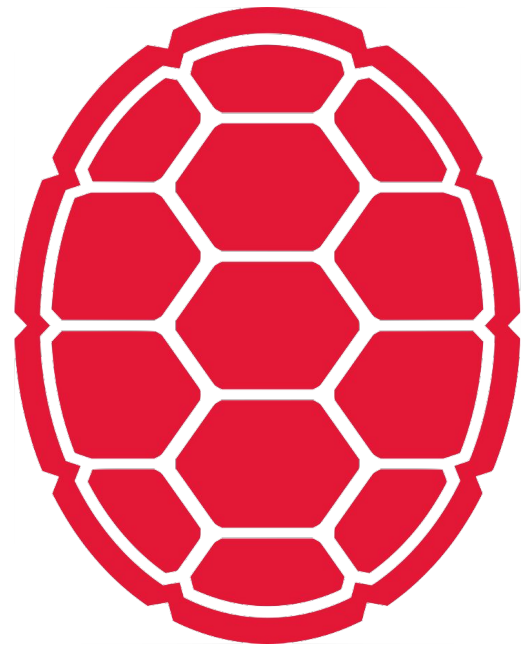
Optimization

Jordan Boyd-Graber

University of Maryland

ADAM

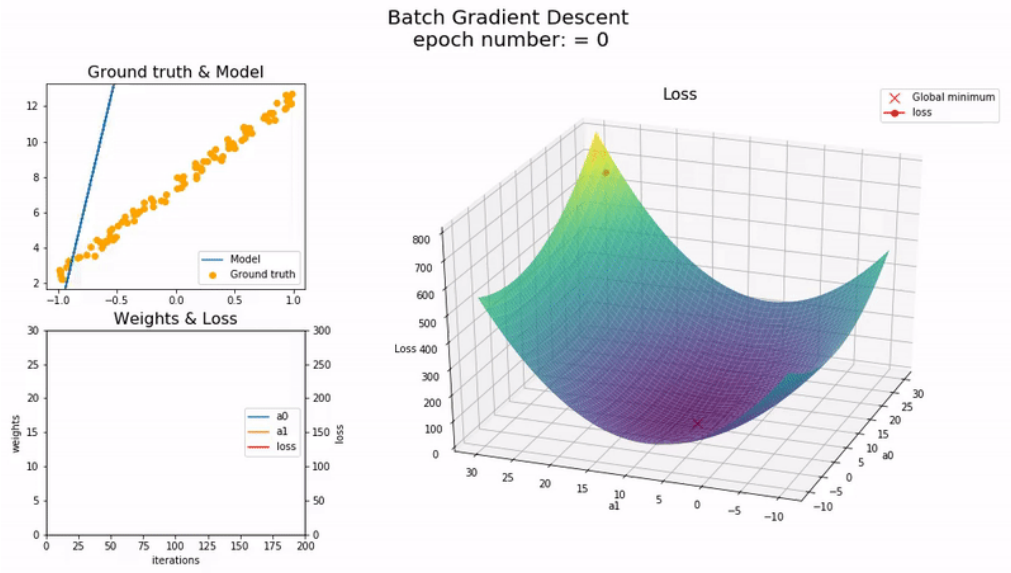


Slides adapted from Fei-Fei Li, Ehsan Adeli, Zane Durante

Recap: Gradient Descent vs. Batch Optimization

- Batch optimization queries all data points to find the best direction.
- Stochastic Gradient Descent (SGD) queries only a few random samples.
- Both aim to minimize loss by moving against the gradient.

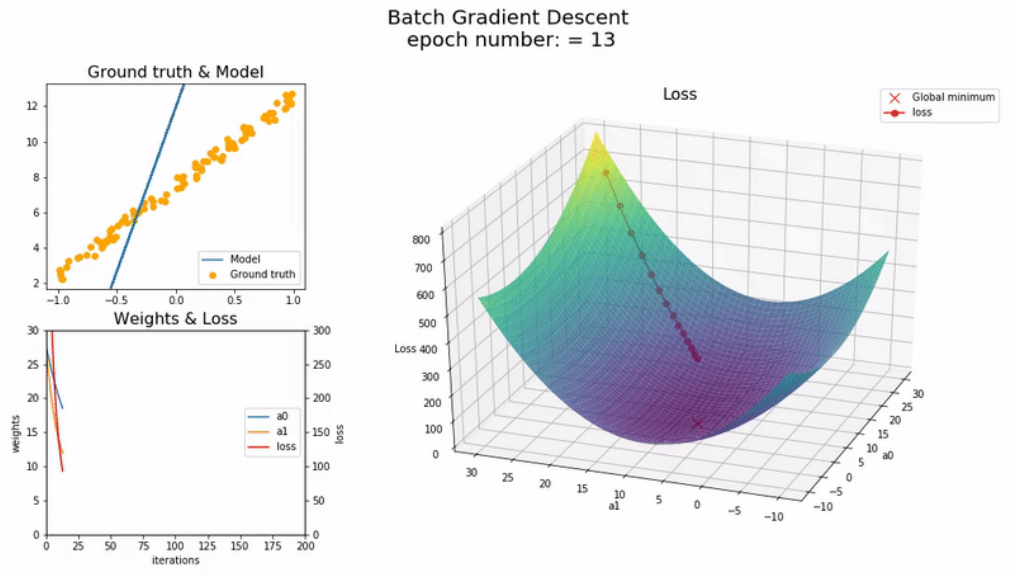
$$\vec{\beta} = \vec{\beta}' - \eta \vec{\nabla}_{\beta} \mathcal{L}(\vec{\beta}')$$



Recap: Gradient Descent vs. Batch Optimization

- Batch optimization queries all data points to find the best direction.
- Stochastic Gradient Descent (SGD) queries only a few random samples.
- Both aim to minimize loss by moving against the gradient.

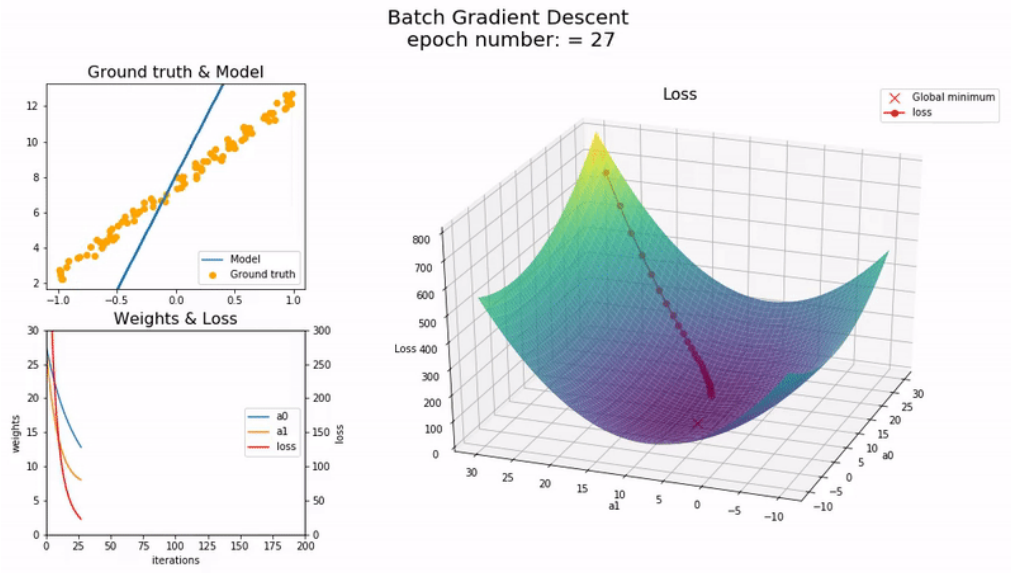
$$\vec{\beta} = \vec{\beta}' - \eta \vec{\nabla}_{\beta} \mathcal{L}(\vec{\beta}')$$



Recap: Gradient Descent vs. Batch Optimization

- Batch optimization queries all data points to find the best direction.
- Stochastic Gradient Descent (SGD) queries only a few random samples.
- Both aim to minimize loss by moving against the gradient.

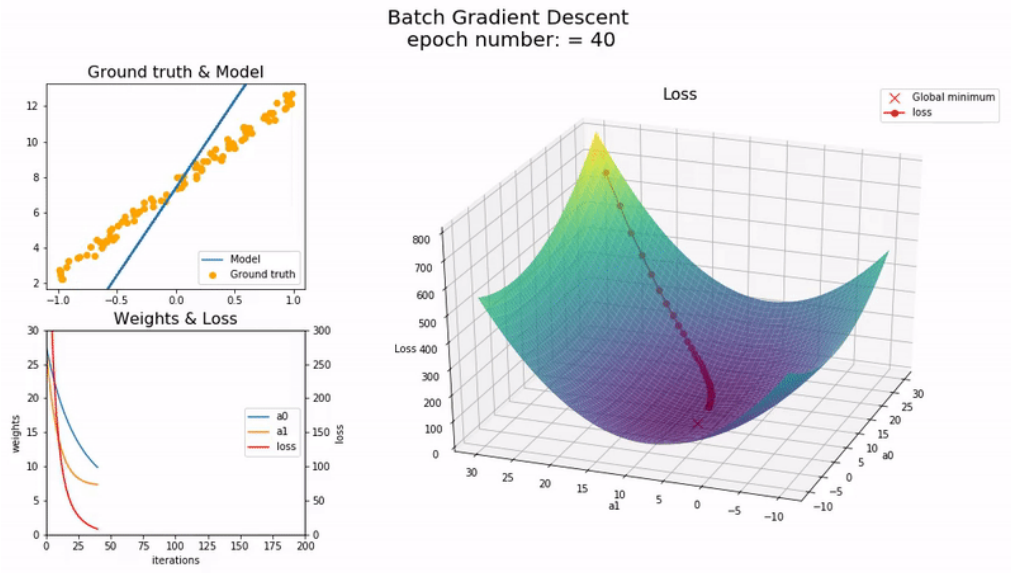
$$\vec{\beta} = \vec{\beta}' - \eta \vec{\nabla}_{\beta} \mathcal{L}(\vec{\beta}')$$



Recap: Gradient Descent vs. Batch Optimization

- Batch optimization queries all data points to find the best direction.
- Stochastic Gradient Descent (SGD) queries only a few random samples.
- Both aim to minimize loss by moving against the gradient.

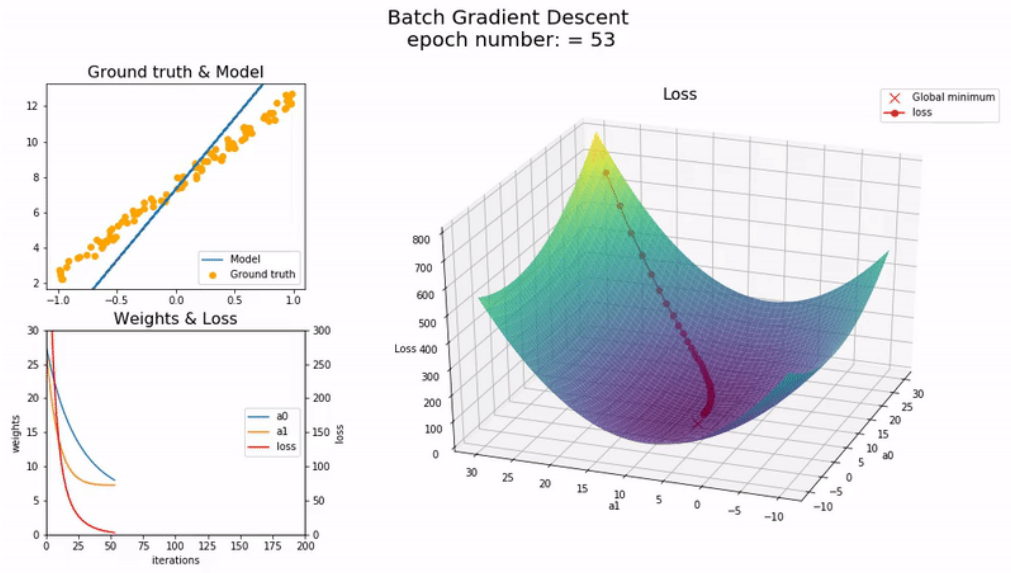
$$\vec{\beta} = \vec{\beta}' - \eta \vec{\nabla}_{\beta} \mathcal{L}(\vec{\beta}')$$



Recap: Gradient Descent vs. Batch Optimization

- Batch optimization queries all data points to find the best direction.
- Stochastic Gradient Descent (SGD) queries only a few random samples.
- Both aim to minimize loss by moving against the gradient.

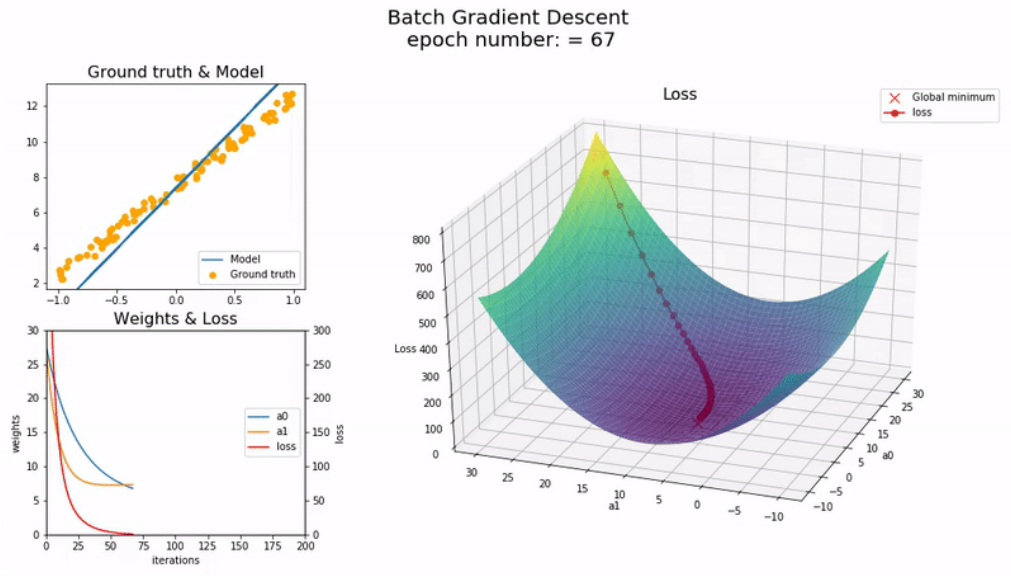
$$\vec{\beta} = \vec{\beta}' - \eta \vec{\nabla}_{\beta} \mathcal{L}(\vec{\beta}')$$



Recap: Gradient Descent vs. Batch Optimization

- Batch optimization queries all data points to find the best direction.
- Stochastic Gradient Descent (SGD) queries only a few random samples.
- Both aim to minimize loss by moving against the gradient.

