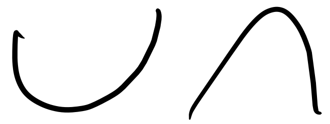
Gradient Descent

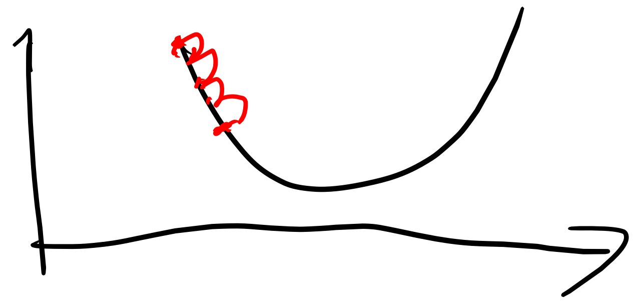
Normally: find derivative, set to zero, analyse inflection point



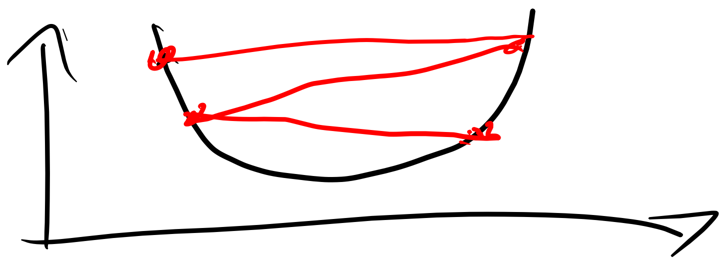
GD: iteratively descend in the direction where change is greatest.

GD doesn't terminate! We need a stop-condition.

Step-length is called "learning rate".



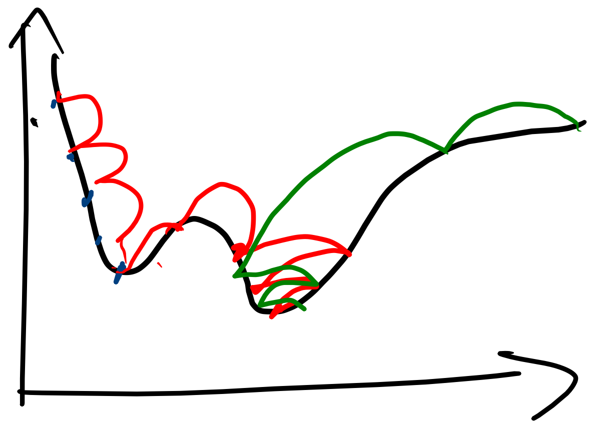
If we use fixed number epochs
a too small lr will miss the optimum.



A too large lr will "skip over" the minimum.

So . . . we want dynamic number of epochs and adjust lr as we go.

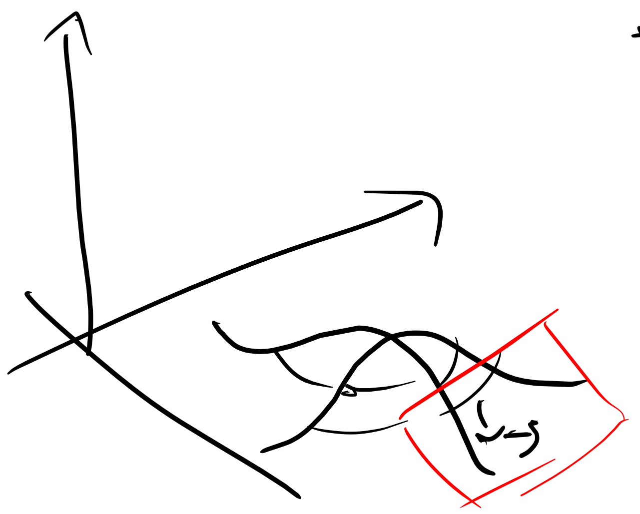
⇒ Early Stopping ⇒ keep track of loss metric
when below threshold: stop.



Adaptive Descent with Momentum (ADAM)

Adaptive learingrate: decreases with epoch

momentum: we add a physics-like condition to the cost function, that really acts like momentum (trögheit)



∇f - gradient always points in the direction of largest change

If we evaluate all points in the data:

Batch Gradient Descent

- expensive

+ good results

One step:

$$\theta_{n+1} = \theta_n - \eta \nabla_\theta MSE(\theta)$$

evaluated at each point for each parameter

Too expensive!

Stochastic Gradient Descent:

May or may not work: it's random!

- Choose a random point, derive at that point, take a small step.

Mini-batch Gradient Descent

Choose a random sample of points, evaluate, descend

By far the most common!