We can do better

Momentum and ReLU

Two

~~Problems~~

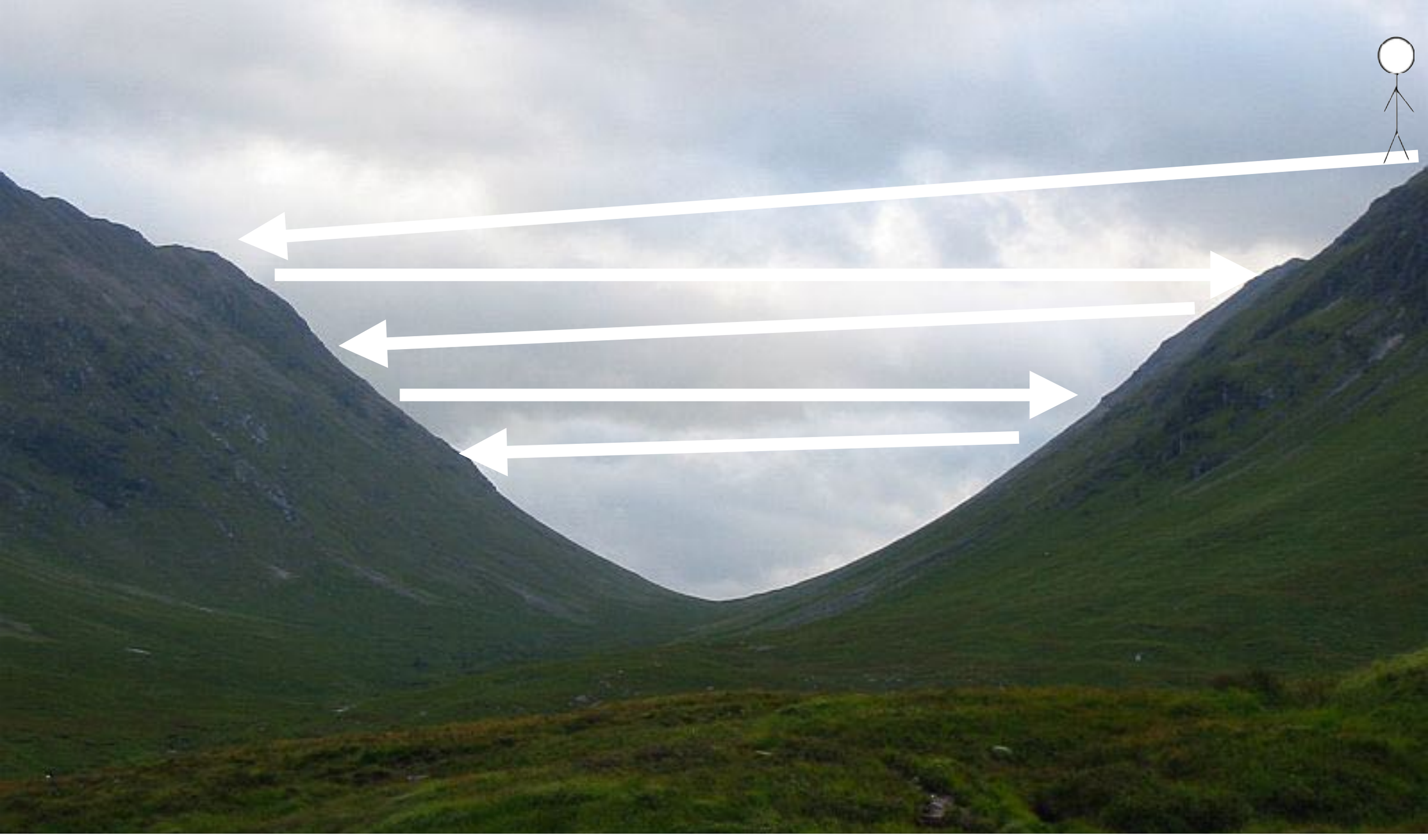
Improvements

The Valley Problem

The Valley Problem

The Valley Problem

The Valley Problem



Zig-Zag Sadness

Momentum-based Stochastic Gradient Descent (SGD)

Solution?