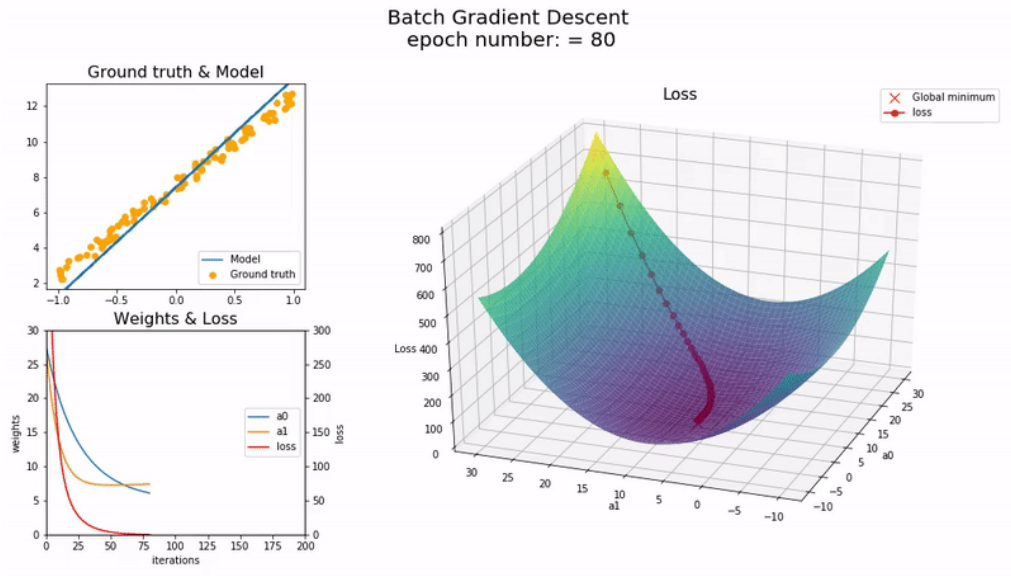
$$\vec{\beta} = \vec{\beta}' - \eta \vec{\nabla}_{\beta} \mathcal{L}(\vec{\beta}')$$



Recap: Gradient Descent vs. Batch Optimization

- Batch optimization queries all data points to find the best direction.
- Stochastic Gradient Descent (SGD) queries only a few random samples.
- Both aim to minimize loss by moving against the gradient.

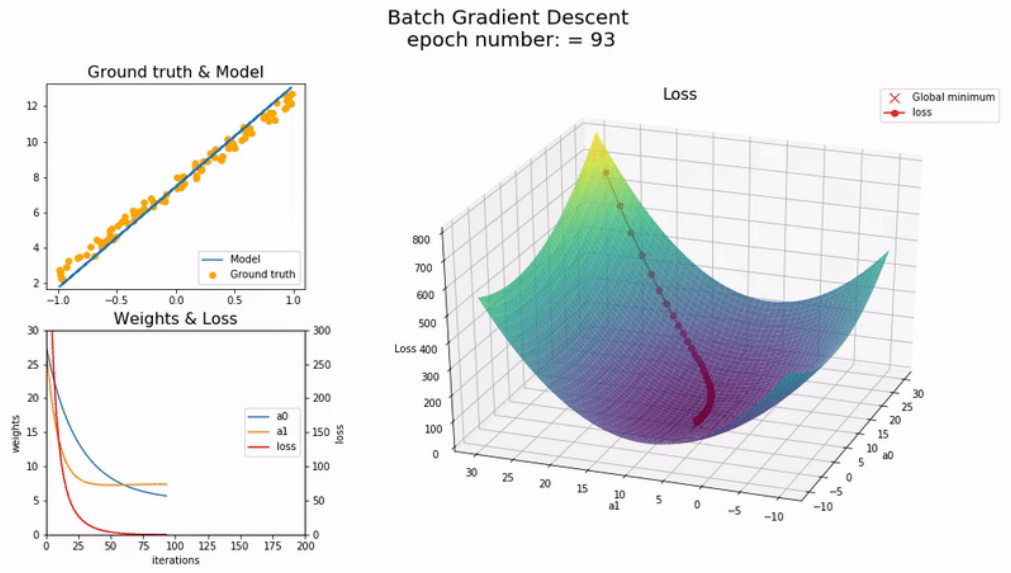
$$\vec{\beta} = \vec{\beta}' - \eta \vec{\nabla}_{\beta} \mathcal{L}(\vec{\beta}')$$



Recap: Gradient Descent vs. Batch Optimization

- Batch optimization queries all data points to find the best direction.
- Stochastic Gradient Descent (SGD) queries only a few random samples.
- Both aim to minimize loss by moving against the gradient.

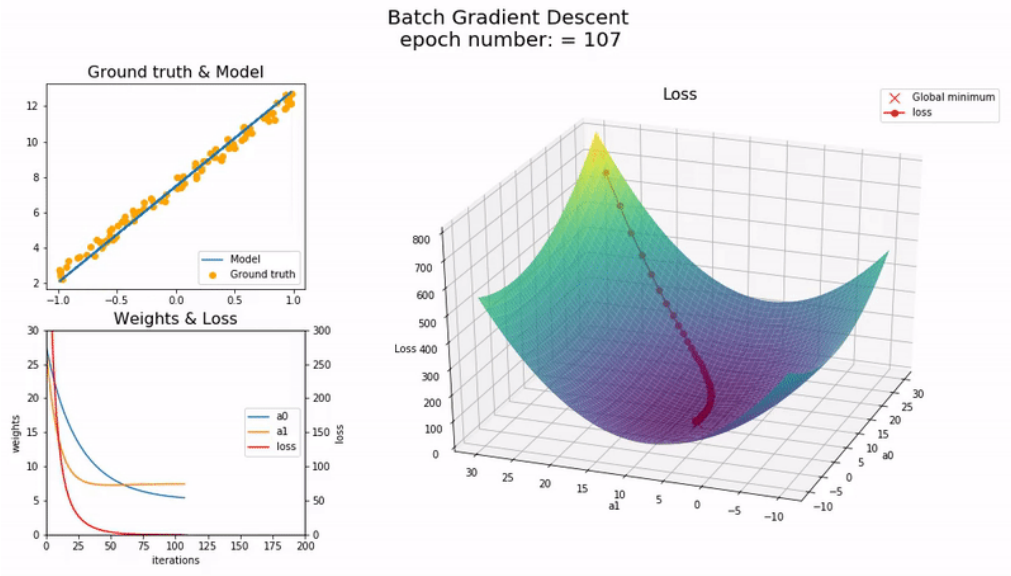
$$\vec{\beta} = \vec{\beta}' - \eta \vec{\nabla}_{\beta} \mathcal{L}(\vec{\beta}')$$



Recap: Gradient Descent vs. Batch Optimization

- Batch optimization queries all data points to find the best direction.
- Stochastic Gradient Descent (SGD) queries only a few random samples.
- Both aim to minimize loss by moving against the gradient.

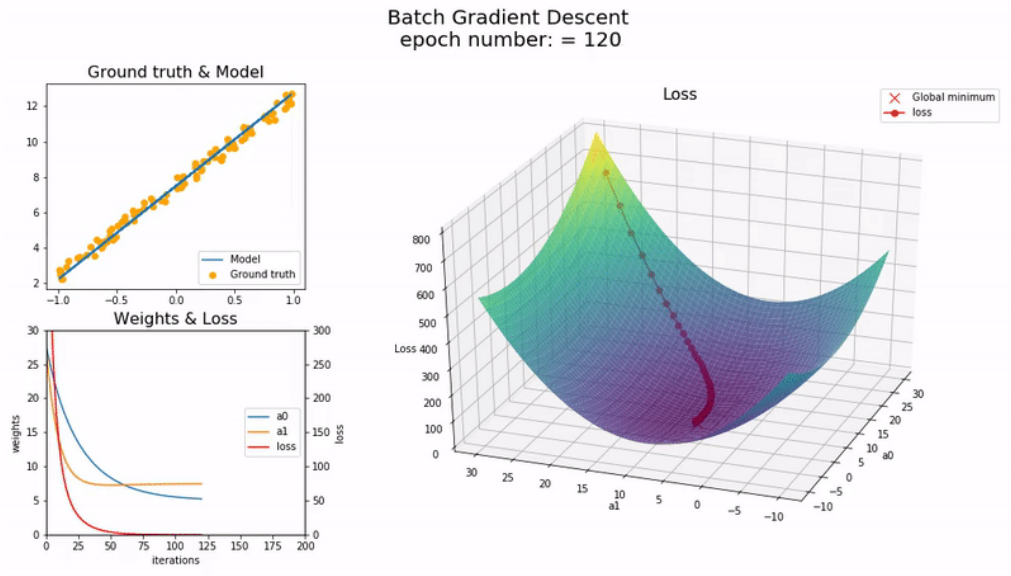
$$\vec{\beta} = \vec{\beta}' - \eta \vec{\nabla}_{\beta} \mathcal{L}(\vec{\beta}')$$



Recap: Gradient Descent vs. Batch Optimization

- Batch optimization queries all data points to find the best direction.
- Stochastic Gradient Descent (SGD) queries only a few random samples.
- Both aim to minimize loss by moving against the gradient.

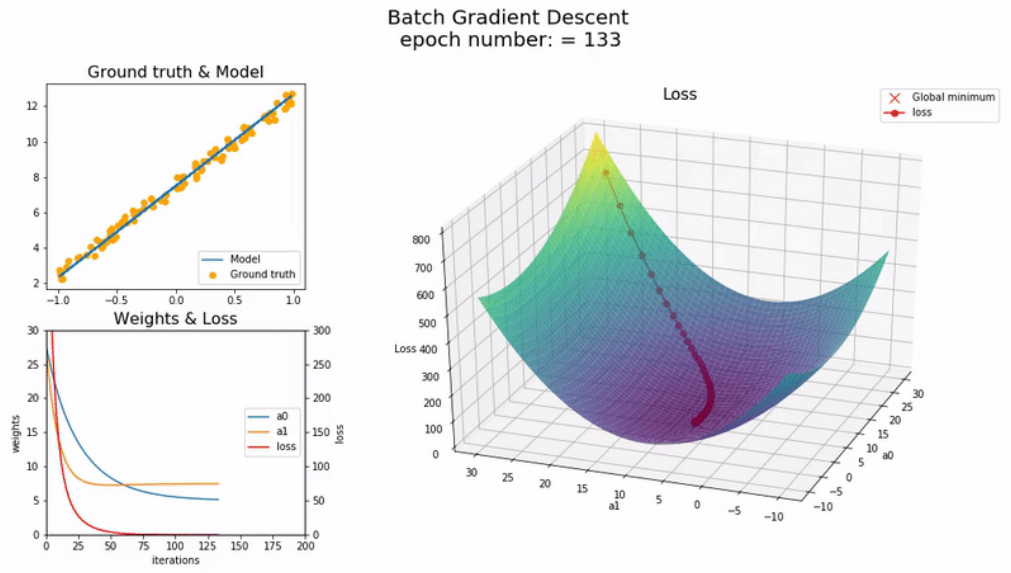
$$\vec{\beta} = \vec{\beta}' - \eta \vec{\nabla}_{\beta} \mathcal{L}(\vec{\beta}')$$



Recap: Gradient Descent vs. Batch Optimization

- Batch optimization queries all data points to find the best direction.
- Stochastic Gradient Descent (SGD) queries only a few random samples.
- Both aim to minimize loss by moving against the gradient.

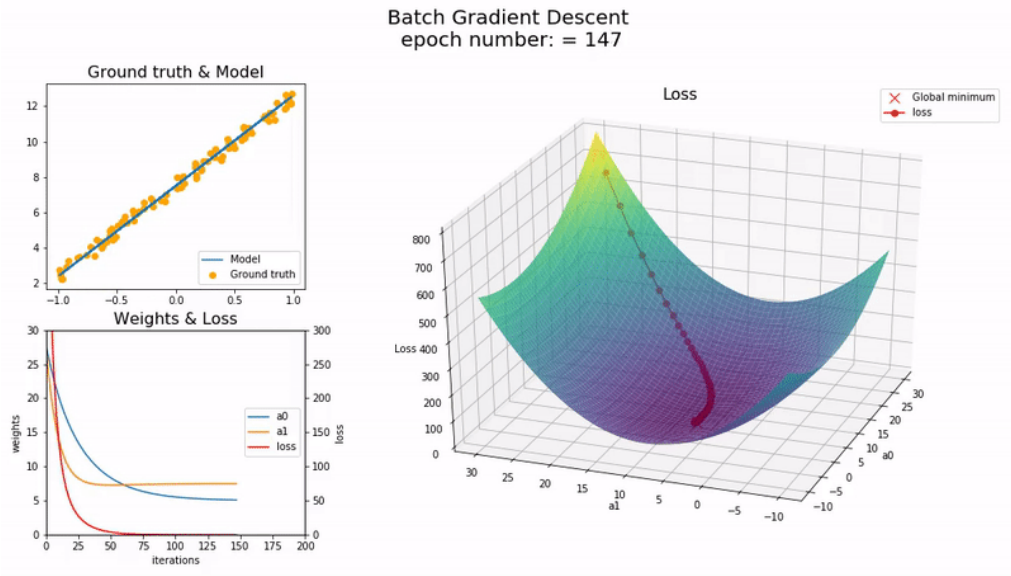
$$\vec{\beta} = \vec{\beta}' - \eta \vec{\nabla}_{\beta} \mathcal{L}(\vec{\beta}')$$



Recap: Gradient Descent vs. Batch Optimization

- Batch optimization queries all data points to find the best direction.
- Stochastic Gradient Descent (SGD) queries only a few random samples.
- Both aim to minimize loss by moving against the gradient.

