Gradient Descent

Boston University CS 506 - Lance Galletti

Gradient Descent (intuition)

Optimization method when there is no closed form solution to finding the extrema of a function.

Gradient Descent (intuition)

Optimization method when there is no closed form solution to finding the extrema of a function.

What to do if we don’t have an optimization method?

Gradient Descent (intuition)

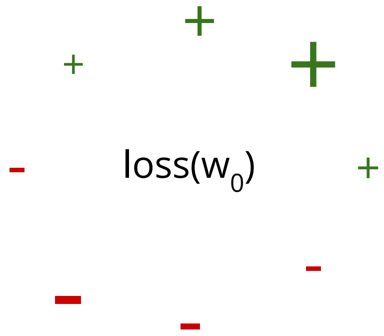
Optimization method when there is no closed form solution to finding the extrema of a function.

Example: Logistic Regression

Goal: find a sequence of w_i's (and b's) that converge toward **a** minimum.

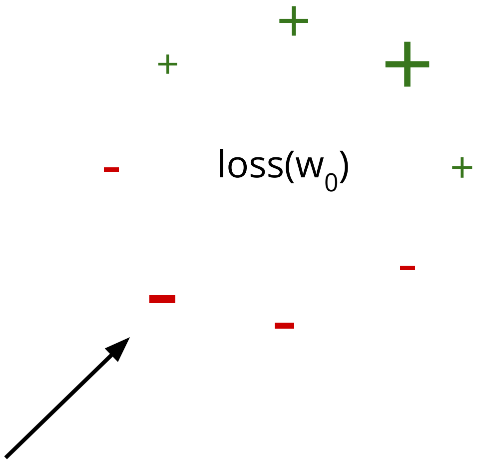
Gradient Descent (intuition)

Consider a random weight w_0. What happens to Loss(w_0) as you nudge w_0 slightly?



Gradient Descent (intuition)

Consider a random weight w_0. What happens to Loss(w_0) as you nudge w_0 slightly?



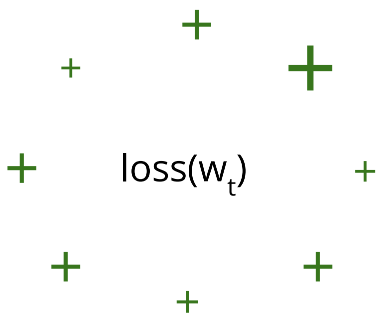
Clearly this is the best nudge to give w_0 to reduce our Loss

Gradient Descent (intuition)

As such we can define the following sequence:

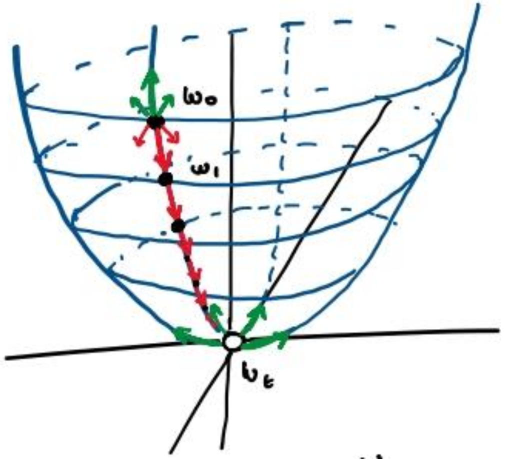
w_1 = best nudge to w_0
w_2 = best nudge to w_1
...

Until we reach w_t that looks like this:

