Gradient Descent

① It's relatively straightforward to find the optimum of certain loss functions using standard calculus techniques, e.g.

$$L(\theta) = (20-5\theta)^2 + (41-12\theta)^2$$

$$\Rightarrow \frac{dL(\theta)}{d\theta} = 338\theta - 1184$$

$$\frac{dL(\theta)}{d\theta} = 0 \Rightarrow \theta = \frac{1184}{338} \approx 3.5$$

Thus:

$$\underset{\theta}{\operatorname{argmin}}\ L(\theta) \approx 3.5$$

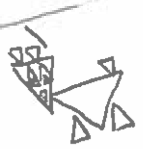
② But often (almost always in this course), the loss function is more complicated. What if it were

$$L(\theta) = (\sin 2\theta)(\log \theta^2)$$

Well, the derivative is doable:

Use the Product Rule! $\frac{d(u \cdot v)}{d\theta} = \frac{du}{d\theta} \cdot v + u \cdot \frac{dv}{d\theta}$

$$\frac{dL(\theta)}{d\theta} = 2\cos 2\theta \log \theta^2 + (\sin 2\theta) \cdot \frac{2\theta}{\theta^2}$$



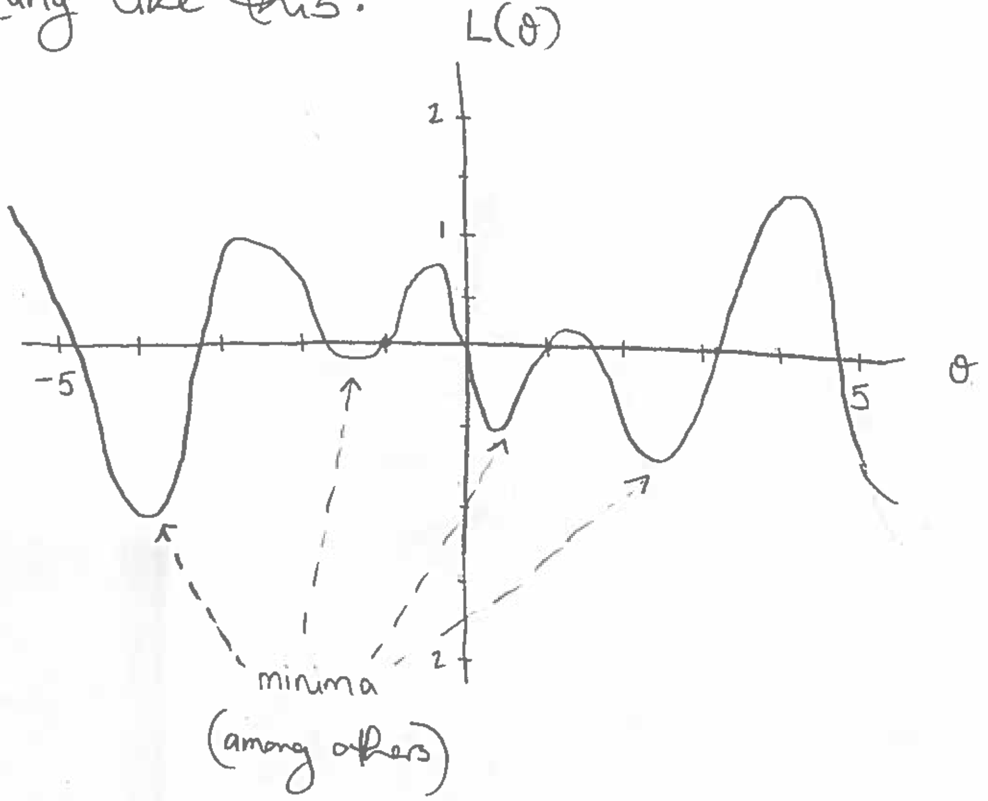
But then you have to set it to zero and solve for θ:

$$2\cos 2\theta \log \theta^2 + (\sin 2\theta) \cdot \frac{2\theta}{\theta^2} = 0$$

$$\Rightarrow \theta = ???$$

③ So, what to do?