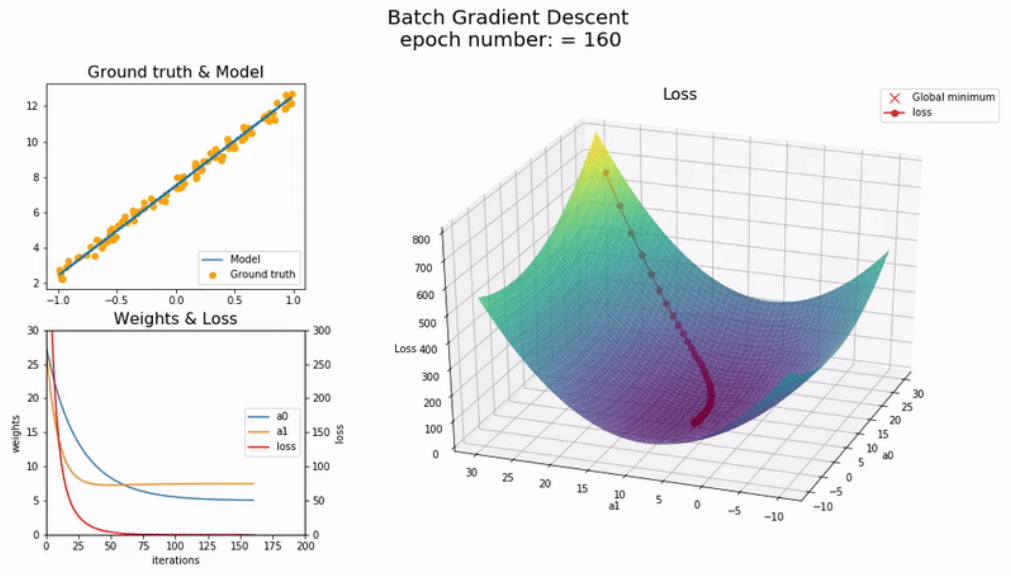
$$\vec{\beta} = \vec{\beta}' - \eta \vec{\nabla}_{\beta} \mathcal{L}(\vec{\beta}')$$



Recap: Gradient Descent vs. Batch Optimization

- Batch optimization queries all data points to find the best direction.
- Stochastic Gradient Descent (SGD) queries only a few random samples.
- Both aim to minimize loss by moving against the gradient.

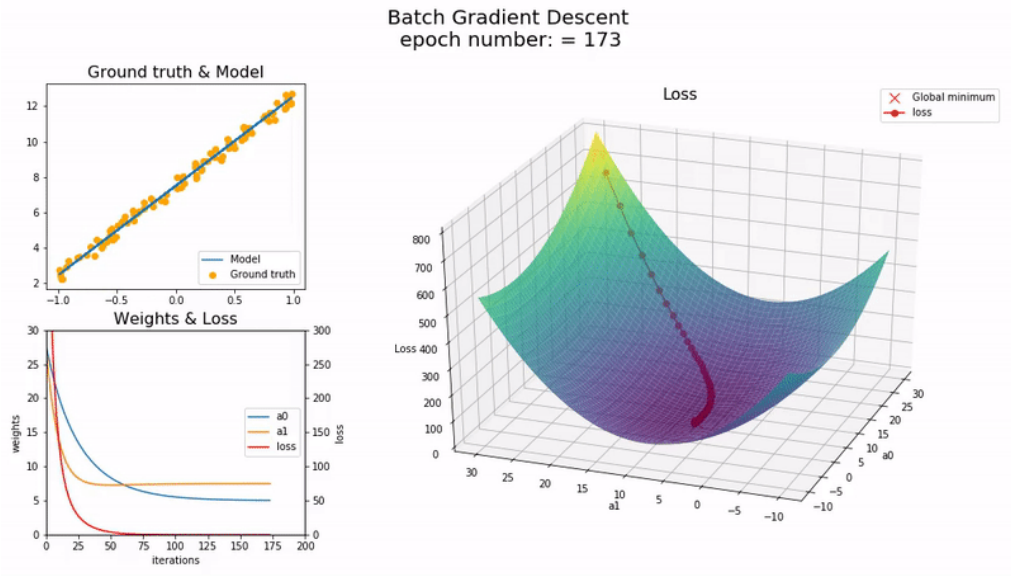
$$\vec{\beta} = \vec{\beta}' - \eta \vec{\nabla}_{\beta} \mathcal{L}(\vec{\beta}')$$



Recap: Gradient Descent vs. Batch Optimization

- Batch optimization queries all data points to find the best direction.
- Stochastic Gradient Descent (SGD) queries only a few random samples.
- Both aim to minimize loss by moving against the gradient.

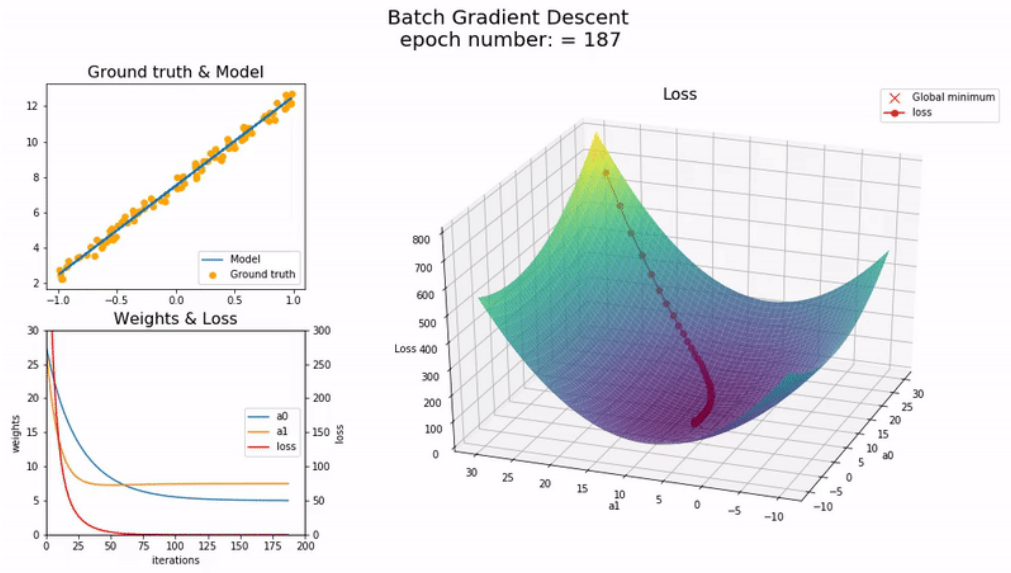
$$\vec{\beta} = \vec{\beta}' - \eta \vec{\nabla}_{\beta} \mathcal{L}(\vec{\beta}')$$



Recap: Gradient Descent vs. Batch Optimization

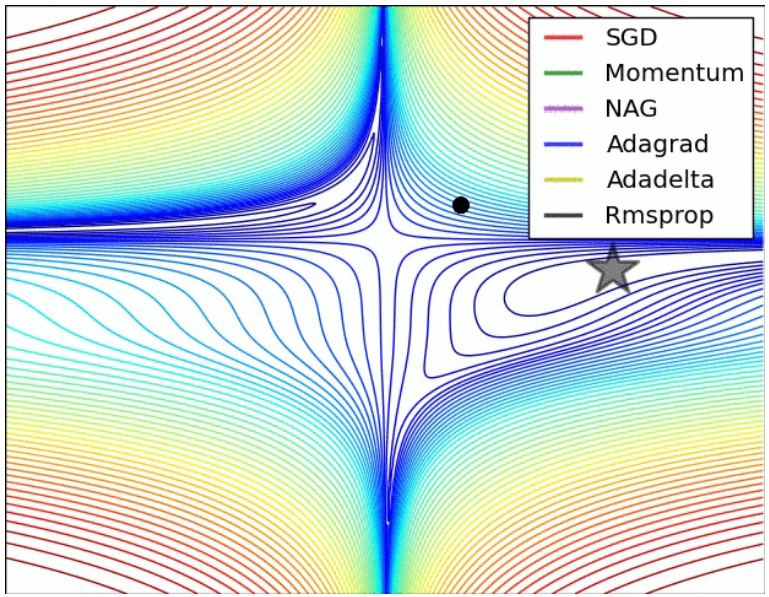
- Batch optimization queries all data points to find the best direction.
- Stochastic Gradient Descent (SGD) queries only a few random samples.
- Both aim to minimize loss by moving against the gradient.

$$\vec{\beta} = \vec{\beta}' - \eta \vec{\nabla}_{\beta} \mathcal{L}(\vec{\beta}')$$



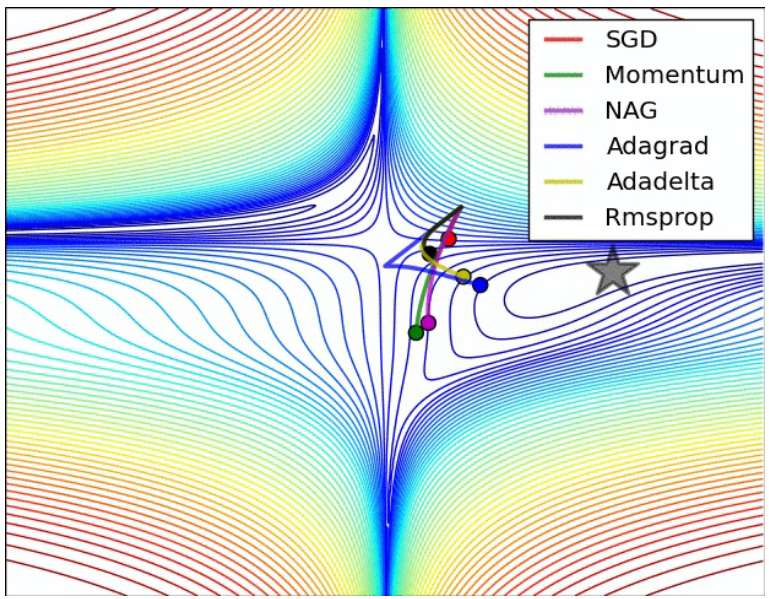
SGD's Isotropy Problem

- SGD treats all dimensions equally (isotropic updates).
- Causes slow progress along shallow gradient directions.
- Inputs with varying scales worsen this problem.
- Adaptive learning rates help by scaling updates per dimension.



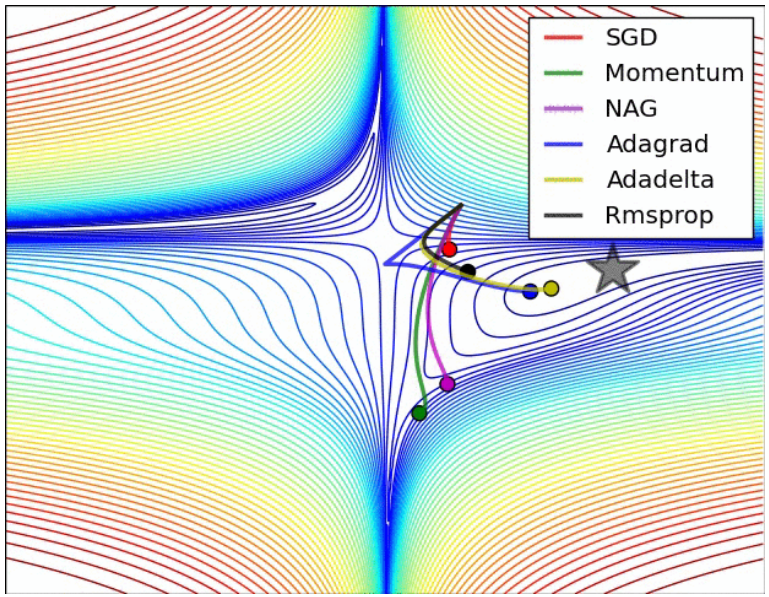
SGD's Isotropy Problem

- SGD treats all dimensions equally (isotropic updates).
- Causes slow progress along shallow gradient directions.
- Inputs with varying scales worsen this problem.
- Adaptive learning rates help by scaling updates per dimension.



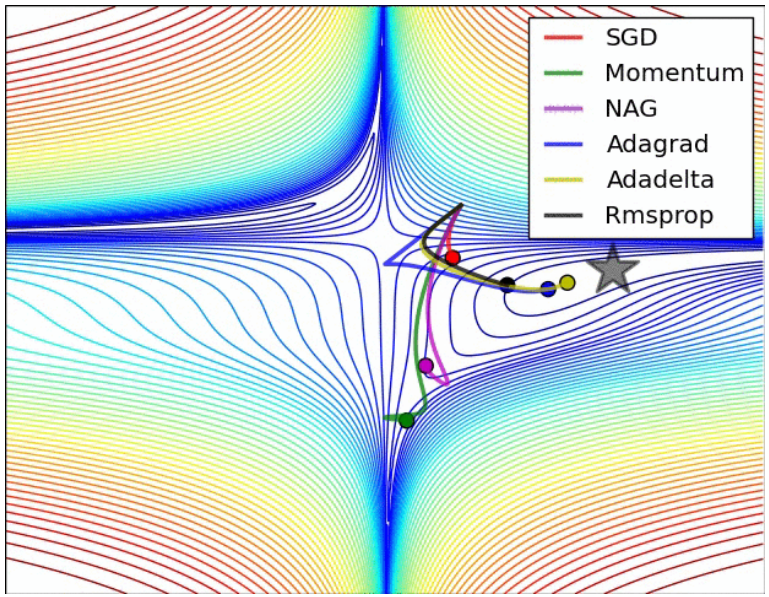
SGD's Isotropy Problem

- SGD treats all dimensions equally (isotropic updates).
- Causes slow progress along shallow gradient directions.
- Inputs with varying scales worsen this problem.
- Adaptive learning rates help by scaling updates per dimension.



