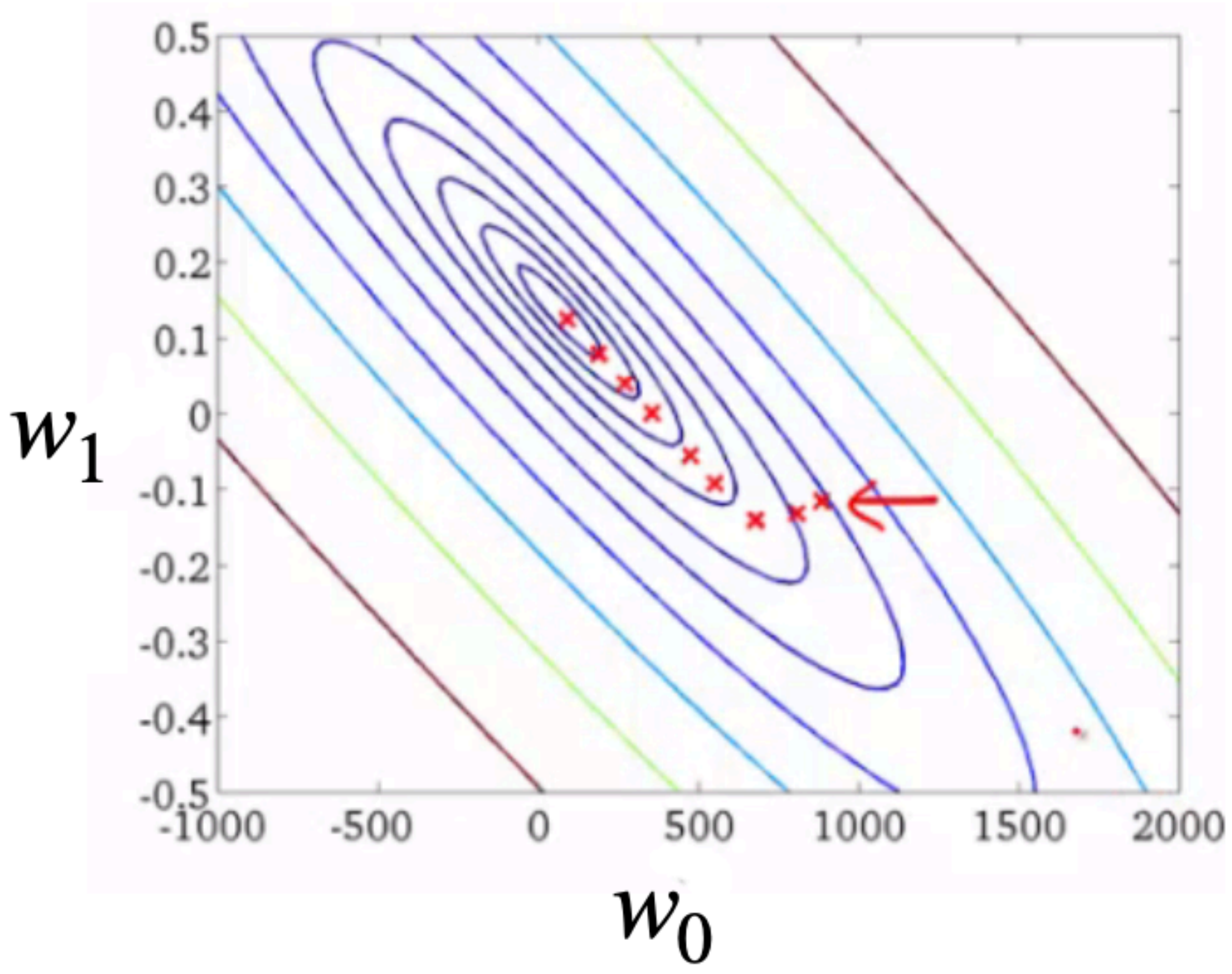
Exploring Stochastic Gradient Descent and its Modifications

Kevin Kleiner

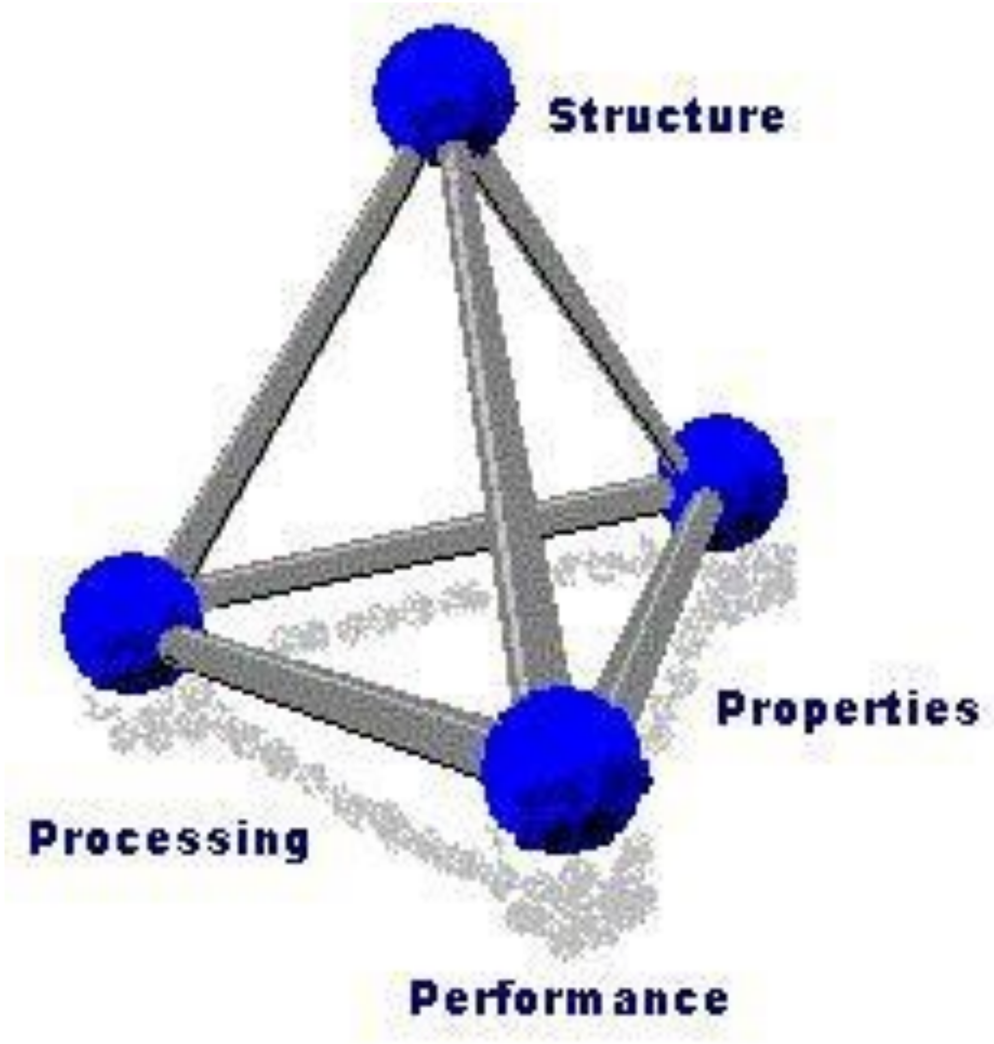
Algorithm Interest Group on March 22, 2021