If we were to graph the loss function, we'd get something like this:

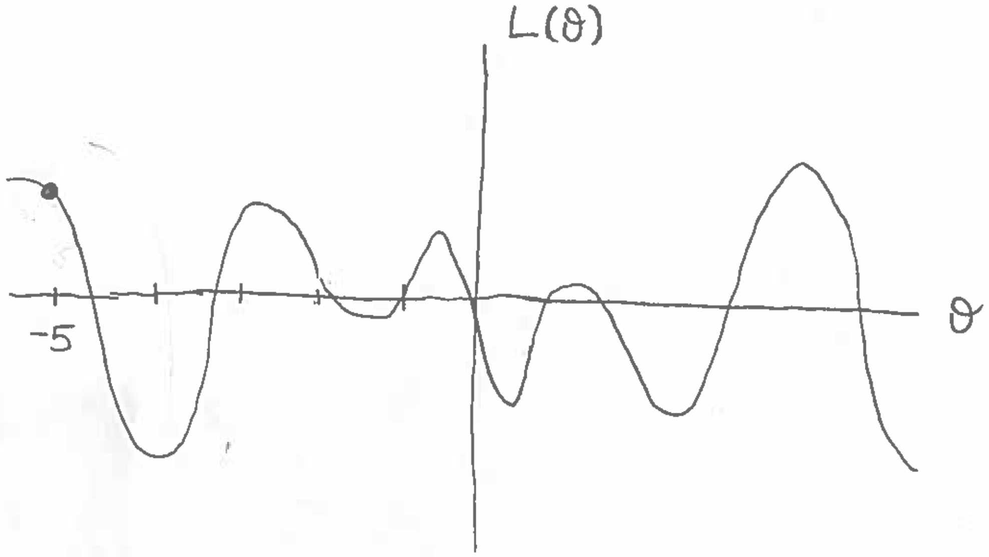


④ Now maybe we're ok with just finding a small loss, rather than insisting on the absolutely smallest loss. So $\theta = -4$ would be great, but $\theta = 2.5$ wouldn't be the end of the world.

GRADIENT DESCENT

5. The strategy we'll mostly use is a simple one called <u>gradient descent</u>.

We start by guessing (probably arbitrarily) some value for θ, like $\theta = -5$:



We know the derivative of $L(\theta)$, so we can compute

$$\frac{dL(-5)}{d\theta} = 2\cos(-10)\log(25) + \sin(-10)\cdot\frac{(-10)}{25}$$

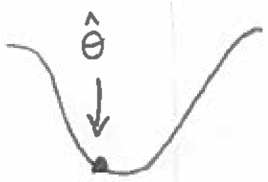
$$\approx -2.56$$

6. Since the derivative at $\theta = -5$ is <u>negative</u>, that means $L(\theta)$ is <u>decreasing</u> as we <u>increase</u> θ from -5. We're trying to minimize $L(\theta)$, so we want to <u>increase</u> θ.

7) But by how much? Gradient descent uses the following intuition:

> the steeper the function at our guess $\hat{\theta}$, the more we should increase (or decrease) our guess

It's easier to rationalize this by thinking about when the derivative has a very small magnitude:



Suppose that's our guess $\hat{\theta}$. The guess is very close to the optimum, so the derivative at $\hat{\theta}$ is pretty close to zero (e.g. say it's -0.001). The magnitude of the derivative warns us that we're getting close to the optimum, so we don't want to increase it by much.