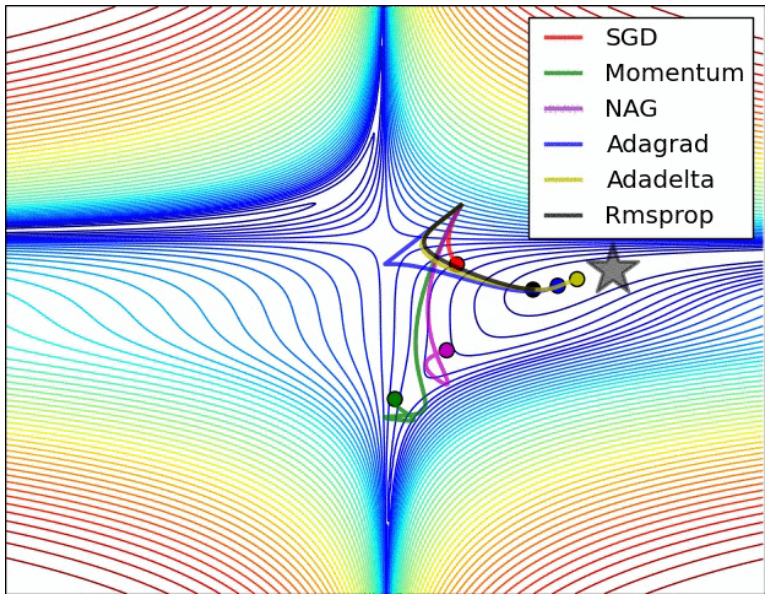
SGD's Isotropy Problem

- SGD treats all dimensions equally (isotropic updates).
- Causes slow progress along shallow gradient directions.
- Inputs with varying scales worsen this problem.
- Adaptive learning rates help by scaling updates per dimension.



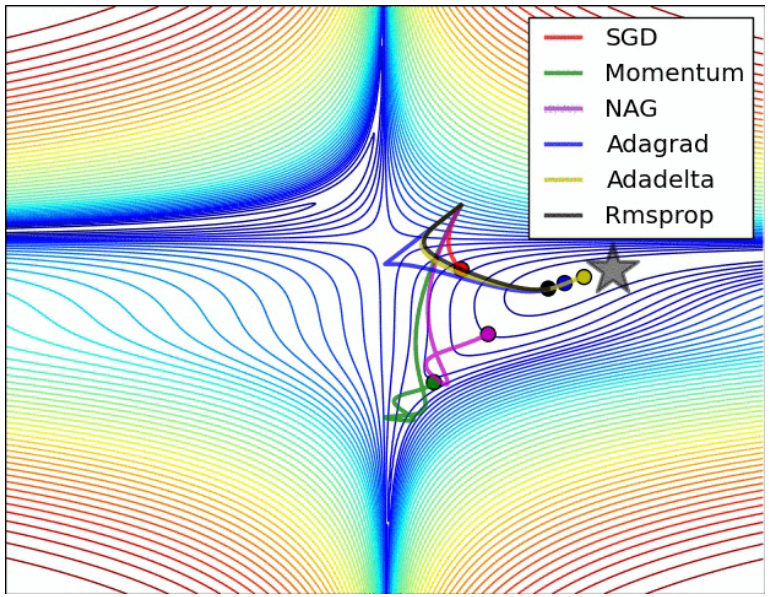
SGD's Isotropy Problem

- SGD treats all dimensions equally (isotropic updates).
- Causes slow progress along shallow gradient directions.
- Inputs with varying scales worsen this problem.
- Adaptive learning rates help by scaling updates per dimension.



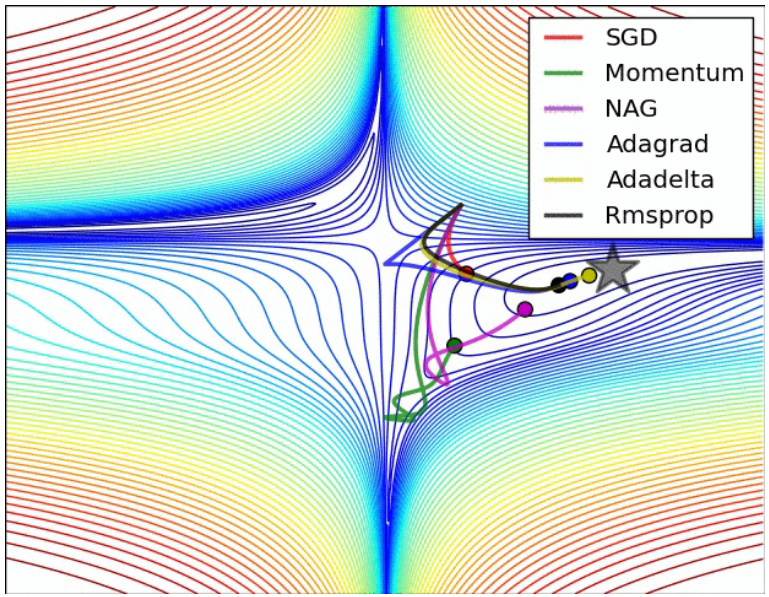
SGD's Isotropy Problem

- SGD treats all dimensions equally (isotropic updates).
- Causes slow progress along shallow gradient directions.
- Inputs with varying scales worsen this problem.
- Adaptive learning rates help by scaling updates per dimension.



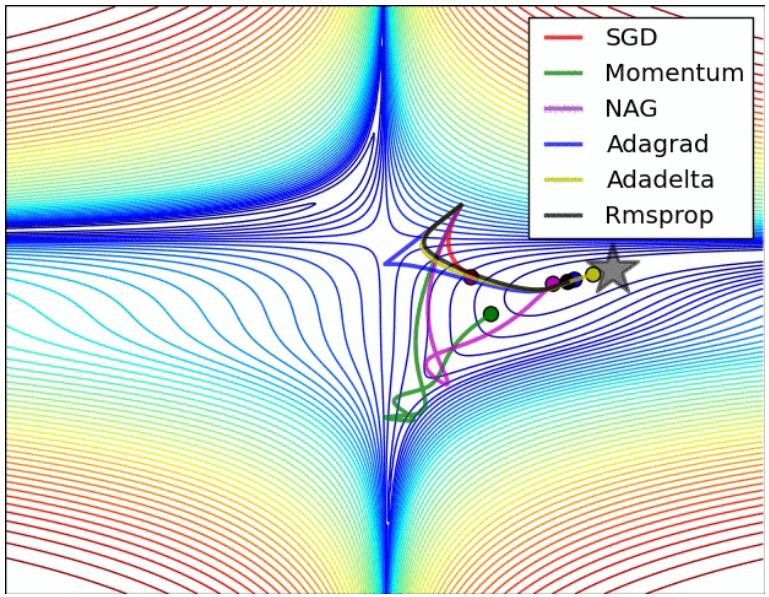
SGD's Isotropy Problem

- SGD treats all dimensions equally (isotropic updates).
- Causes slow progress along shallow gradient directions.
- Inputs with varying scales worsen this problem.
- Adaptive learning rates help by scaling updates per dimension.



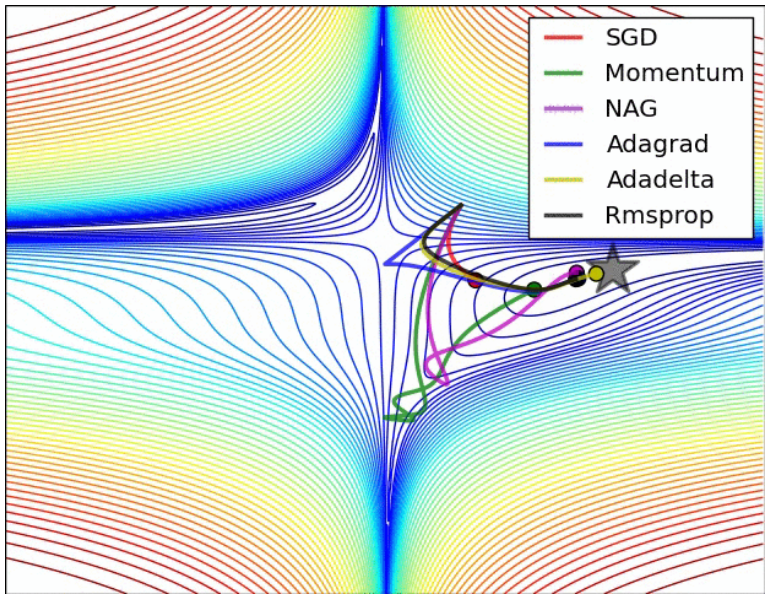
SGD's Isotropy Problem

- SGD treats all dimensions equally (isotropic updates).
- Causes slow progress along shallow gradient directions.
- Inputs with varying scales worsen this problem.
- Adaptive learning rates help by scaling updates per dimension.



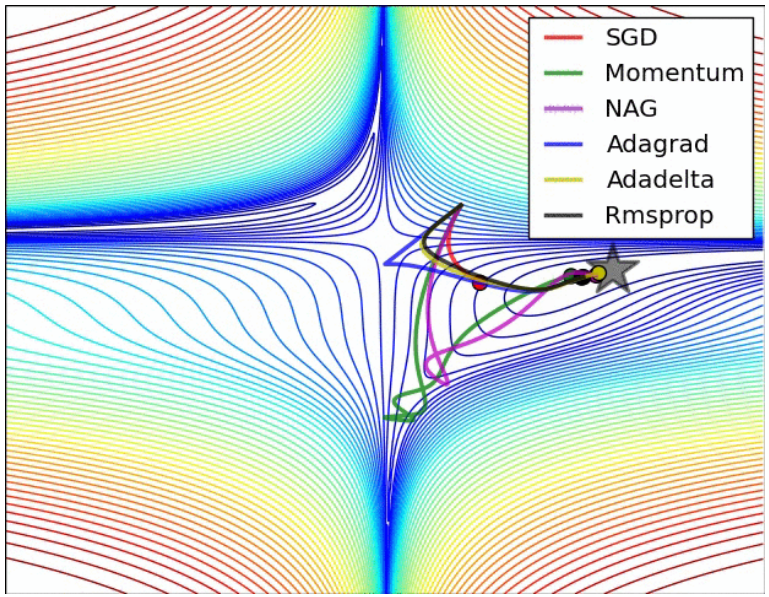
SGD's Isotropy Problem

- SGD treats all dimensions equally (isotropic updates).
- Causes slow progress along shallow gradient directions.
- Inputs with varying scales worsen this problem.
- Adaptive learning rates help by scaling updates per dimension.



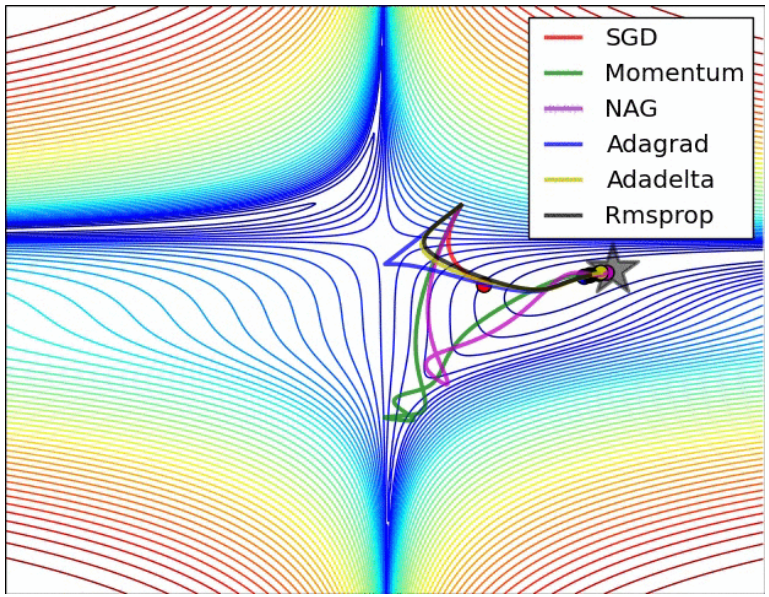
SGD's Isotropy Problem

- SGD treats all dimensions equally (isotropic updates).
- Causes slow progress along shallow gradient directions.
- Inputs with varying scales worsen this problem.
- Adaptive learning rates help by scaling updates per dimension.



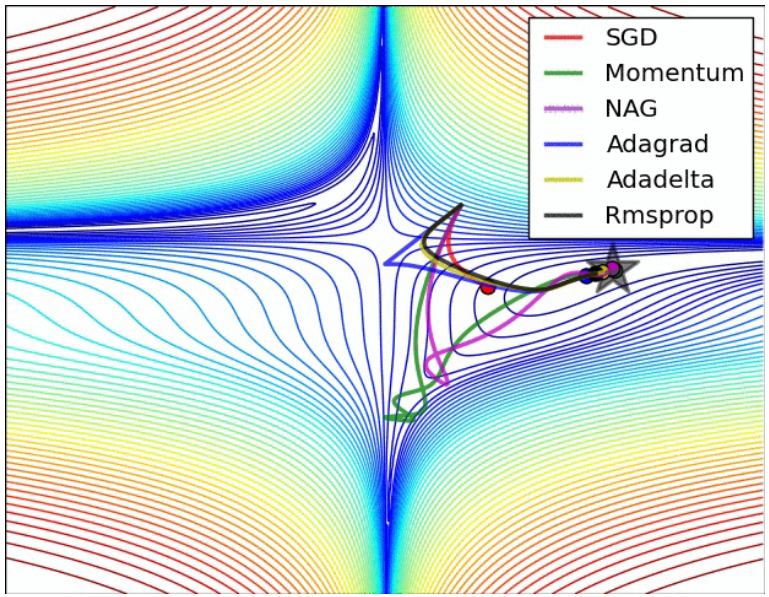
SGD's Isotropy Problem

- SGD treats all dimensions equally (isotropic updates).
- Causes slow progress along shallow gradient directions.
- Inputs with varying scales worsen this problem.
- Adaptive learning rates help by scaling updates per dimension.



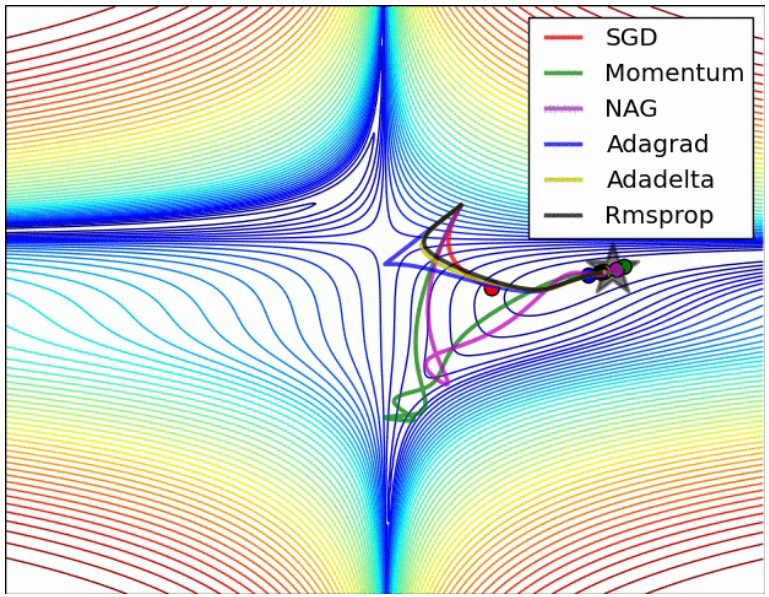
SGD's Isotropy Problem

- SGD treats all dimensions equally (isotropic updates).
- Causes slow progress along shallow gradient directions.
- Inputs with varying scales worsen this problem.
- Adaptive learning rates help by scaling updates per dimension.