Model Fitting and Optimization

- From linear models to neural networks, here are some problems which call for model fitting

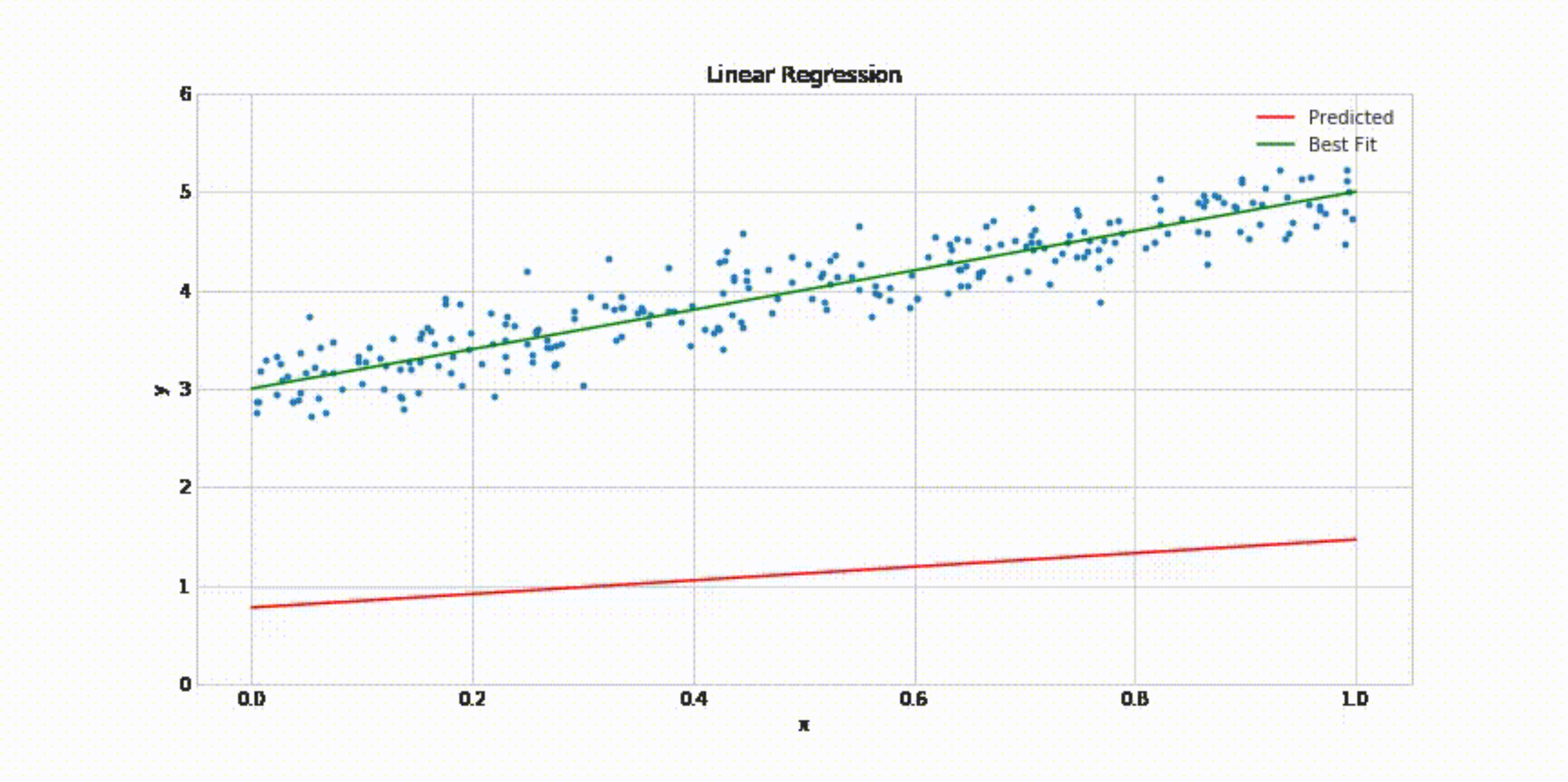
Collected training data	Optimized model
1. Crystal structures and respective optical properties	1. Structure-property relationships to inform design
2. Time series of relative positions of celestial objects	2. Planetary trajectories to inform mapping
3. Time series of many companies' stock prices	3. Stock predictions to inform investing



Which model fitting problems have you encountered in your work?