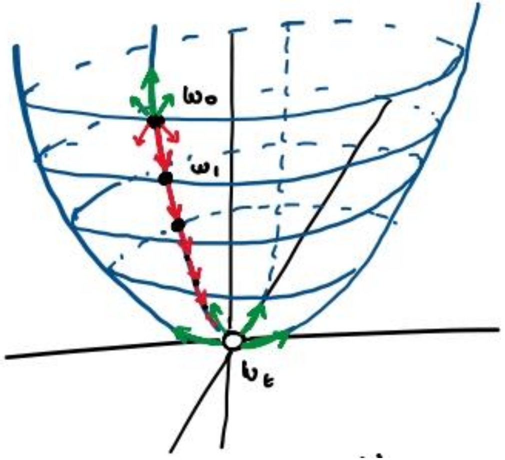


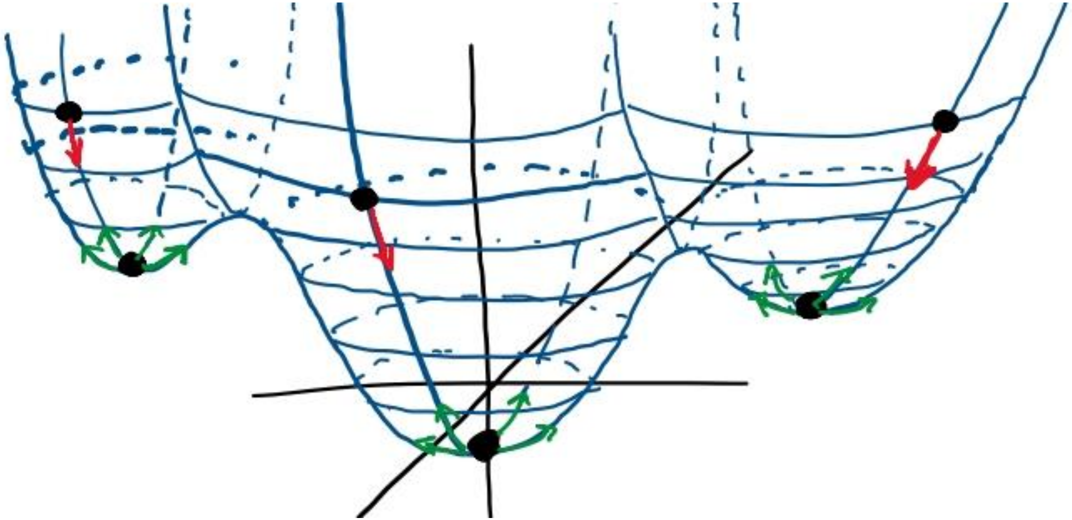
At this point we can stop updating w. Why?

Gradient Descent (intuition)



Gradient Descent (intuition)





Gradient Descent (intuition)

How can we know how much to nudge and in what direction?

Gradients

Intuitively the best nudge should be in the direction of the largest rate of change (steepness) of the function.

Gradients

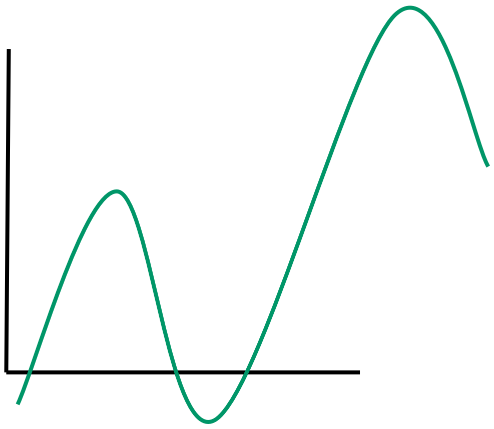
Intuitively the best nudge should be in the direction of the largest rate of change (steepness) of the function.

Rate of change -> think derivatives

Gradients

Intuitively the best nudge should be in the direction of the largest rate of change (steepness) of the function.

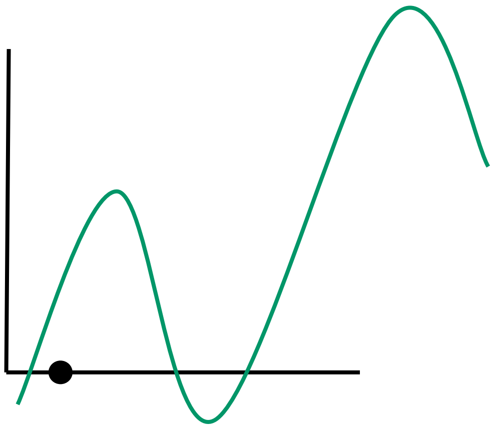
Rate of change -> think derivatives



Gradients

Intuitively the best nudge should be in the direction of the largest rate of change (steepness) of the function.