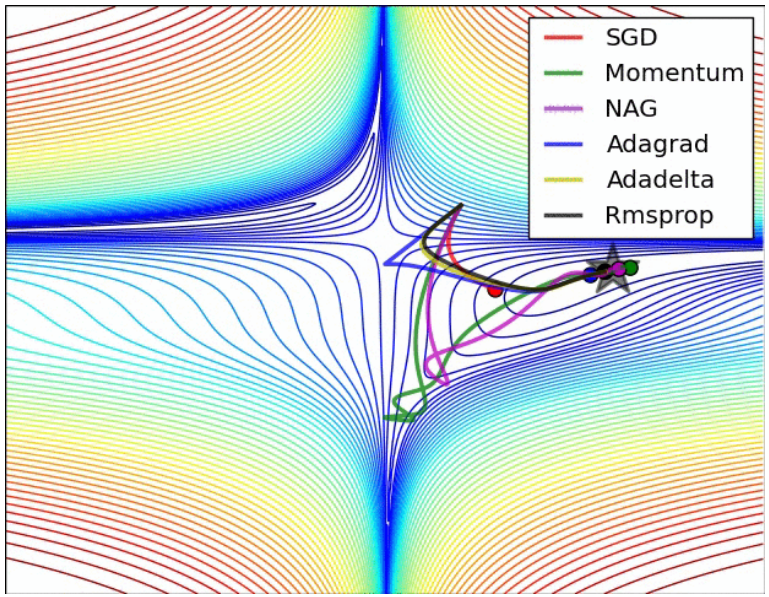
SGD's Isotropy Problem

- SGD treats all dimensions equally (isotropic updates).
- Causes slow progress along shallow gradient directions.
- Inputs with varying scales worsen this problem.
- Adaptive learning rates help by scaling updates per dimension.



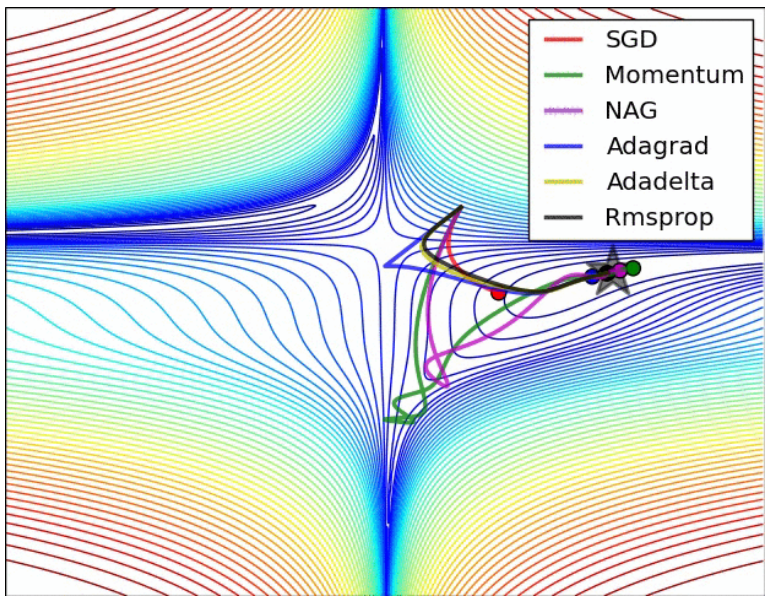
SGD's Isotropy Problem

- SGD treats all dimensions equally (isotropic updates).
- Causes slow progress along shallow gradient directions.
- Inputs with varying scales worsen this problem.
- Adaptive learning rates help by scaling updates per dimension.



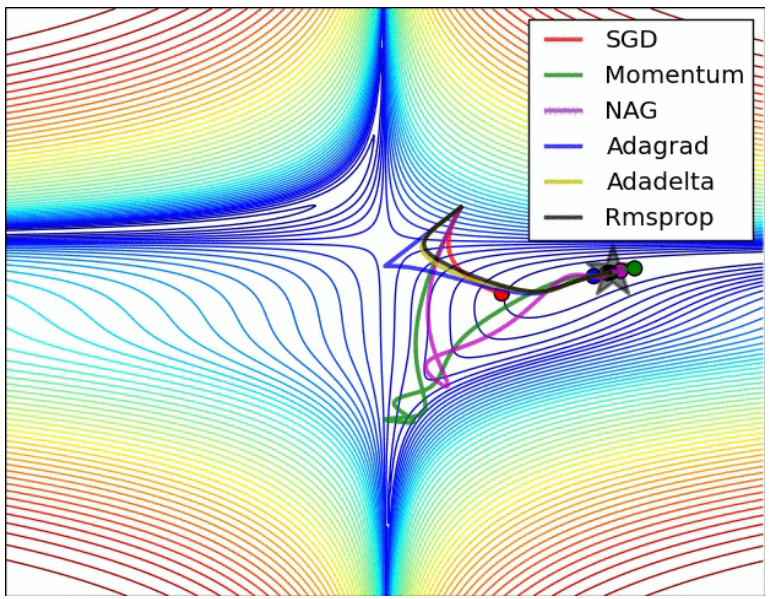
SGD's Isotropy Problem

- SGD treats all dimensions equally (isotropic updates).
- Causes slow progress along shallow gradient directions.
- Inputs with varying scales worsen this problem.
- Adaptive learning rates help by scaling updates per dimension.



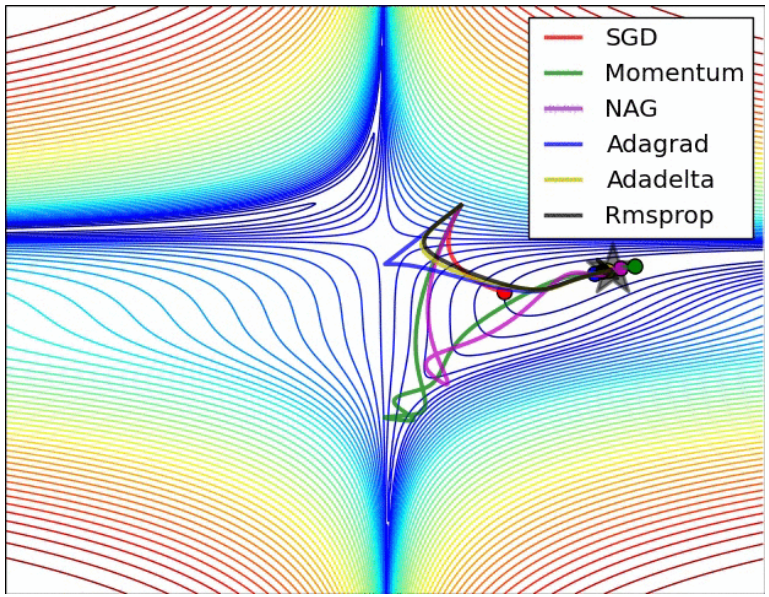
SGD's Isotropy Problem

- SGD treats all dimensions equally (isotropic updates).
- Causes slow progress along shallow gradient directions.
- Inputs with varying scales worsen this problem.
- Adaptive learning rates help by scaling updates per dimension.



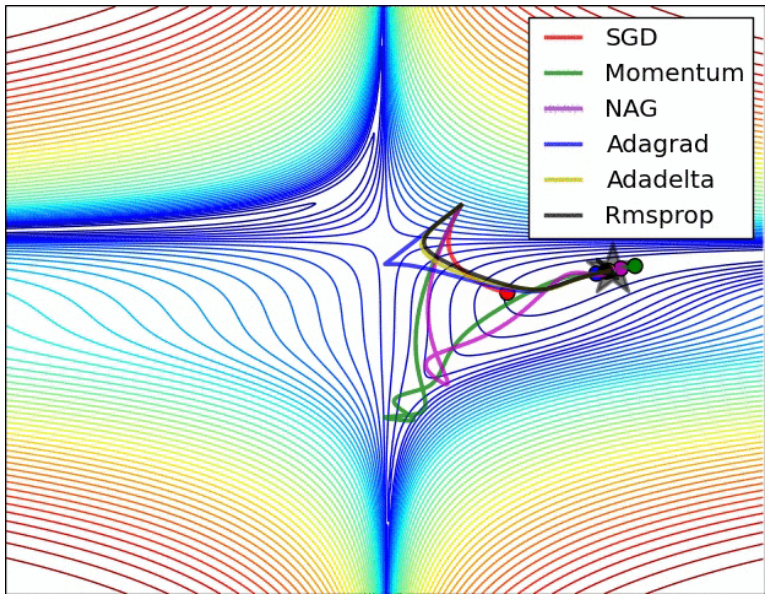
SGD's Isotropy Problem

- SGD treats all dimensions equally (isotropic updates).
- Causes slow progress along shallow gradient directions.
- Inputs with varying scales worsen this problem.
- Adaptive learning rates help by scaling updates per dimension.



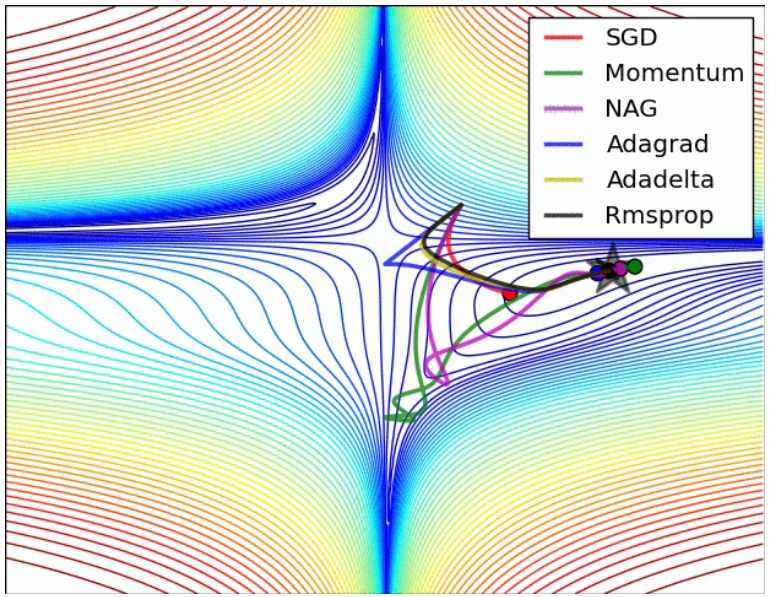
SGD's Isotropy Problem

- SGD treats all dimensions equally (isotropic updates).
- Causes slow progress along shallow gradient directions.
- Inputs with varying scales worsen this problem.
- Adaptive learning rates help by scaling updates per dimension.



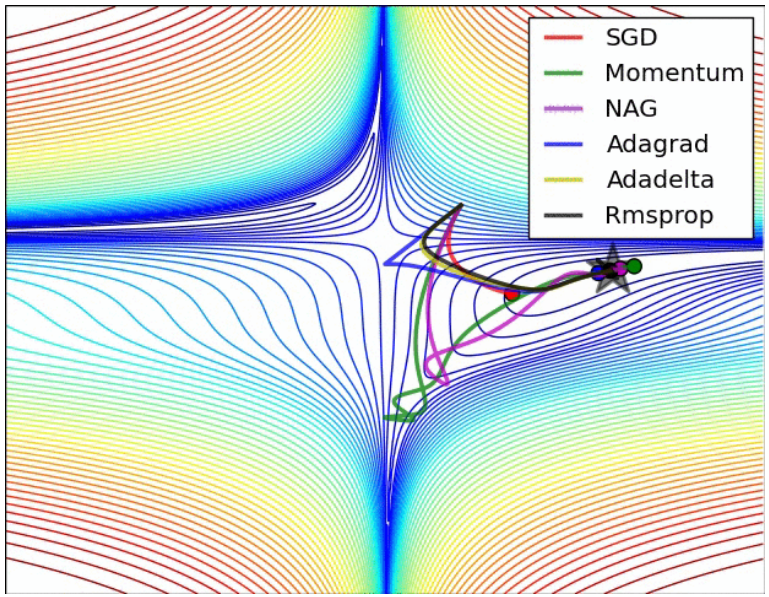
SGD's Isotropy Problem

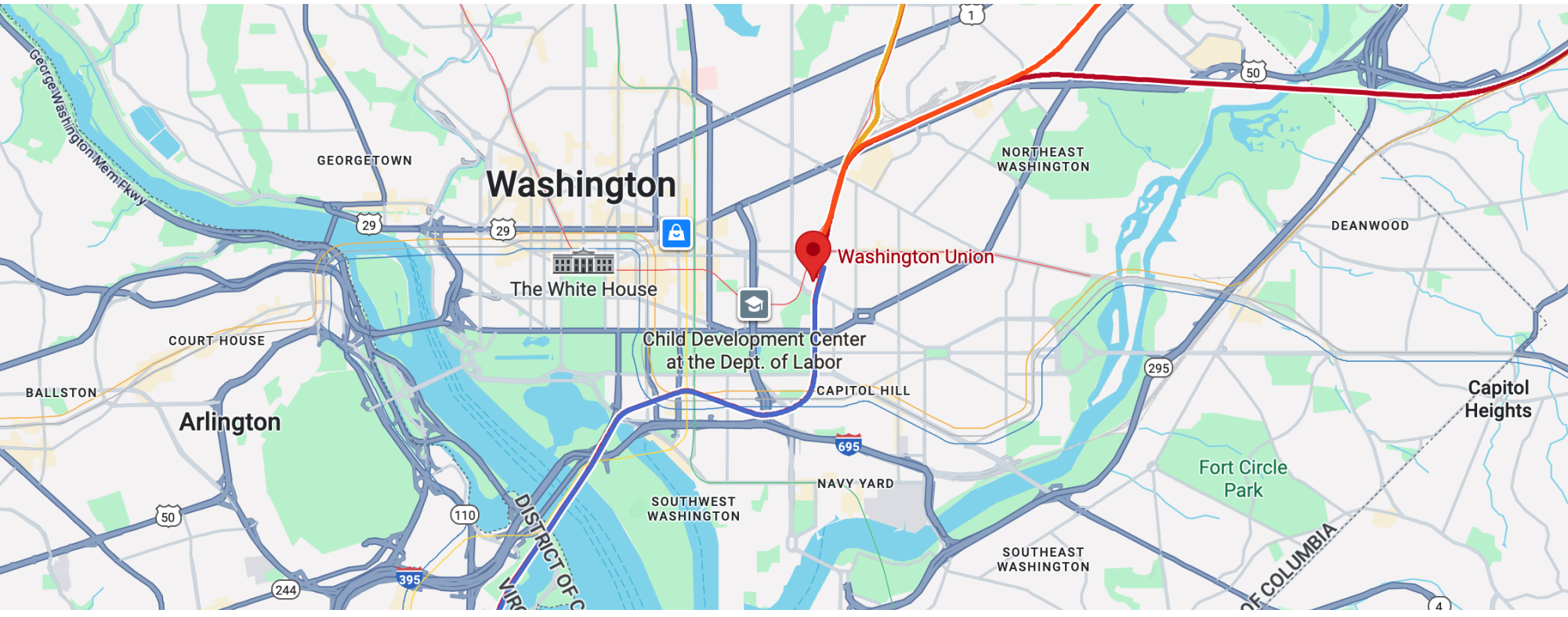
- SGD treats all dimensions equally (isotropic updates).
- Causes slow progress along shallow gradient directions.
- Inputs with varying scales worsen this problem.
- Adaptive learning rates help by scaling updates per dimension.