Model Fitting and Optimization

- All these models involve "weights" that should correctly map the inputs to an output
- We need a rigorous, systematic method to determine the "best possible" model weights



Animation from: https://hackernoon.com/visualizing-linear-regression-with-pytorch-9261f49edb09

Optimize Model Weights with Gradient Descent

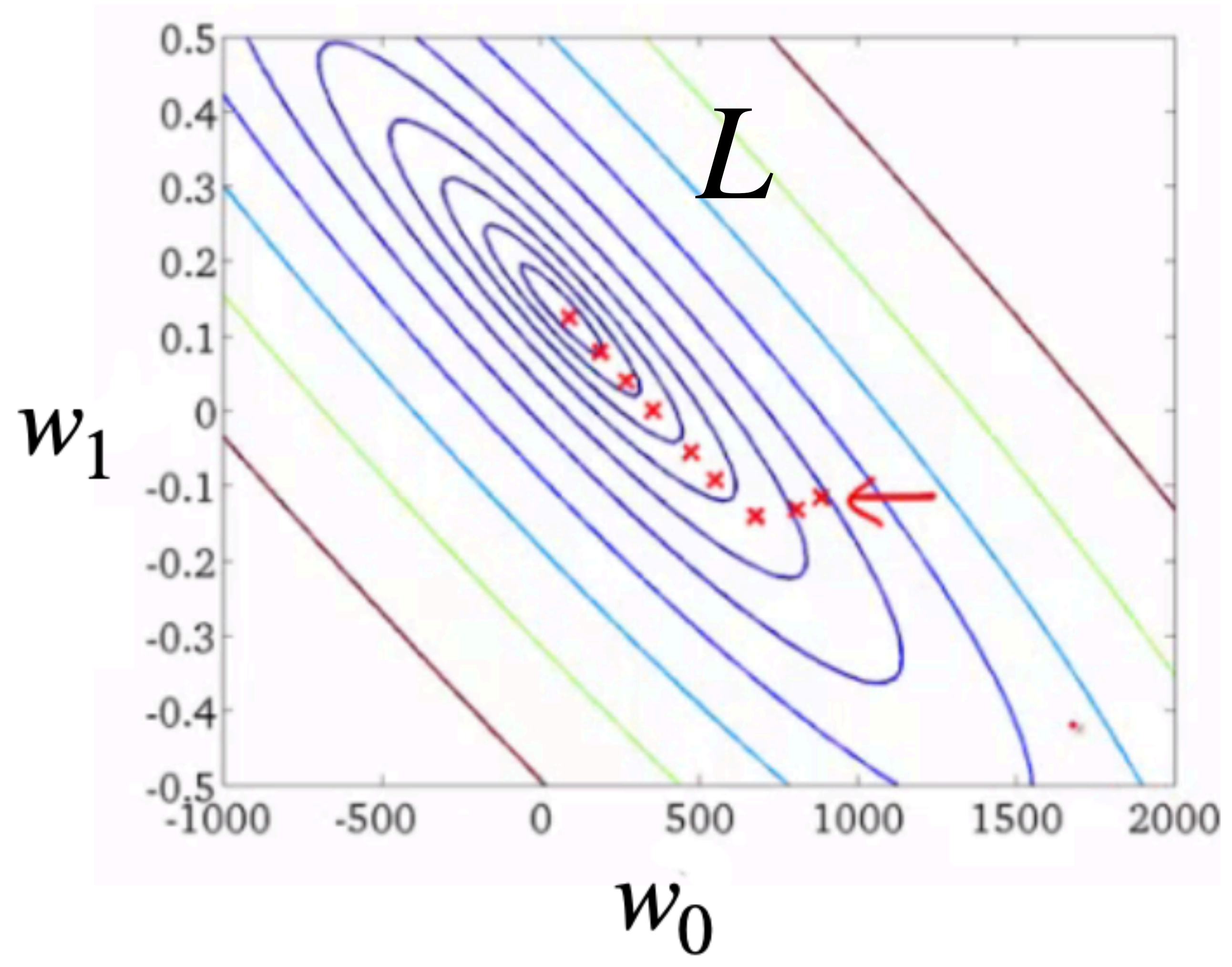
- Introducing the most popular approach,

$$w_{t+1} = w_t - \eta \nabla_{w_t} L(\{x_i, y_i\}; w_t)$$

- w_{t+1}: Weights at iteration $t+1$
- w_t: Weights at iteration t
- η: Step size: $0 < \eta < 1$
- L: Loss function
- $\{x_i, y_i\}$: Set of all observations/ training data

$$L_{LS}(\{x_i, y_i\}; w_t) \equiv \sum_i (\hat{y}_i(x_i; w_t) - y_i)^2$$

- L_{LS}: Least squares loss function
- $\hat{y}_i(x_i; w_t)$: Model-predicted y using x and w_t



Each Gradient Evaluation can be Expensive

- Say we want to fit a linear model $\hat{y} = w_0 + w_1 x_1 + w_2 x_2 + \cdots + w_N x_N$ to $N \sim 10^6$ data points

$$\nabla_{w_t} L = \nabla_{w_t} (\hat{y}_1(x_1; w_t) - y_1)^2 + \nabla_{w_t} (\hat{y}_2(x_2; w_t) - y_2)^2 + \cdots + \nabla_{w_t} (\hat{y}_N(x_N; w_t) - y_N)^2$$