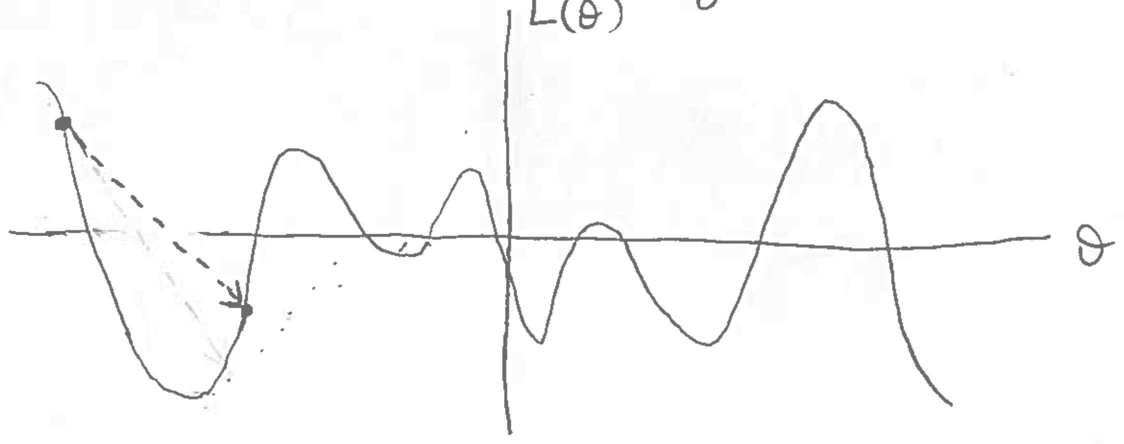
By contrast, a derivative of high magnitude gives us more freedom to jump ahead, heedless, into the abyss.

⑧ So suppose we increase θ by $-\alpha \cdot \frac{dL}{d\theta}(-5)$, where $\alpha = 0.5$.

This constant α, called the <u>learning rate</u>, is here being chosen arbitrarily, but let's see what happens.

$$\begin{aligned} \theta' &= \theta - \alpha \cdot \frac{dL}{d\theta}(\theta) \\ &= (-5) - 0.5 \cdot \frac{dL}{d\theta}(-5) \\ &\approx -3.72 \end{aligned}$$

⑨ Our new guess for θ is -3.72. We see we've descended further into the valley:



So the slope has become gentler:

$$\begin{aligned} \frac{dL}{da}(-3.72) &= 2\cos(2 \cdot -3.72)\log(-3.72^2) + \sin(2 \cdot -3.72) \cdot \frac{(-3.72)}{(-3.72)^2} \\ &\approx 1.16 \end{aligned}$$

(10) Since the derivative at $\theta = -3.72$ is <u>positive</u>, that means $L(\theta)$ is <u>increasing</u> as we <u>increase</u> θ from -3.72. We're trying to minimize $L(\theta)$, so we want to <u>decrease</u> θ.

In other words, we want to compute:

$$\theta' = \theta - \alpha \cdot \frac{dL}{d\theta}(\theta)$$

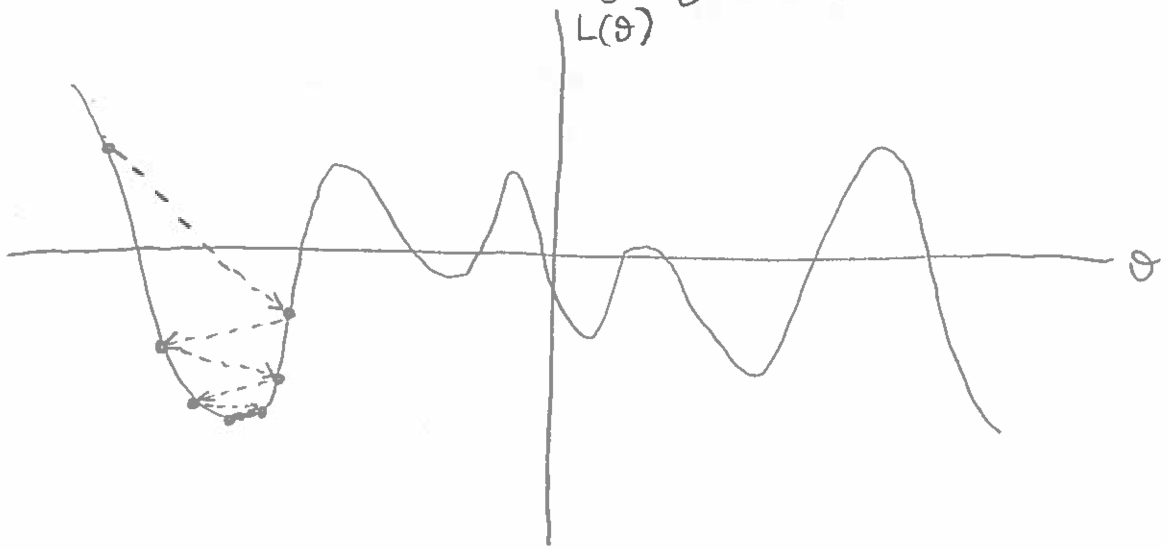
which is conveniently the exact same thing we did when the derivative was negative.