~~Solution?~~

Improvement

Throwback to Physics 101

$$F = -\nabla U$$

force felt by the particle is precisely the (negative) **gradient** of the cost function

we need to
gradually
slow down our
travel down the
cost mountain

let’s be clear

physical momentum

!=

SGD momentum

Previous SGD

$$w \rightarrow w' = w - \eta \nabla C$$

```
# Parameter update
w += -learning_rate * dw
```

SGD +
Momentum

$$v \rightarrow v' = \mu v - \eta \nabla C$$
$$w \rightarrow w' = w + v'$$

$$0 \leq \mu \leq 1$$

```
# Momentum update
v = mu * v - learning_rate * dx
x += v
```

🔑 Takeaways

- Gradient-based SGD improves the learning rate and helps the cost function settle at minima.
- Momentum may be a bit of a misnomer, but there is an applicable analogy that connects SGD momentum with Physical momentum: resistance to change.

We can improve on Momentum-based SGD

{Slightly}

Nesterov Momentum

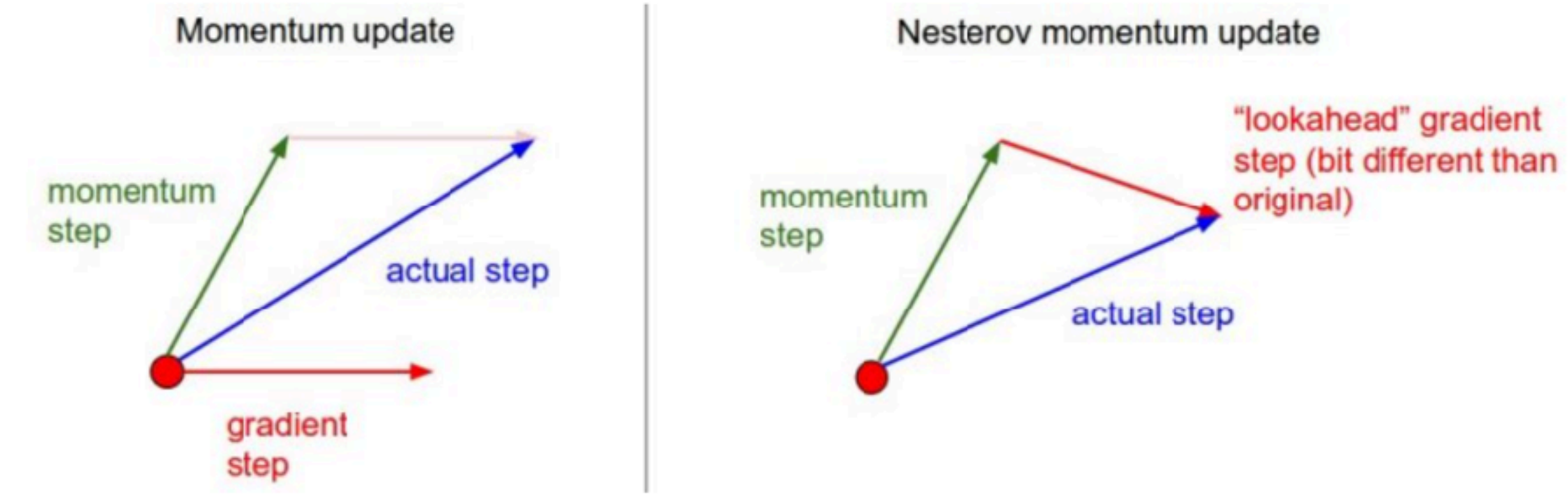
- We already plan on updating **w** by **μ*v** before we even need to calculate **η*∇C**

- Remember: **w’ = w + v’ = w + (μ*v - η*∇C)**

- We treat the future approximate position **x + μ * v** as a “lookahead”

- It makes sense to compute the gradient at **x + μ * v** instead of at the “old/stale” position x.