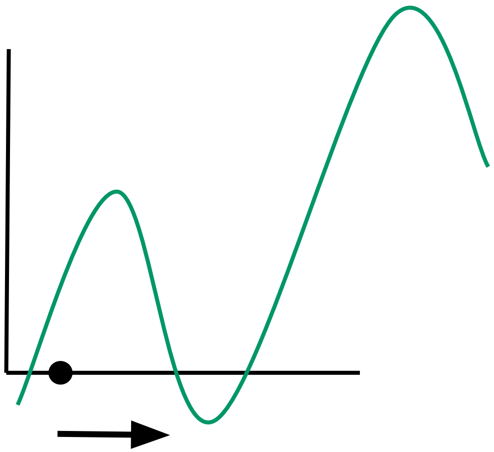
Rate of change -> think derivatives



Gradients

Intuitively the best nudge should be in the direction of the largest rate of change (steepness) of the function.

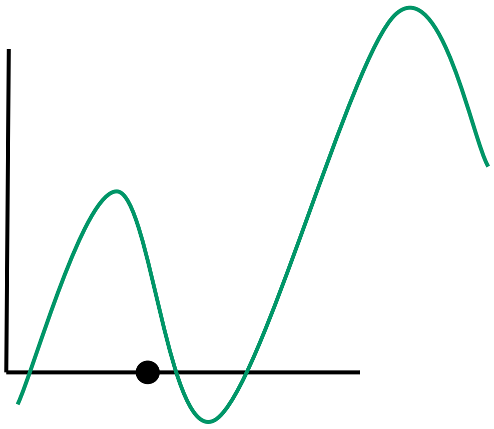
Rate of change -> think derivatives



Gradients

Intuitively the best nudge should be in the direction of the largest rate of change (steepness) of the function.

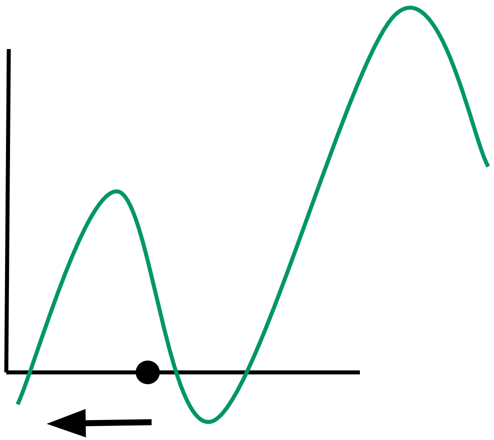
Rate of change -> think derivatives



Gradients

Intuitively the best nudge should be in the direction of the largest rate of change (steepness) of the function.

Rate of change -> think derivatives

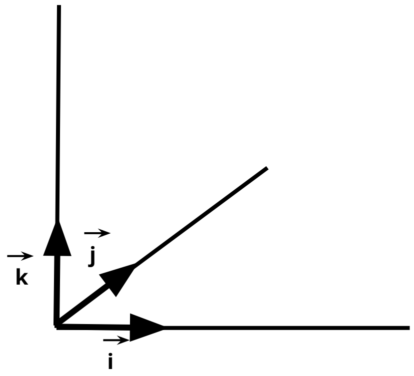


Gradients

Intuitively, the rate of change of a multi-dimensional function should be a combination of the rate change in each dimension.

Gradients

Intuitively, the rate of change of a multi-dimensional function should be a combination of the rate change in each dimension. For a 3-dimensional function, the rate of change would be:



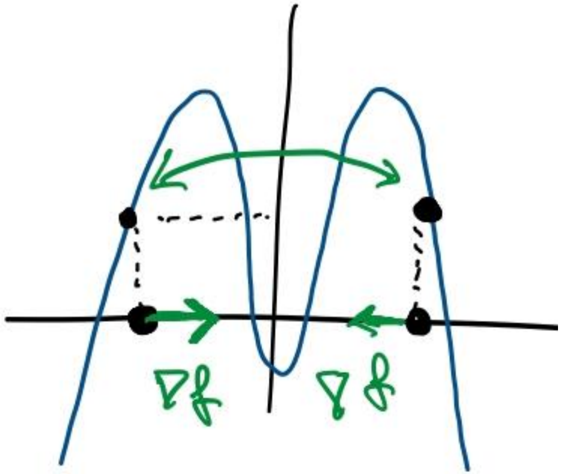
$$\nabla f(x,y,z) = \frac{\partial f}{\partial x}\vec{i} + \frac{\partial f}{\partial y}\vec{j} + \frac{\partial f}{\partial z}\vec{k}$$