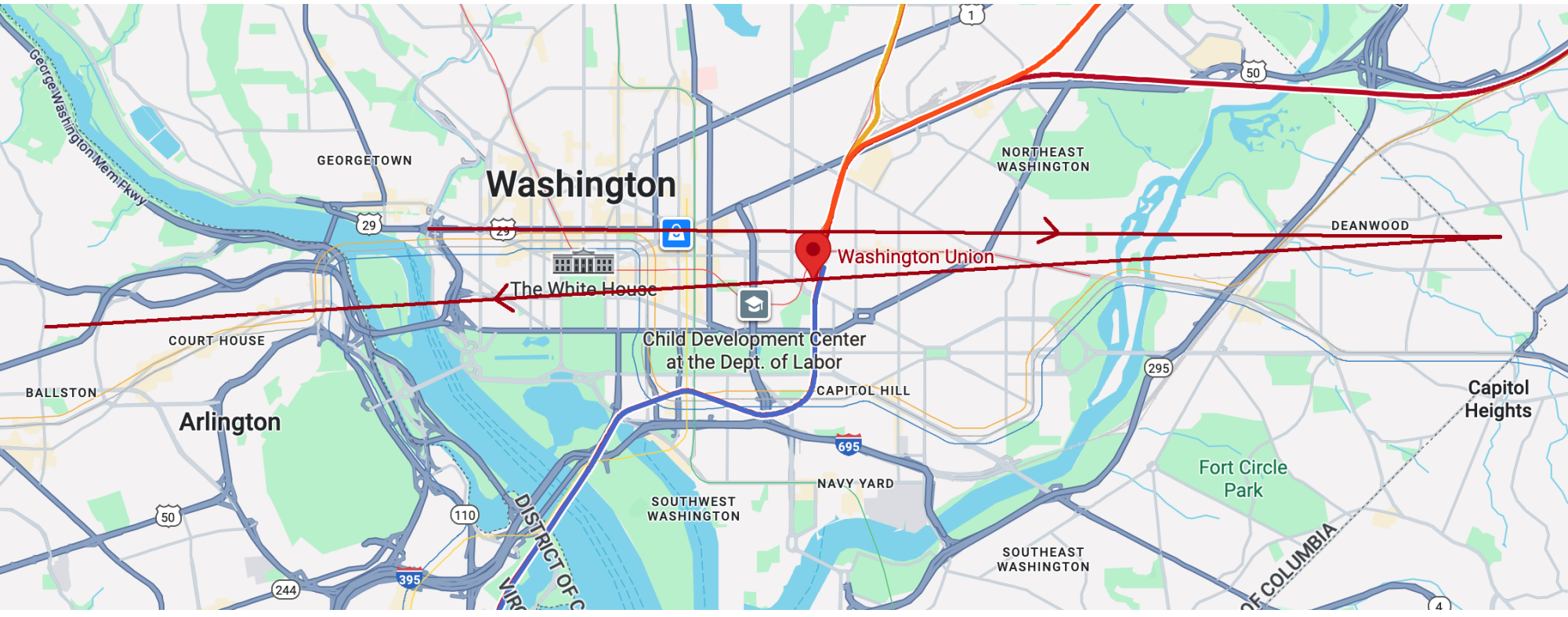


SGD's Isotropy Problem

- SGD treats all dimensions equally (isotropic updates).
- Causes slow progress along shallow gradient directions.
- Inputs with varying scales worsen this problem.
- Adaptive learning rates help by scaling updates per dimension.

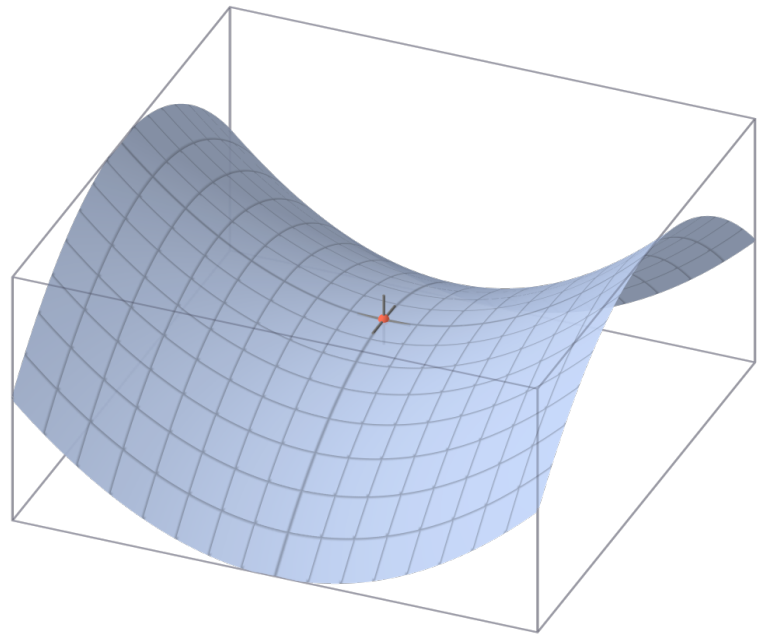



Washington
GEORGETOWN
NORTHEAST WASHINGTON
DEANWOOD
Washington Union
The White House
Child Development Center at the Dept. of Labor
COURT HOUSE
BALLSTON
Arlington
CAPITOL HILL
Capitol Heights
NAVY YARD
SOUTHWEST WASHINGTON
Fort Circle Park
SOUTHEAST WASHINGTON
DISTRICT OF COLUMBIA
29
50
1
295
695
110
395
244
4


Washington
GEORGETOWN
NORTHEAST WASHINGTON
DEANWOOD
Washington Union
The White House
Child Development Center at the Dept. of Labor
COURT HOUSE
BALLSTON
Arlington
CAPITOL HILL
Capitol Heights
NAVY YARD
SOUTHWEST WASHINGTON
Fort Circle Park
SOUTHEAST WASHINGTON
DISTRICT OF COLUMBIA
29
50
1
295
695
110
395
244
4

Understanding Saddle Points

- Saddle points have zero gradient but are not minima or maxima.
- Example: $f(x, y) = x^2 - y^2$ at the origin.
- Gradient descent can get stuck or slow down near saddle points.
- They dominate critical points in high-dimensional networks.



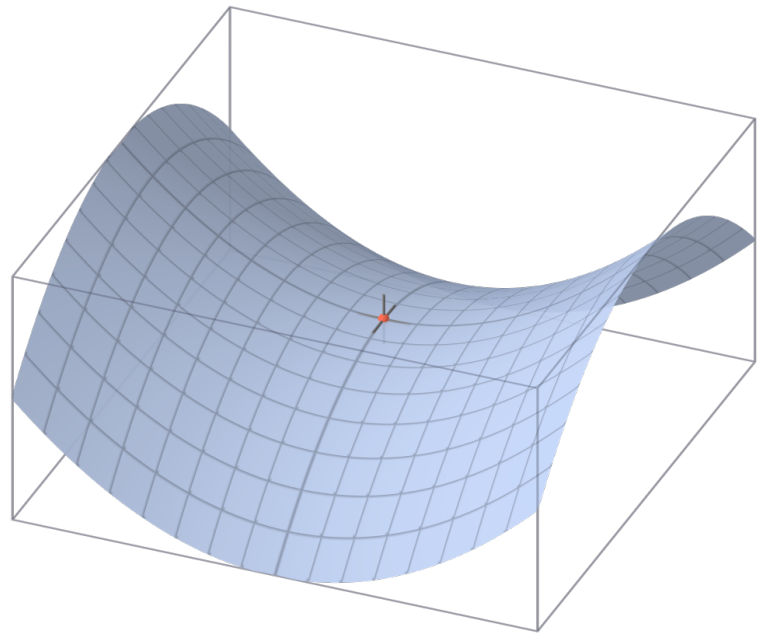
Understanding Saddle Points

- Saddle points have zero gradient but are not minima or maxima.
- Example: $f(x, y) = x^2 - y^2$ at the origin.

$$\frac{\partial f}{\partial x} = 2x \tag{1}$$

$$\frac{\partial f}{\partial y} = 2y \tag{2}$$

$$\tag{3}$$



- Gradient descent can get stuck or slow down near saddle points.
- They dominate critical points in high-dimensional networks.

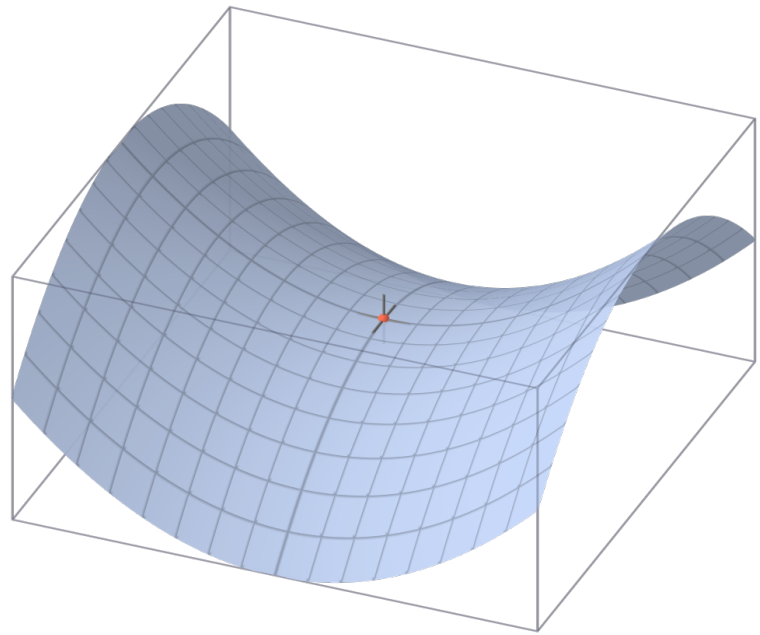
Understanding Saddle Points

- Saddle points have zero gradient but are not minima or maxima.
- Example: $f(x, y) = x^2 - y^2$ at the origin.

$$\frac{\partial f}{\partial x}(0,0) = 0 \tag{1}$$

$$\frac{\partial f}{\partial y}(0,0) = 0 \tag{2}$$

$$\tag{3}$$



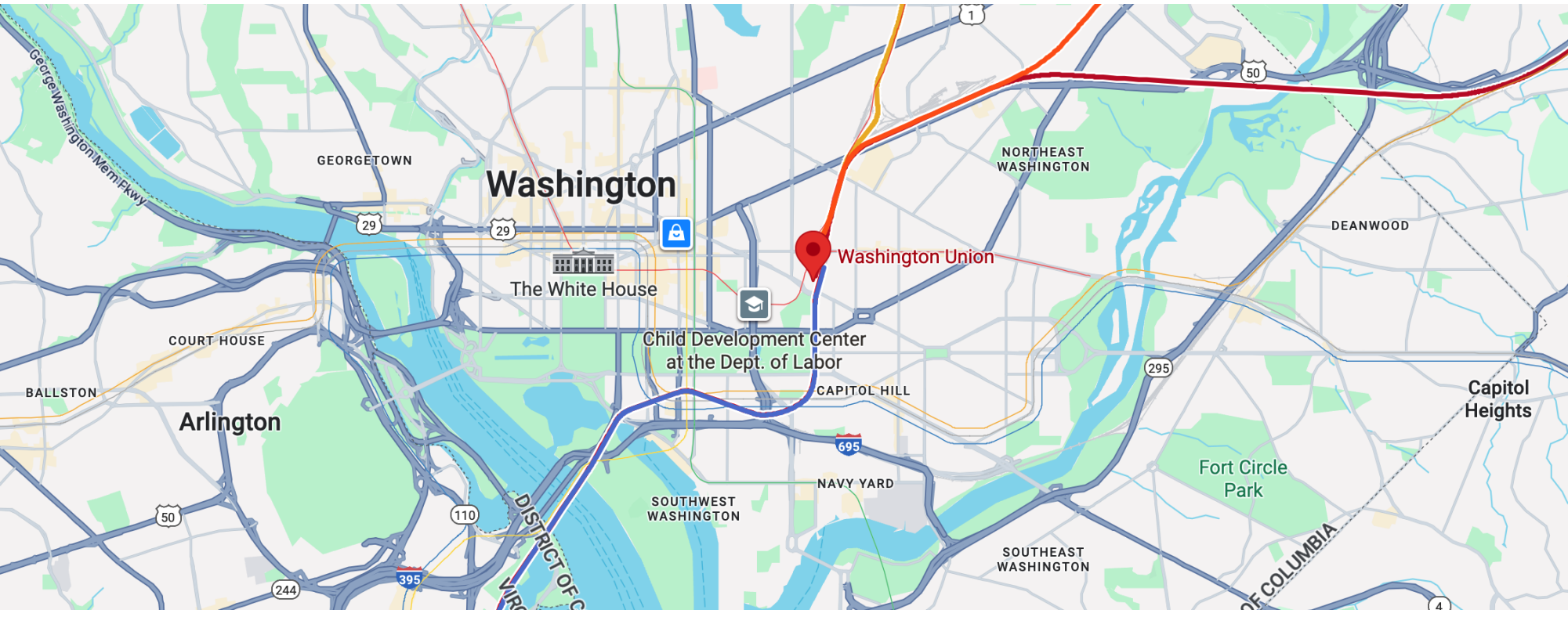
- Gradient descent can get stuck or slow down near saddle points.
- They dominate critical points in high-dimensional networks.