Derivatives with respect to all N weights

Gradient operates on all N residual functions

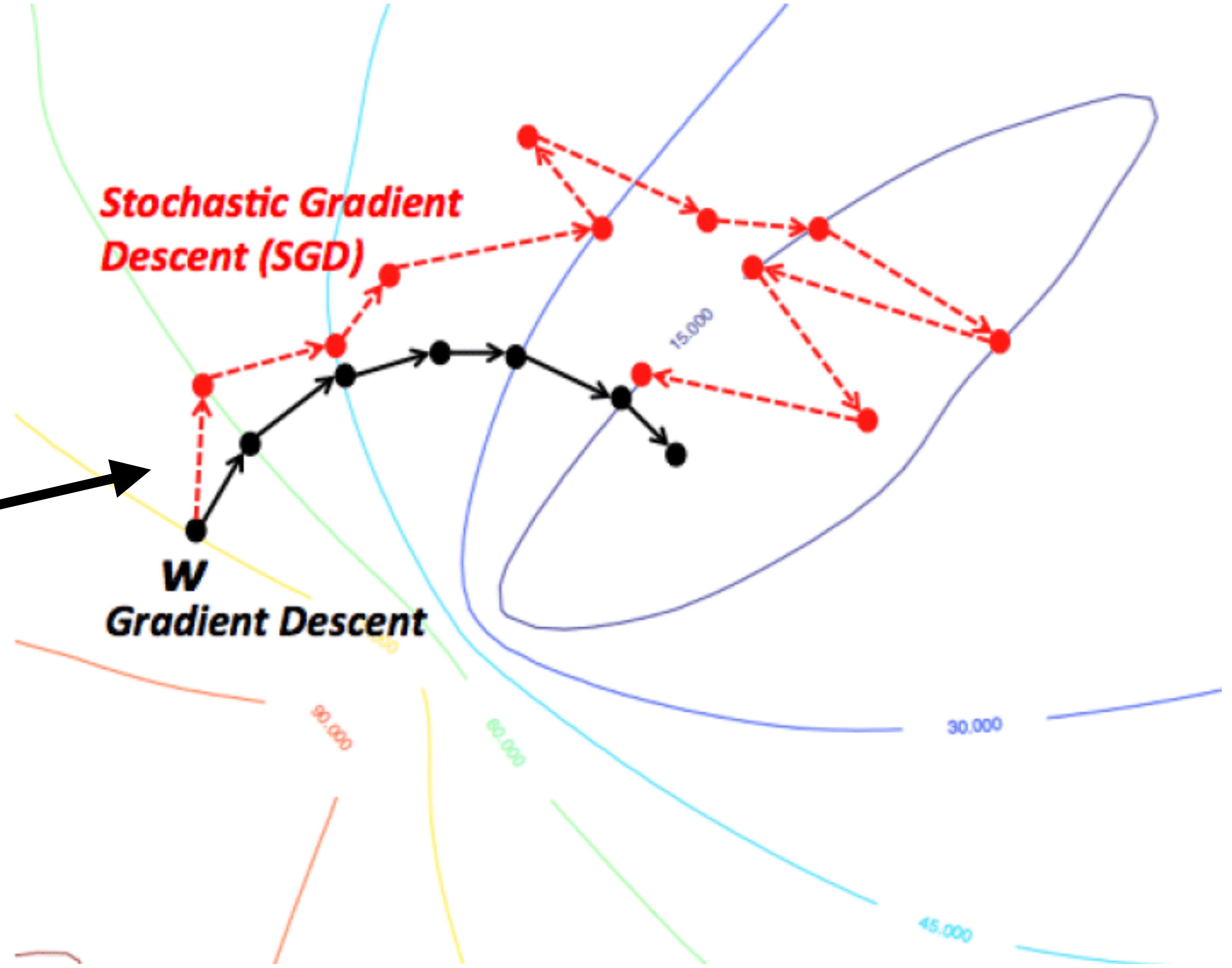
- How can the weights be updated more frequently, while still converging to the correct loss minimum?
- Consider a slight modification to improve scaling and see how the optimization behavior changes

Stochastic Gradient Descent Enables Faster Evaluations

- Instead of updating weights with $\nabla_{w_t} L(\{x_i, y_i\}; w_t)$, update them with $\nabla_{w_t} L_i(x_i, y_i; w_t)$
- This simplified loss function only depends on <u>one randomly chosen observation</u> (x_i, y_i)

$$w_{t+1} = w_t - \eta \nabla_{w_t} L_i(x_i, y_i; w_t)$$

$$L_{LS,i}(x_i, y_i; w_t) \equiv (\hat{y}_i(x_i; w_t) - y_i)^2$$



Each red update is much, much faster than each black update, and both paths <u>should</u> converge to roughly the same point

Image from: David Macedo PhD Thesis 2017

SGD Updates are too Sensitive to Individual Data

- So weight optimization trajectory becomes very noisy, especially after the beginning
- We now want to reduce the SGD noise without returning to GD scaling

- Replace $\nabla_{w_t} L_i(x_i, y_i; w_t)$ with $\nabla_{w_t} L(\{x_i, y_i\}_{MB}; w_t)$
- $\{x_i, y_i\}_{MB}$ is a random mini-batch of data points

Mini-Batch Stochastic Gradient Descent

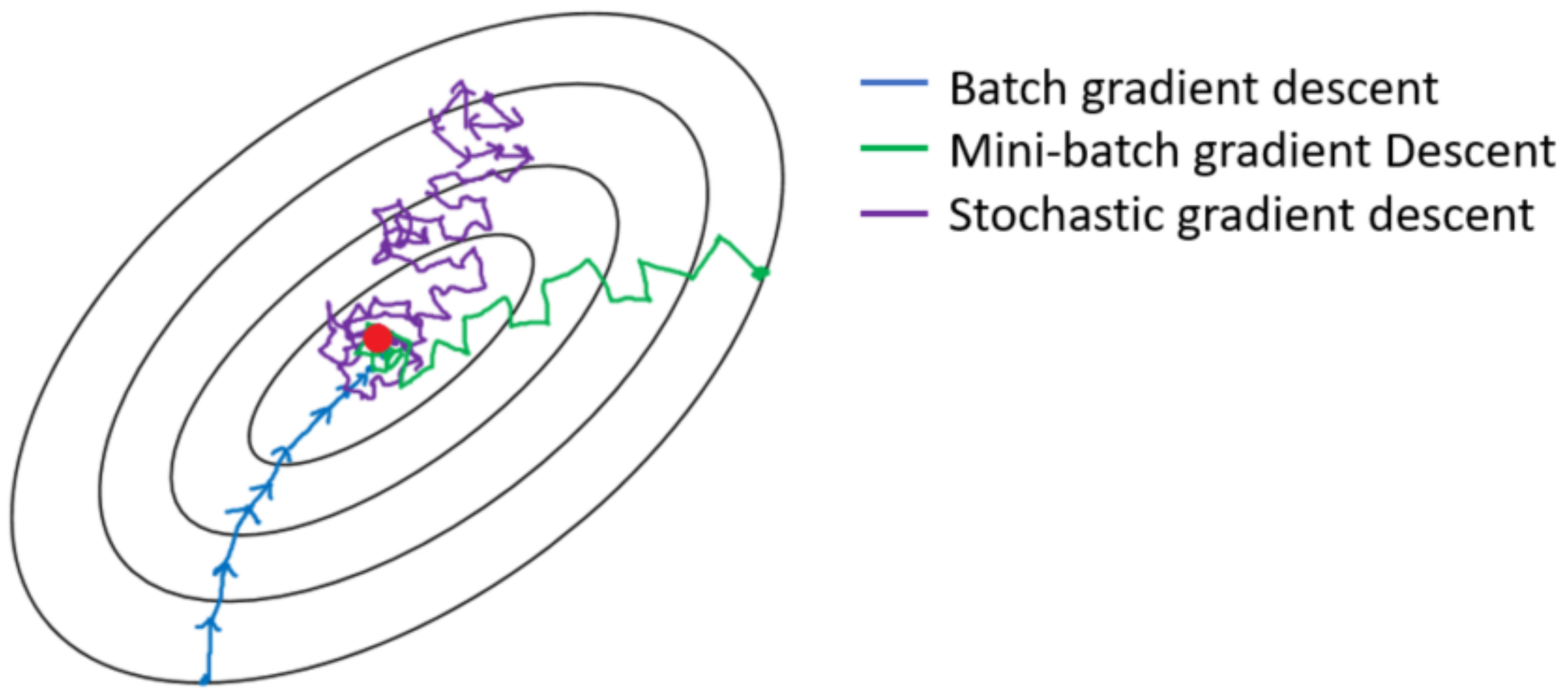


Image from: https://towardsdatascience.com/gradient-descent-algorithm-and-its-variants-10f652806a3