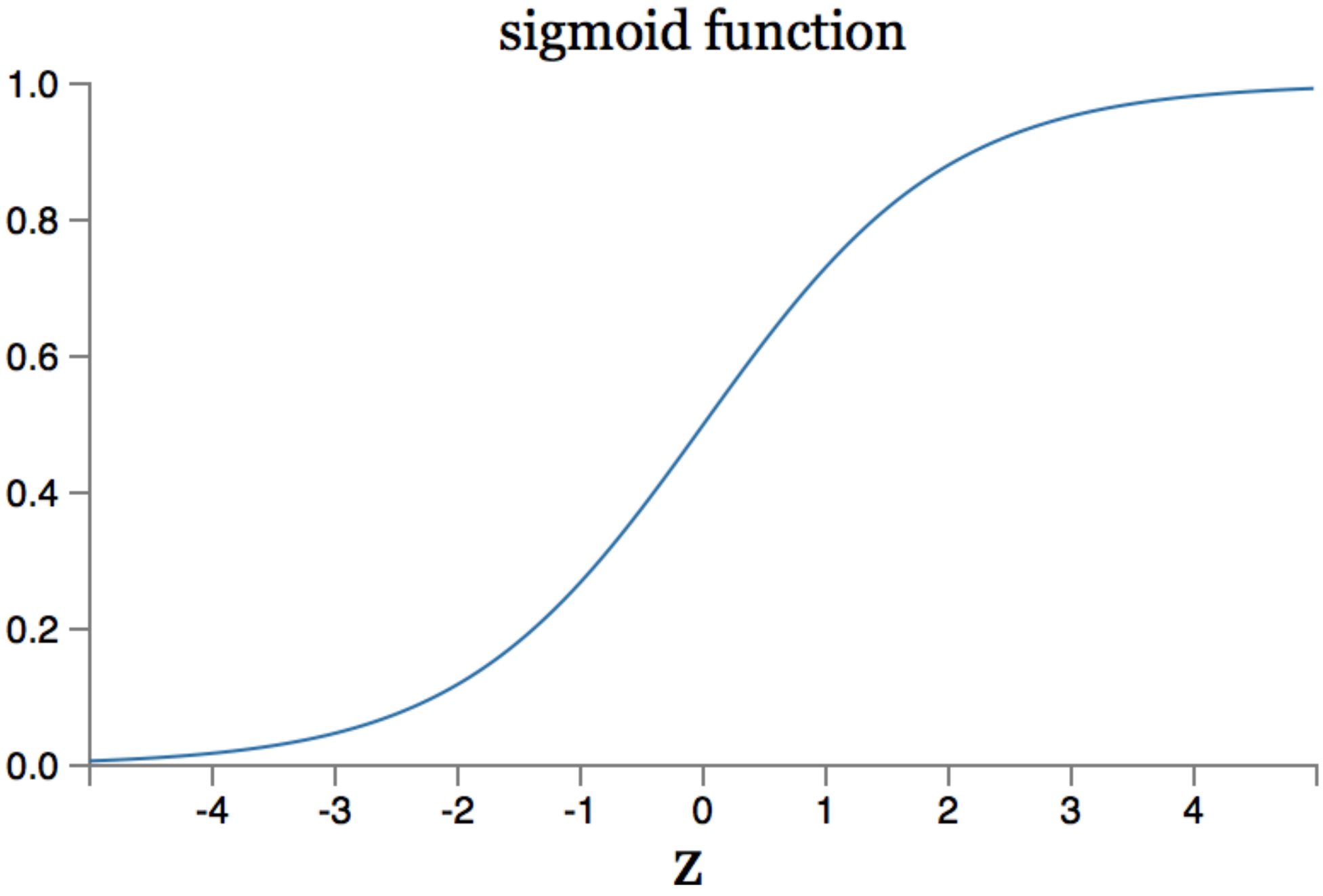
Nesterov Momentum



Nesterov momentum. Instead of evaluating gradient at the current position (red circle), we know that our momentum is about to carry us to the tip of the green arrow. With Nesterov momentum we therefore instead evaluate the gradient at this "looked-ahead" position.

Why σ? (Recap)

- It came first and is still in wide spread use
- It is related to softmax, which is used for the output layer for classification (linear output is for regression)
- Sigmoid itself is also sometimes used for the output layer
    - Especially for binary classification and feature detection
- The [0,1] range fits with probability interpretations
- It is a closer fit to neuron models than is the rectified linear unit (ReLU)
- It's fun to have a crazy looking derivative $\sigma' = \sigma(1 - \sigma)$



$$\sigma(z) \equiv \frac{1}{1+e^{-z}}.$$

Why not σ?

Sigmoids saturate and kill gradients

An improvement we have seen already?

We can change our cost function.

Cross-entropy cost to the rescue!