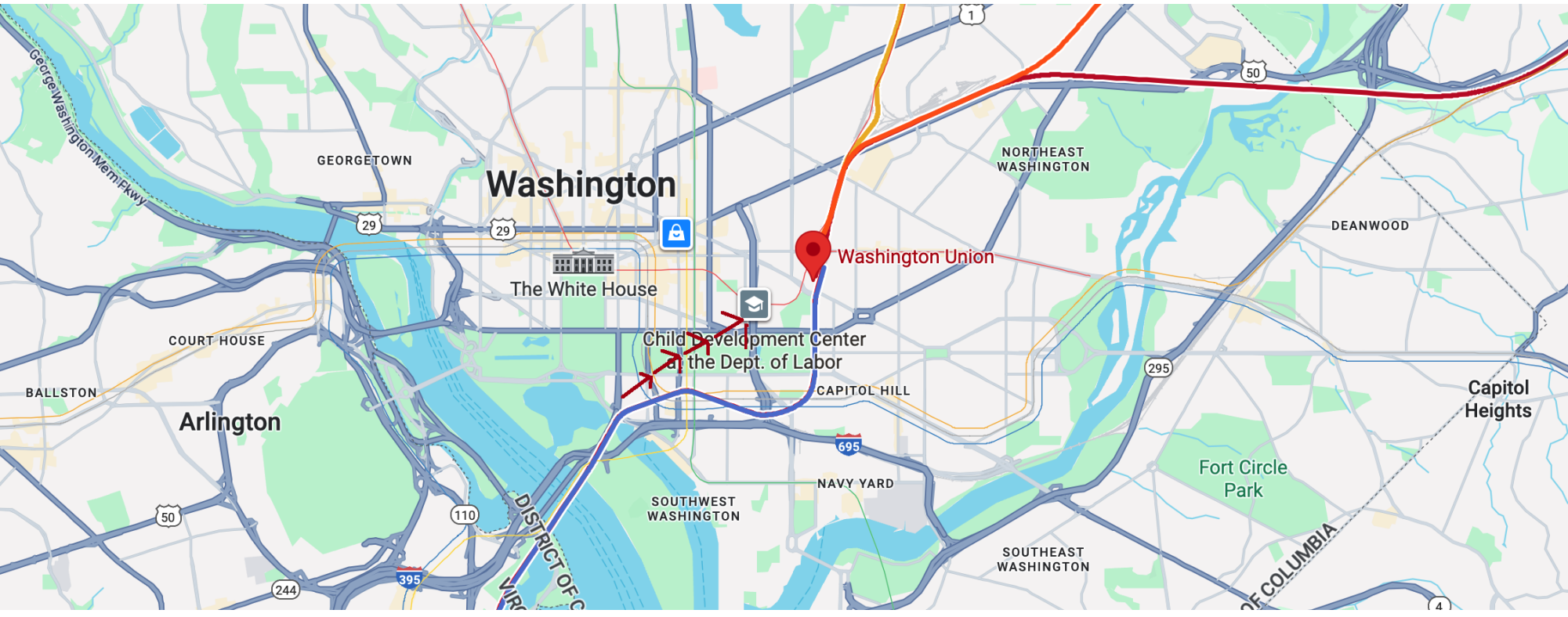
On the Importance of Initialization and Momentum

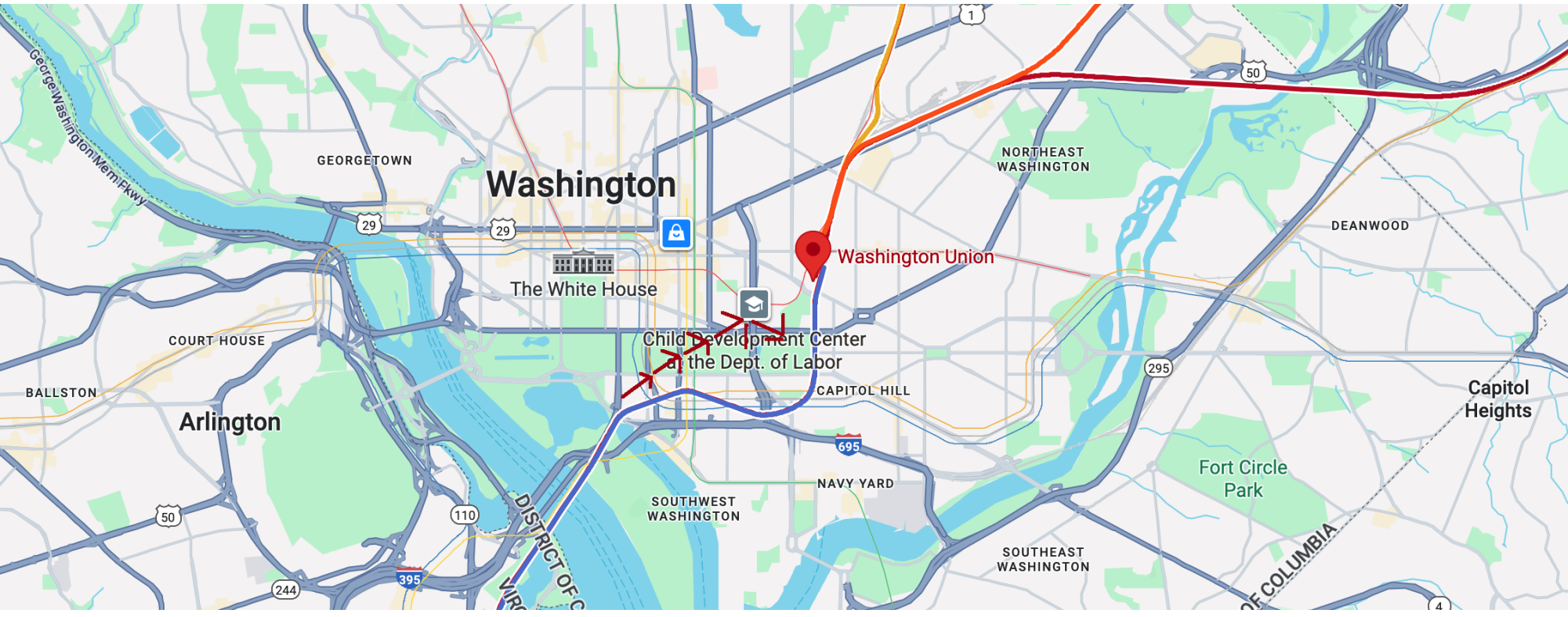
On the importance of initialization and momentum in deep learning

Ilya Sutskever[1] ILYASU@GOOGLE.COM
James Martens JMARTENS@CS.TORONTO.EDU
George Dahl GDAHL@CS.TORONTO.EDU
Geoffrey Hinton HINTON@CS.TORONTO.EDU

- Momentum: If you're going in a direction, keep going in that direction
- Mathematically: accumulate past gradients to create vector of past directions
- Helps maintain direction through flat regions and saddle points


Washington
GEORGETOWN
NORTHEAST WASHINGTON
DEANWOOD
Washington Union
The White House
Child Development Center at the Dept. of Labor
COURT HOUSE
BALLSTON
Arlington
CAPITOL HILL
Capitol Heights
NAVY YARD
SOUTHWEST WASHINGTON
Fort Circle Park
SOUTHEAST WASHINGTON
DISTRICT OF COLUMBIA
29
50
1
295
695
110
395
244
4


Washington
GEORGETOWN
The White House
Washington Union
Child Development Center
at the Dept. of Labor
COURT HOUSE
BALLSTON
Arlington
CAPITOL HILL
NORTHEAST
WASHINGTON
DEANWOOD
Capitol
Heights
Fort Circle
Park
NAVY YARD
SOUTHWEST
WASHINGTON
SOUTHEAST
WASHINGTON
DISTRICT OF COLUMBIA
George Washington Memorial Pkwy
1
50
29
29
295
695
50
110
395
244
4


Washington
GEORGETOWN
The White House
Washington Union
Child Development Center
at the Dept. of Labor
COURT HOUSE
BALLSTON
Arlington
CAPITOL HILL
NORTHEAST
WASHINGTON
DEANWOOD
Capitol
Heights
Fort Circle
Park
NAVY YARD
SOUTHWEST
WASHINGTON
SOUTHEAST
WASHINGTON
DISTRICT OF COLUMBIA
29
50
1
295
695
110
395
244
4

Momentum: Improving Gradient Descent Direction

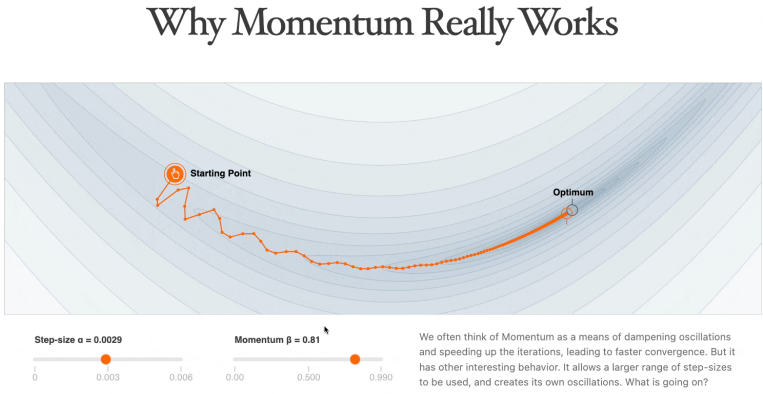
- Momentum averages successive gradients for smoother updates.
- Cancels conflicting directions, accelerates aligned ones.
- Adds a velocity term to keep optimization moving forward.

$$v_t = \tau v_{t-1} + \nabla_\beta \mathcal{L}(\beta_t) \quad (4)$$

$$\beta = \beta' - \tau v_t \quad (5)$$

Momentum: Improving Gradient Descent Direction

- Momentum averages successive gradients for smoother updates.
- Cancels conflicting directions, accelerates aligned ones.
- Adds a velocity term to keep optimization moving forward.



$$v_t = \tau v_{t-1} + \nabla_\beta \mathcal{L}(\beta_t) \quad (4)$$

$$\beta = \beta' - \tau v_t \quad (5)$$

Momentum: Improving Gradient Descent Direction

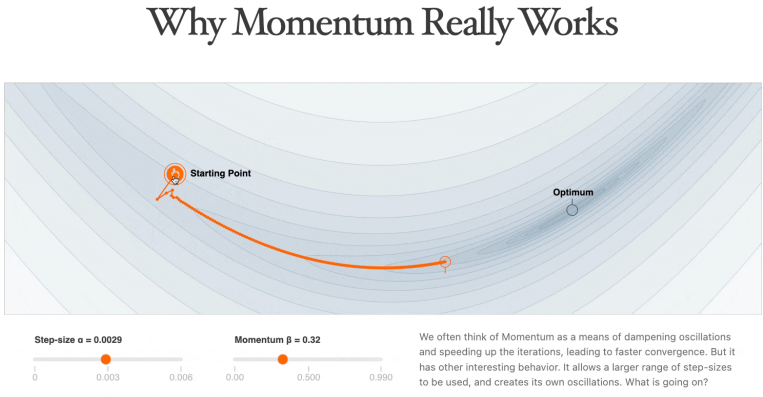
- Momentum averages successive gradients for smoother updates.
- Cancels conflicting directions, accelerates aligned ones.
- Adds a velocity term to keep optimization moving forward.

$$v_t = \tau v_{t-1} + \nabla_\beta \mathcal{L}(\beta_t) \tag{4}$$

$$\beta = \beta' - \tau v_t \tag{5}$$

Momentum: Improving Gradient Descent Direction

- Momentum averages successive gradients for smoother updates.
- Cancels conflicting directions, accelerates aligned ones.
- Adds a velocity term to keep optimization moving forward.

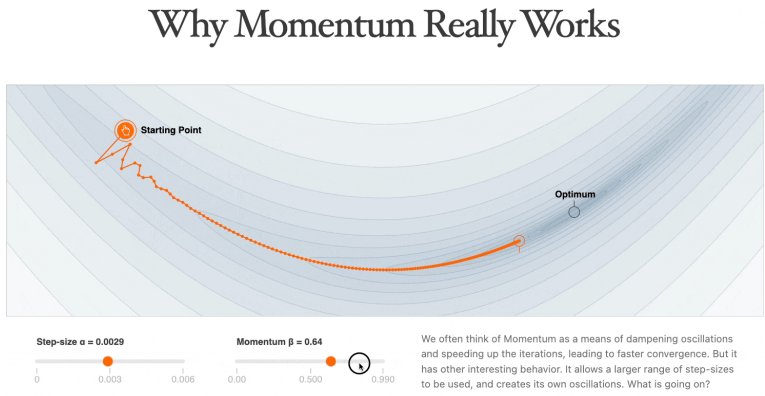


$$v_t = \tau v_{t-1} + \nabla_\beta \mathcal{L}(\beta_t) \quad (4)$$

$$\beta = \beta' - \tau v_t \quad (5)$$

Momentum: Improving Gradient Descent Direction

- Momentum averages successive gradients for smoother updates.
- Cancels conflicting directions, accelerates aligned ones.
- Adds a velocity term to keep optimization moving forward.



$$v_t = \tau v_{t-1} + \nabla_\beta \mathcal{L}(\beta_t) \quad (4)$$

$$\beta = \beta' - \tau v_t \quad (5)$$

Momentum: Improving Gradient Descent Direction

- Momentum averages successive gradients for smoother updates.
- Cancels conflicting directions, accelerates aligned ones.
- Adds a velocity term to keep optimization moving forward.

$$v_t = \tau v_{t-1} + \nabla_\beta \mathcal{L}(\beta_t) \quad (4)$$

$$\beta = \beta' - \tau v_t \quad (5)$$

Momentum: Improving Gradient Descent Direction

- Momentum averages successive gradients for smoother updates.
- Cancels conflicting directions, accelerates aligned ones.
- Adds a velocity term to keep optimization moving forward.

$$v_t = \tau v_{t-1} + \nabla_\beta \mathcal{L}(\beta_t) \quad (4)$$

$$\beta = \beta' - \tau v_t \quad (5)$$

Objective

Position

Gradient Scale and Adaptive Learning Rates

- Gradient magnitude varies greatly during training.
- Large gradients cause unstable updates; small gradients slow learning.
- Normalize gradient magnitude per dimension to stabilize steps.
- Leads to adaptive learning rates per parameter.