Break: Demonstrate Previous Methods in Live Demo

Using height-weight data from Kaggle:

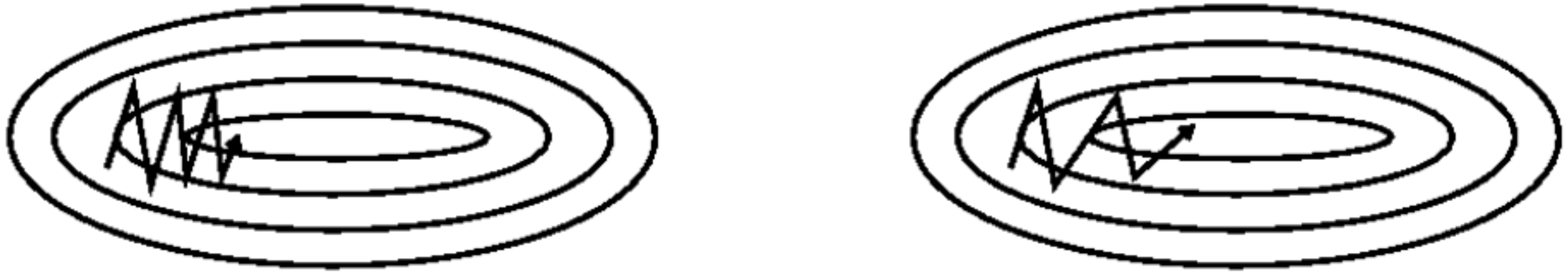
https://www.kaggle.com/mustafaali96/weight-height

Modifications Beyond Mini-Batch SGD

- Further modifications become problem-dependent, which requires knowing the data well
- Suppose the weight optimization still experiences difficulties:

  1. Convergence to an undesired local minimum - depends on loss function shape
  2. Noisy zig-zag updates - depends on redundancies or bias in the data

  - A common solution is introducing "momentum"



(a) SGD without momentum (b) SGD with momentum

S. Ruder, et al., arXiv:1609.04747 (2017).

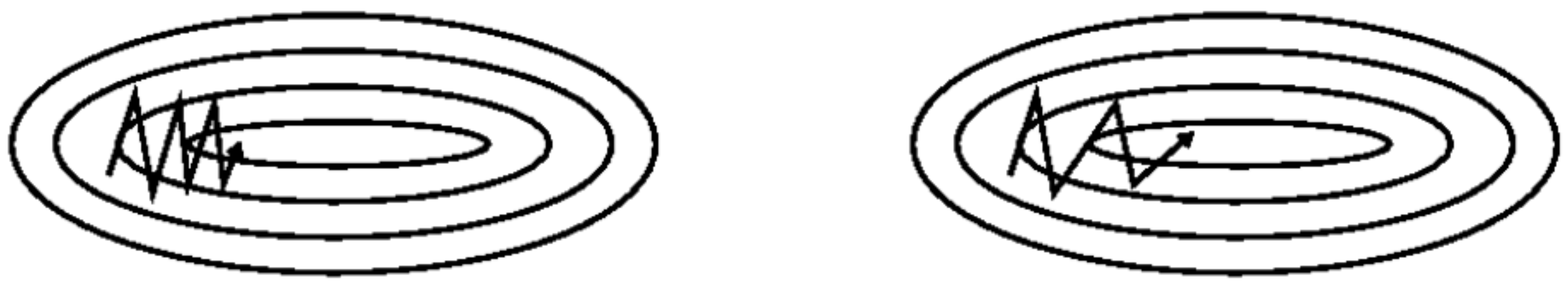
Mini-Batch SGD with Momentum

- Update not only the weights, but also a "weight velocity"
- Analogous to a velocity-dependent damping force in Newton's laws

$$w_{t+1} = w_t - v_t$$

$$v_t = \alpha v_{t-1} + \eta \nabla_{w_t} L(\{x_i, y_i\}_{MB}; w_t)$$

1. Speeds up only the updates heading towards the minimum
2. Pushes updates out of possible traps



(a) SGD without momentum (b) SGD with momentum

S. Ruder, et al., arXiv:1609.04747 (2017).

Visual Schematic of SGD with Momentum

$$v_t = \alpha v_{t-1} + \eta \nabla_{w_t} L(\{x_i, y_i\}_{MB}; w_t)$$

$$w_{t+1} = w_t - v_t$$

Image from: https://towardsdatascience.com/a-visual-explanation-of-gradient-descent-methods-momentum-adagrad-rmsprop-adam-f898b102325c

Visual Schematic of SGD with Momentum Cont.

$$v_t = \alpha v_{t-1} + \eta \nabla_{w_t} L(\{x_i, y_i\}_{MB}; w_t) \qquad w_{t+1} = w_t - v_t$$

