?

*Figure from *Understanding Deep Learning by* Simon J.D. Prince Chapter 6.4 page 89

Adaptive moment estimation (Adam)

Combine the momentum with gradient normalization in SGD to prevent oscillations due to badly conditioned curvature and smooth stochastic gradients



Badly conditioned curvature of the loss

a) Small learning rate ⟶ slow convergence

b) Large learning rate ⟶ oscillations

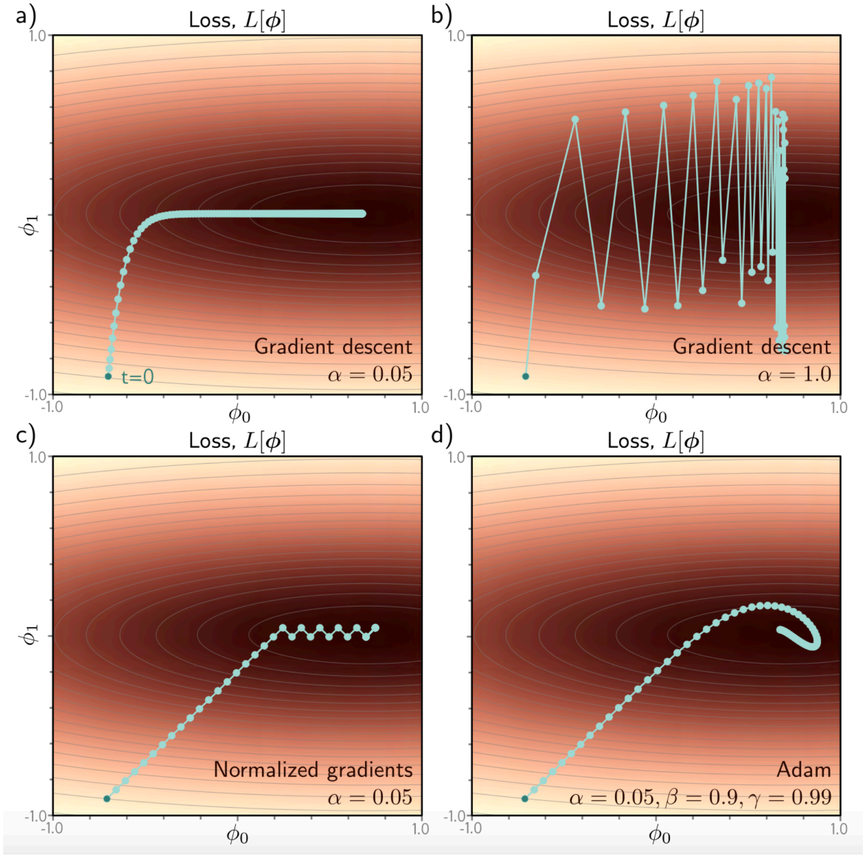
c) ?

d) ?

*Figure from *Understanding Deep Learning by* Simon J.D. Prince Chapter 6.4 page 89

Adaptive moment estimation (Adam)

Combine the momentum with gradient normalization in SGD to prevent oscillations due to badly conditioned curvature and smooth stochastic gradients



Badly conditioned curvature of the loss

a) Small learning rate ⟶ slow convergence

b) Large learning rate ⟶ oscillations

c) Normalization ⟶ moving downhill along both axis

d) Adding momentum ⟶ smoothing stochastic gradients

*Figure from *Understanding Deep Learning by* Simon J.D. Prince Chapter 6.4 page 89

Summary

Topics

- Learning and Optimization difference
- Learning diagnostic: training loss
- Learning rate and batch size
- Momentum, Gradient normalization, Adam

Reading material

- Dive into Deep Learning - Chapter 3.6, 3.6.1
- Understanding Deep Learning - Chapter 6 (until 6.3.1 excluded)
- Roger Grosse notes - Optimization (excluded 4.7)