AdaGrad and RMSProp: Adaptive Optimizers

- **AdaGrad:**

$$\vec{g}_t \equiv \frac{1}{N} \sum_{i \in \text{minibatch t}} \nabla_\beta \mathscr{L}\left(x^{(i)}, y^{(i)}, \vec{\beta}_t\right) \tag{6}$$

$$\mathbf{G}_t = \sum j \vec{g}_j g_j^\top \tag{7}$$

$$\beta_{t+1} = \beta_t - \frac{\eta}{\sqrt{\epsilon I + \text{diag}(G_t)}} \vec{g}_t \tag{8}$$

- **RMSProp:**

$$\mathbb{E}\left[g_t^2\right] = \gamma \mathbb{E}\left[g_{t-1}^2\right] + (1-\gamma)\left(\frac{\partial \mathscr{L}}{\partial \beta}\right)^2 \tag{9}$$

$$\beta_{t+1} = \beta_t - \frac{\eta}{\sqrt{\mathbb{E}\left[g_t^2\right] + \epsilon}} \frac{\partial \mathscr{L}}{\partial \beta} \tag{10}$$

Adam: Combining Momentum and RMSProp

- Adam merges momentum (first moment) and RMSProp (second moment).
- Maintains running averages of gradients and their squares.
- Corrects bias from zero initialization.
- Adapts learning rates dynamically per parameter.

$$v_t = \gamma_1 v_{t-1} + (1-\gamma_1)\nabla_\beta \mathcal{L}(\beta_t)$$

$$\mathbb{E}\left[g_t^2\right] = \gamma_2 \mathbb{E}\left[g_{t-1}^2\right] + (1-\gamma_2)\left(\frac{\partial \mathcal{L}}{\partial \beta}\right)^2$$

Algorithm 1: *Adam*, our proposed algorithm for stochastic optimization. See section 2 for details, and for a slightly more efficient (but less clear) order of computation. g_t^2 indicates the elementwise square $g_t \odot g_t$. Good default settings for the tested machine learning problems are $\alpha = 0.001$, $\beta_1 = 0.9$, $\beta_2 = 0.999$ and $\epsilon = 10^{-8}$. All operations on vectors are element-wise. With β_1^t and β_2^t we denote β_1 and β_2 to the power t.

Require: α: Stepsize
Require: $\beta_1, \beta_2 \in [0, 1)$: Exponential decay rates for the moment estimates
Require: $f(\theta)$: Stochastic objective function with parameters θ
Require: θ_0: Initial parameter vector
  $m_0 \leftarrow 0$ (Initialize 1st moment vector)
  $v_0 \leftarrow 0$ (Initialize 2nd moment vector)
  $t \leftarrow 0$ (Initialize timestep)
  while θ_t not converged **do**
    $t \leftarrow t + 1$
    $g_t \leftarrow \nabla_\theta f_t(\theta_{t-1})$ (Get gradients w.r.t. stochastic objective at timestep t)
    $m_t \leftarrow \beta_1 \cdot m_{t-1} + (1 - \beta_1) \cdot g_t$ (Update biased first moment estimate)
    $v_t \leftarrow \beta_2 \cdot v_{t-1} + (1 - \beta_2) \cdot g_t^2$ (Update biased second raw moment estimate)
    $\widehat{m}_t \leftarrow m_t / (1 - \beta_1^t)$ (Compute bias-corrected first moment estimate)
    $\widehat{v}_t \leftarrow v_t / (1 - \beta_2^t)$ (Compute bias-corrected second raw moment estimate)
    $\theta_t \leftarrow \theta_{t-1} - \alpha \cdot \widehat{m}_t / (\sqrt{\widehat{v}_t} + \epsilon)$ (Update parameters)
  end while
  return θ_t (Resulting parameters)

Adam: Combining Momentum and RMSProp

- Adam merges momentum (first moment) and RMSProp (second moment).
- Maintains running averages of gradients and their squares.
- Corrects bias from zero initialization.
- Adapts learning rates dynamically per parameter.

$$v_t = \gamma_1 v_{t-1} + (1-\gamma_1)\nabla_\beta \mathcal{L}(\beta_t)$$

$$\mathbb{E}\left[g_t^2\right] = \gamma_2 \mathbb{E}\left[g_{t-1}^2\right] + (1-\gamma_2)\left(\frac{\partial \mathcal{L}}{\partial \beta}\right)^2$$

Algorithm 1: *Adam*, our proposed algorithm for stochastic optimization. See section 2 for details, and for a slightly more efficient (but less clear) order of computation. g_t^2 indicates the elementwise square $g_t \odot g_t$. Good default settings for the tested machine learning problems are $\alpha = 0.001$, $\beta_1 = 0.9$, $\beta_2 = 0.999$ and $\epsilon = 10^{-8}$. All operations on vectors are element-wise. With β_1^t and β_2^t we denote β_1 and β_2 to the power t.

Require: α: Stepsize
Require: $\beta_1, \beta_2 \in [0, 1)$: Exponential decay rates for the moment estimates
Require: $f(\theta)$: Stochastic objective function with parameters θ
Require: θ_0: Initial parameter vector
  $m_0 \leftarrow 0$ (Initialize 1st moment vector)
  $v_0 \leftarrow 0$ (Initialize 2nd moment vector)
  $t \leftarrow 0$ (Initialize timestep)
  while θ_t not converged **do**
    $t \leftarrow t + 1$
    $g_t \leftarrow \nabla_\theta f_t(\theta_{t-1})$ (Get gradients w.r.t. stochastic objective at timestep t)
    $m_t \leftarrow \beta_1 \cdot m_{t-1} + (1 - \beta_1) \cdot g_t$ (Update biased first moment estimate)
    $v_t \leftarrow \beta_2 \cdot v_{t-1} + (1 - \beta_2) \cdot g_t^2$ (Update biased second raw moment estimate)
    $\widehat{m}_t \leftarrow m_t / (1 - \beta_1^t)$ (Compute bias-corrected first moment estimate)
    $\widehat{v}_t \leftarrow v_t / (1 - \beta_2^t)$ (Compute bias-corrected second raw moment estimate)
    $\theta_t \leftarrow \theta_{t-1} - \alpha \cdot \widehat{m}_t / (\sqrt{\widehat{v}_t} + \epsilon)$ (Update parameters)
  end while
  return θ_t (Resulting parameters)

Diff Notation

Adam: Combining Momentum and RMSProp

- Adam merges momentum (first moment) and RMSProp (second moment).
- Maintains running averages of gradients and their squares.
- Corrects bias from zero initialization.
- Adapts learning rates dynamically per parameter.

$$v_t = \gamma_1 v_{t-1} + (1-\gamma_1)\nabla_\beta \mathcal{L}(\beta_t)$$

$$\mathbb{E}\left[g_t^2\right] = \gamma_2 \mathbb{E}\left[g_{t-1}^2\right] + (1-\gamma_2)\left(\frac{\partial \mathcal{L}}{\partial \beta}\right)^2$$

Algorithm 1: *Adam*, our proposed algorithm for stochastic optimization. See section 2 for details, and for a slightly more efficient (but less clear) order of computation. g_t^2 indicates the elementwise square $g_t \odot g_t$. Good default settings for the tested machine learning problems are $\alpha = 0.001$, $\beta_1 = 0.9$, $\beta_2 = 0.999$ and $\epsilon = 10^{-8}$. All operations on vectors are element-wise. With β_1^t and β_2^t we denote β_1 and β_2 to the power t.

Require: α: Stepsize
Require: $\beta_1, \beta_2 \in [0, 1)$: Exponential decay rates for the moment estimates
Require: $f(\theta)$: Stochastic objective function with parameters θ
Require: θ_0: Initial parameter vector
  $m_0 \leftarrow 0$ (Initialize 1st moment vector)
  $v_0 \leftarrow 0$ (Initialize 2nd moment vector)
  $t \leftarrow 0$ (Initialize timestep)
  while θ_t not converged **do**
    $t \leftarrow t + 1$
    $g_t \leftarrow \nabla_\theta f_t(\theta_{t-1})$ (Get gradients w.r.t. stochastic objective at timestep t)
    $m_t \leftarrow \beta_1 \cdot m_{t-1} + (1 - \beta_1) \cdot g_t$ (Update biased first moment estimate)
    $v_t \leftarrow \beta_2 \cdot v_{t-1} + (1 - \beta_2) \cdot g_t^2$ (Update biased second raw moment estimate)
    $\widehat{m}_t \leftarrow m_t / (1 - \beta_1^t)$ (Compute bias-corrected first moment estimate)
    $\widehat{v}_t \leftarrow v_t / (1 - \beta_2^t)$ (Compute bias-corrected second raw moment estimate)
    $\theta_t \leftarrow \theta_{t-1} - \alpha \cdot \widehat{m}_t / (\sqrt{\widehat{v}_t} + \epsilon)$ (Update parameters)
  end while
  return θ_t (Resulting parameters)

Diff Notation

Huh?

Adam Algorithm Overview

- Bias-corrected estimates:

$$\hat{v}_t = \frac{m_t}{1-\gamma_1^t}, \quad \widehat{\mathbb{E}\left[g_t^2\right]} = \frac{v_t}{1-\gamma_2^t} \tag{11}$$

- Parameter update:

$$\vec{\beta}_{t+1} = \vec{\beta}_t - \frac{\eta}{\sqrt{\widehat{\mathbb{E}\left[g_t^2\right]} + \epsilon}} \hat{v}_t \tag{12}$$

- Combines momentum and adaptive learning, widely used.

Collapse reply thread

ıd influential finding

ICLR 2019 Conference Paper939 Area Chair1

05 Dec 2018, 09:49 (modified: 20 Dec 2018, 20:08) ICLR 2019 Conference Paper939 Meta Review Readers: Everyone Show Revisions

Metareview: Evaluating this paper is somewhat awkward because it has already been through multiple reviewing cycles, and in the meantime, the trick has already become widely adopted and inspired interesting follow-up work. Much of the paper is devoted to reviewing this follow-up work. I think it's clearly time for this to be made part of the published literature, so I recommend acceptance. (And all reviewers are in agreement that the paper ought to be accepted.)

The paper proposes, in the context of Adam, to apply literal weight decay in place of L2 regularization. An impressively thorough set of experiments are given to demonstrate the improved generalization performance, as well as a decoupling of the hyperparameters.

Previous versions of the paper suffered from a lack of theoretical justification for the proposed method. Ordinarily, in such cases, one would worry that the improved results could be due to some sort of experimental confound. But AdamW has been validated by so many other groups on a range of domains that the improvement is well established. And other researchers have offered possible explanations for the improvement.

Recommendation: Accept (Poster)

Confidence: 5: The area chair is absolutely certain

AdamW: Weight Decay Corrected Adam

- AdamW separates weight decay from L_2 regularization.
- Traditional Adam mixes decay with gradient updates.
- Weight decay applied directly to weights improves generalization.

$$\vec{\beta}_{t+1} = \vec{\beta}_t - \eta \left(\frac{1}{\sqrt{\widehat{\mathbb{E}\left[g_t^2\right]}} + \epsilon} \hat{v}_t + \lambda \vec{\beta}_t \right) \tag{13}$$

Adam

for $t = 1$ to $\ldots$ do
  if *maximize* :
    $g_t \leftarrow -\nabla_\theta f_t(\theta_{t-1})$
  else
    $g_t \leftarrow \nabla_\theta f_t(\theta_{t-1})$
  if $\lambda \neq 0$
    $g_t \leftarrow g_t + \lambda \theta_{t-1}$
  $m_t \leftarrow \beta_1 m_{t-1} + (1 - \beta_1) g_t$
  $v_t \leftarrow \beta_2 v_{t-1} + (1 - \beta_2) g_t^2$
  $\widehat{m_t} \leftarrow m_t / (1 - \beta_1^t)$
  $\widehat{v_t} \leftarrow v_t / (1 - \beta_2^t)$
  if *amsgrad*
    $\widehat{v_t}^{max} \leftarrow \max(\widehat{v_t}^{max}, \widehat{v_t})$
    $\theta_t \leftarrow \theta_{t-1} - \gamma \widehat{m_t} / (\sqrt{\widehat{v_t}^{max}} + \epsilon)$
  else
    $\theta_t \leftarrow \theta_{t-1} - \gamma \widehat{m_t} / (\sqrt{\widehat{v_t}} + \epsilon)$

AdamW

for $t = 1$ to $\ldots$ do
  if *maximize* :
    $g_t \leftarrow -\nabla_\theta f_t(\theta_{t-1})$
  else
    $g_t \leftarrow \nabla_\theta f_t(\theta_{t-1})$
  $\theta_t \leftarrow \theta_{t-1} - \gamma \lambda \theta_{t-1}$
  $m_t \leftarrow \beta_1 m_{t-1} + (1 - \beta_1) g_t$
  $v_t \leftarrow \beta_2 v_{t-1} + (1 - \beta_2) g_t^2$
  $\widehat{m_t} \leftarrow m_t / (1 - \beta_1^t)$
  $\widehat{v_t} \leftarrow v_t / (1 - \beta_2^t)$
  if *amsgrad*
    $\widehat{v_t}^{max} \leftarrow \max(\widehat{v_t}^{max}, \widehat{v_t})$
    $\theta_t \leftarrow \theta_t - \gamma \widehat{m_t} / (\sqrt{\widehat{v_t}^{max}} + \epsilon)$
  else
    $\theta_t \leftarrow \theta_t - \gamma \widehat{m_t} / (\sqrt{\widehat{v_t}} + \epsilon)$

Summary and Takeaways

- SGD is foundational but has limitations: isotropy, noise, and saddle points.
- Momentum accelerates optimization and helps overcome plateaus.
- Adaptive methods like RMSProp scale learning rates per parameter.
- Adam combines momentum and adaptive scaling.
- AdamW improves on Adam by properly handling weight decay.