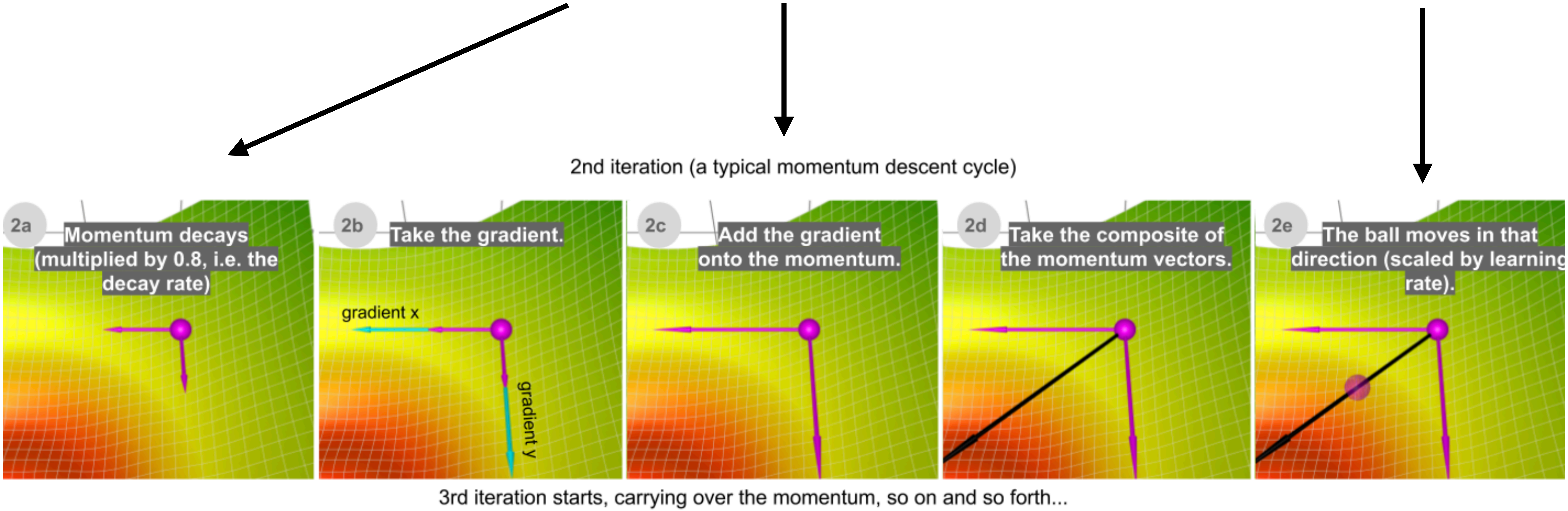
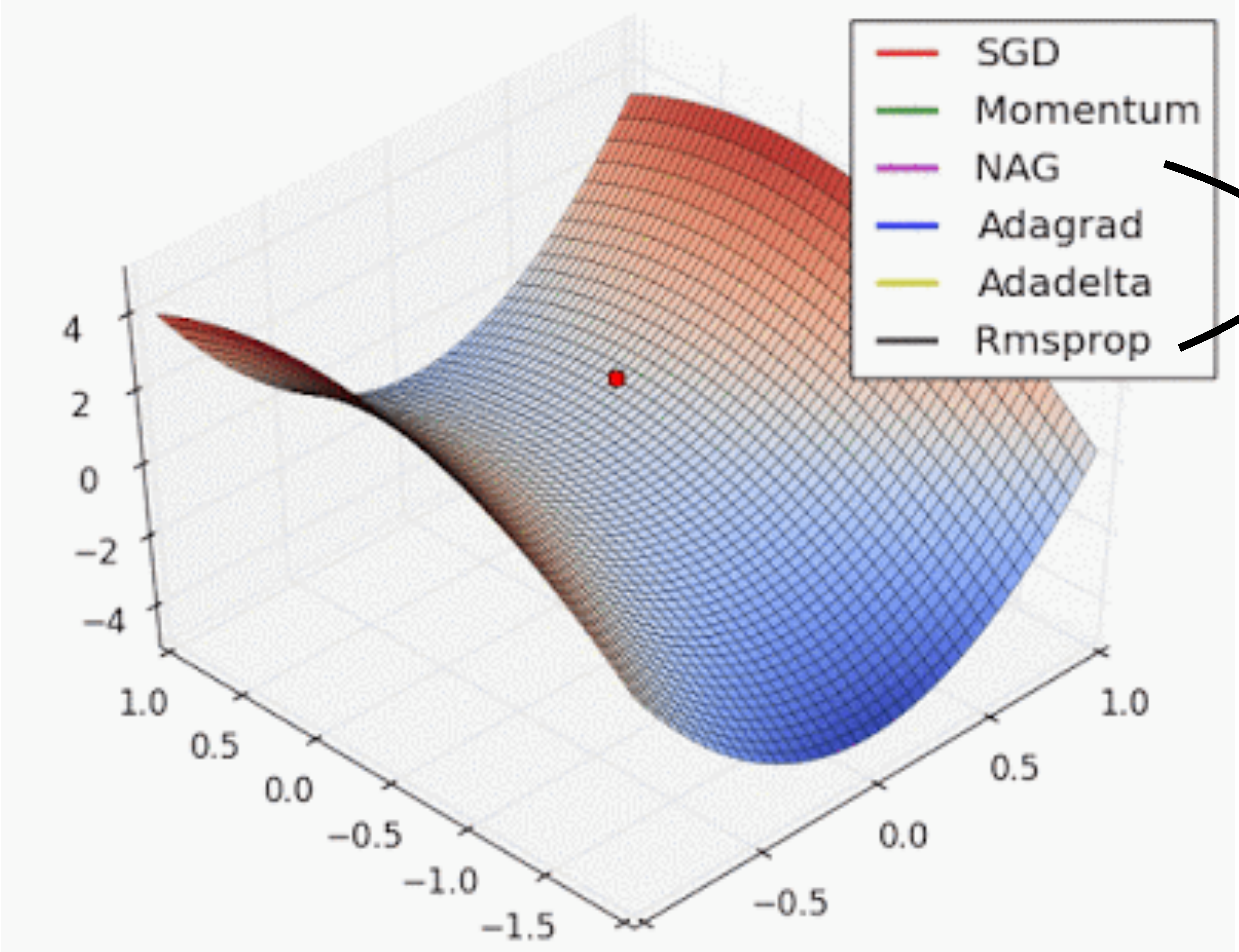


Image from: https://towardsdatascience.com/a-visual-explanation-of-gradient-descent-methods-momentum-adagrad-rmsprop-adam-f898b102325c