Good ‘ole

Quadratic Cost

$$\mathrm{C} = C_{MSE} = (^1/_{n_L}) \sum_{j=1}^{n_L} (y_j - o_j)^2$$

$$\frac{\partial C}{\partial z_{L,j}} = -2(y_k - a_{L,j}) \frac{\partial a_{L,j}}{\partial z_{L,j}} / n_L$$

$$\frac{\partial C}{\partial z_{L,j}} = -2(y_k - a_{L,j})(\sigma(z_{L,j})(1 - \sigma(z_{L,j}))/n_L$$

Cross-Entropy Cost

$$C = -\frac{1}{n} \sum_x [y \ln a + (1-y) \ln(1-a)] \,,$$

$$\frac{\partial C}{\partial w_j} = \frac{1}{n} \sum_x x_j(\sigma(z) - y).$$

A New Approach

We can try changing our activation function

Rectified Linear Unit (ReLU)

f(x) = max(0, x)

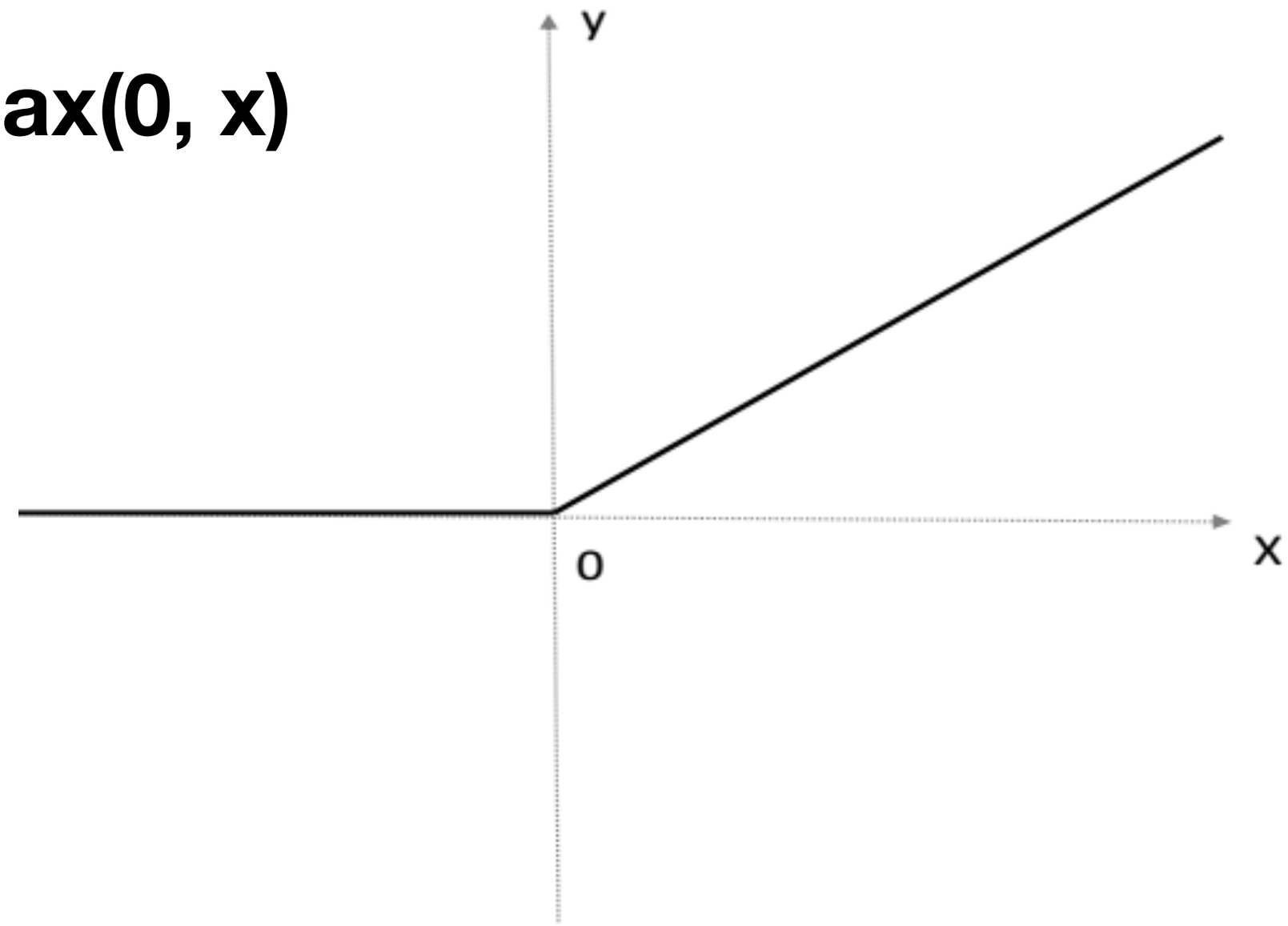


Figure 6.2: ReLU activation function