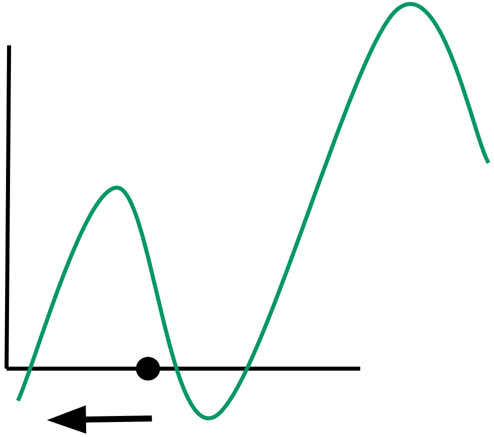
Example

Gradients

However, the gradient expresses the **instantaneous** rate of change. At p, ∇f_p is the steepest but the highest value of f will depend on how many units we step in that direction. If we step too many units away, the instantaneous change in f is no longer representative of what values f will take.

Example:





Gradient Descent

Given a “smooth” function f for which there exists no closed form solution for finding its **maximum**, we can find a local maximum through the following steps:

1. Define a step size α (tuning parameter)
2. Initialize p to be random
3. $p_{new} = \alpha \nabla f_p + p$
4. p □ p_{new}
5. Repeat 3 & 4 until $p \sim p_{new}$

To find a local **minimum**, just use $-\nabla f_p$

Gradient Descent

Notes about α:

- If α is too large, GD may overshoot the maximum, take a long time to or never be able to converge
- If α is too small, GD may take too long to converge

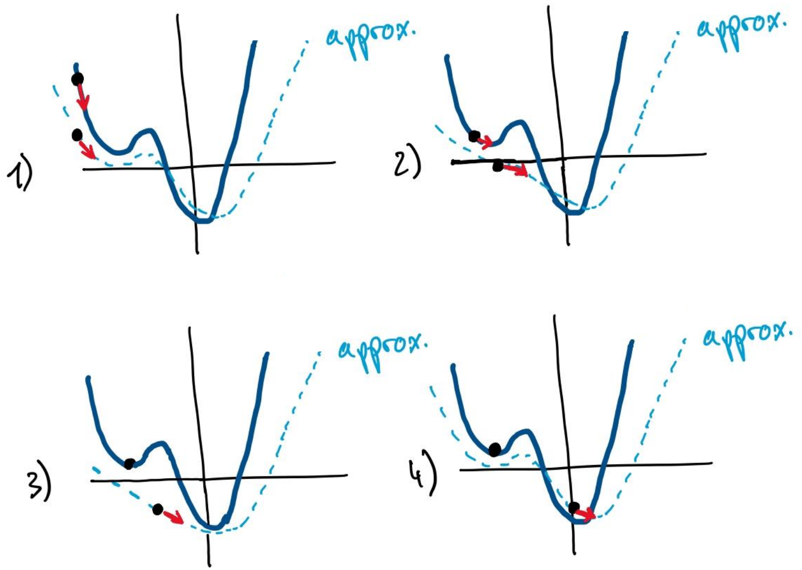
Stochastic Gradient Descent

Recall the Cost is computed for the entire dataset. This has some limitations:

1. It’s expensive to run
2. The result we get depends only on the initial starting point

Stochastic Gradient Descent

Goal: Approximate the gradient of the Cost using a sample of the data (batch)



Note

The magnitude of ∇f_p depends on p. A p gets closer to the min / max, the size of ∇f_p decreases.

This also means that points p that contain more “information” have larger gradients. So the order with which this process is exposed to examples matters.