Sometimes SGD with Momentum Still Isn't Enough

- If the loss function has a deep valley, our previous methods don't work

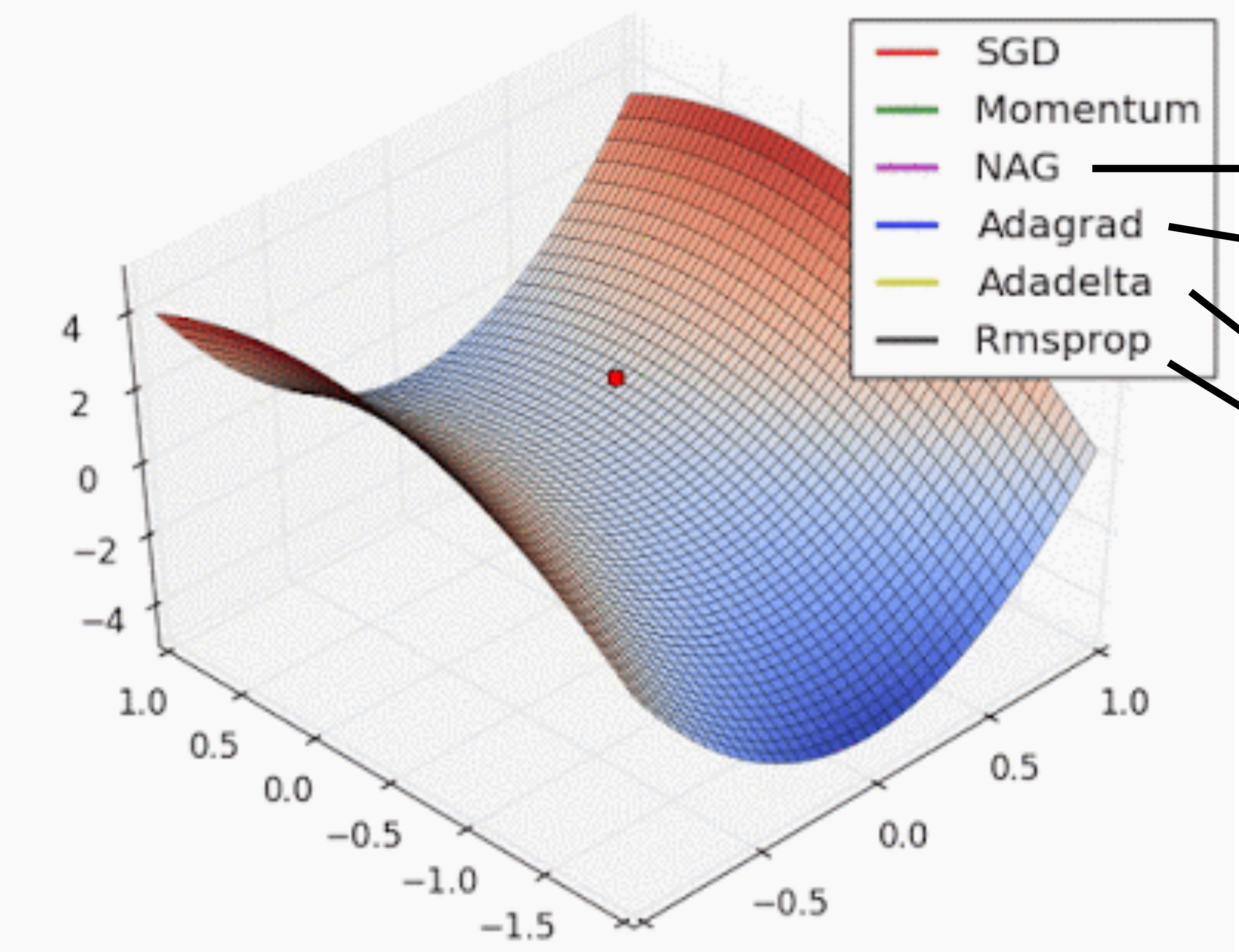
Example: logistic regression on the noisy moons dataset in scikit-learn



Let's see why these other methods help

Animation from: http://www.denizyuret.com/2015/03/alec-radfords-animations-for.html

Including Adaptive Step Sizes



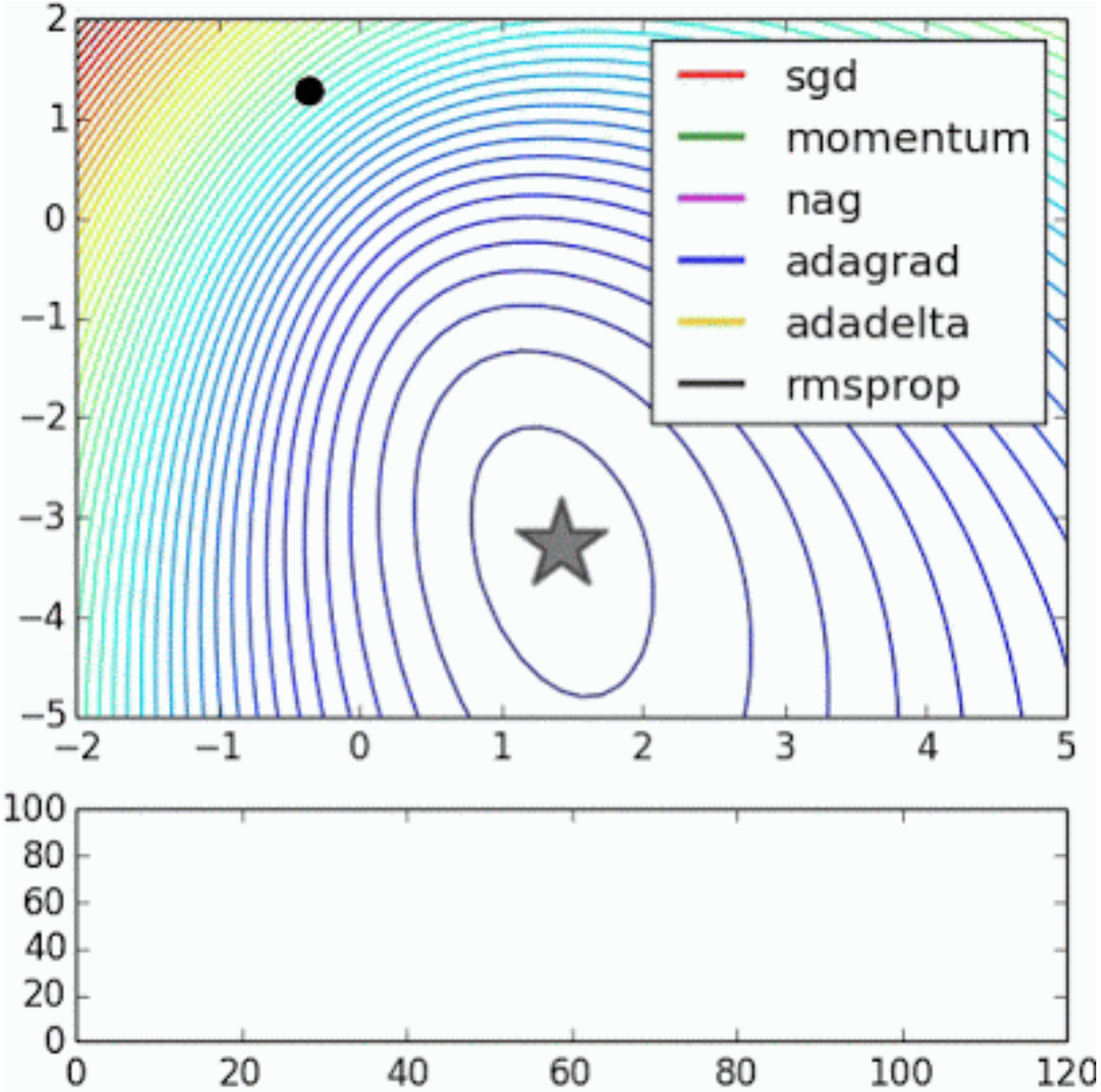
NAG: $L \rightarrow L(\{x_i, y_i\}_{MB}; w_t - \alpha v_{t-1})$

Adagrad: $$\eta_0 \rightarrow \frac{\eta_0}{\sqrt{\sum_{\tilde{t}=1}^{t} (\nabla_{w_{\tilde{t}}} L)^2 + \epsilon}}$$

Adadelta, Rmsprop: Replaces Adagrad sum with weighted average

Animation from: http://www.denizyuret.com/2015/03/alec-radfords-animations-for.html

Connecting Back to Model Accuracy



Animation from: http://www.denizyuret.com/2015/03/alec-radfords-animations-for.html

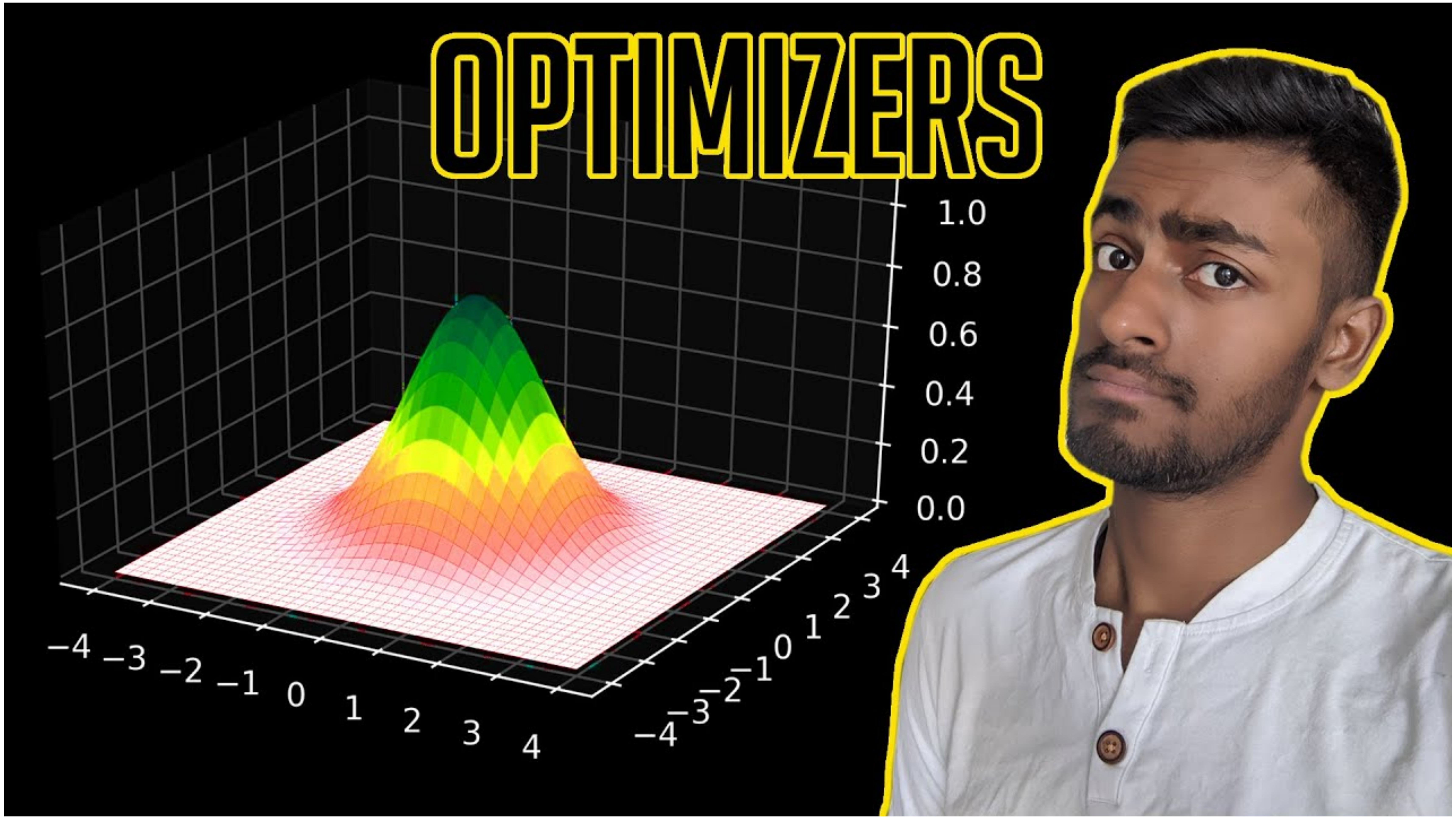
Conclusions: Zoo of Optimization Methods

Method	Advantage	Disadvantage
GD	Least noisy updates	Expensive gradient evaluations
SGD	Cheap gradient evaluations	Most noisy updates
Mini-Batch SGD	Cheap gradient evaluations	Moderately noisy updates
SGD + Momentum	More traversal in desired direction	Unreliable in non-convex terrains
Adagrad and Beyond	Self-corrected step sizes	Step sizes vanish after a while

- When confronted with a model fitting problem with lots of data, choose an optimization method by assessing:

1. Wether it can reliably converge to a loss minimum
2. How long it takes to converge
3. What path it takes through the loss function terrain

Short Overview Video on Optimizers



“Optimizers - EXPLAINED!” by CodeEmporium