$$\theta' = (-3.72) - 0.5 \frac{dL}{d\theta}(-3.72)$$

$$= -3.72 - 0.58$$

$$= -4.3$$

(11) An algorithm has started to emerge. Keep doing this until you barely move at all (or you get tired):

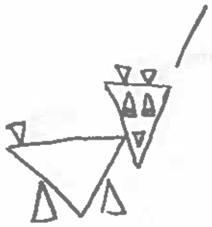


GRADIENT DESCENT

12 More rigourously:

```
GRADIENT DESCENT(loss function L, learning rate $\alpha \in \mathbb{R}$):
initialize $\theta^{(0)}$ to some real number; $t \leftarrow 0$
repeat until happy:
    - let update $\sigma^{(t)} \leftarrow -\alpha \cdot \frac{dL}{d\theta}(\theta^{(t)})$
    - let next guess $\theta^{(t+1)} \leftarrow \theta^{(t)} + \sigma^{(t)}$
    - let $t \leftarrow t+1$
```

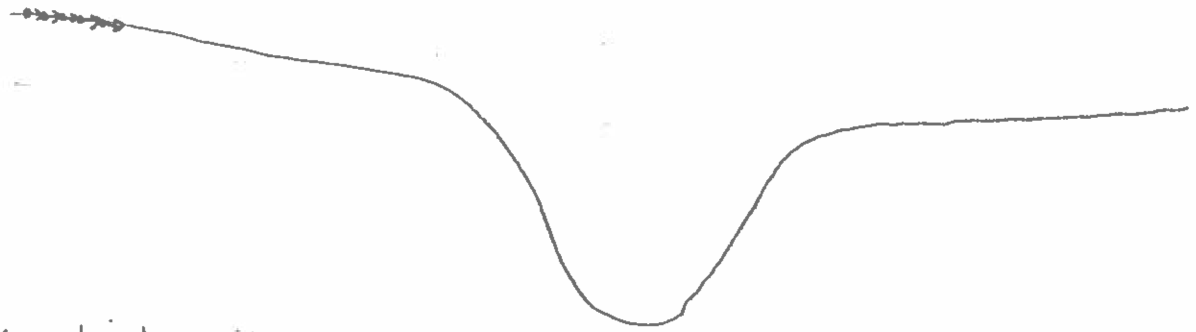
usually until $|\theta^{(t+1)} - \theta^{(t)}| < \epsilon$ for some small $\epsilon > 0$, or until $t > T$ for some large integer T



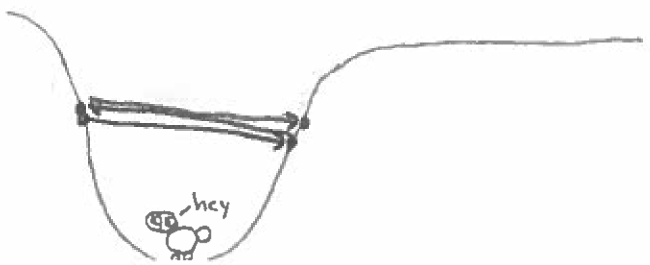
13 The main issue with gradient descent is:

how do you set the learning rate α?

If it's too low, it may take forever to reach a minimum:



If it's too high, it may keep jumping past the minimum:



(14) One strategy for dealing with this conundrum is to have the learning rate adapt, depending on what's happened so far during the gradient descent.

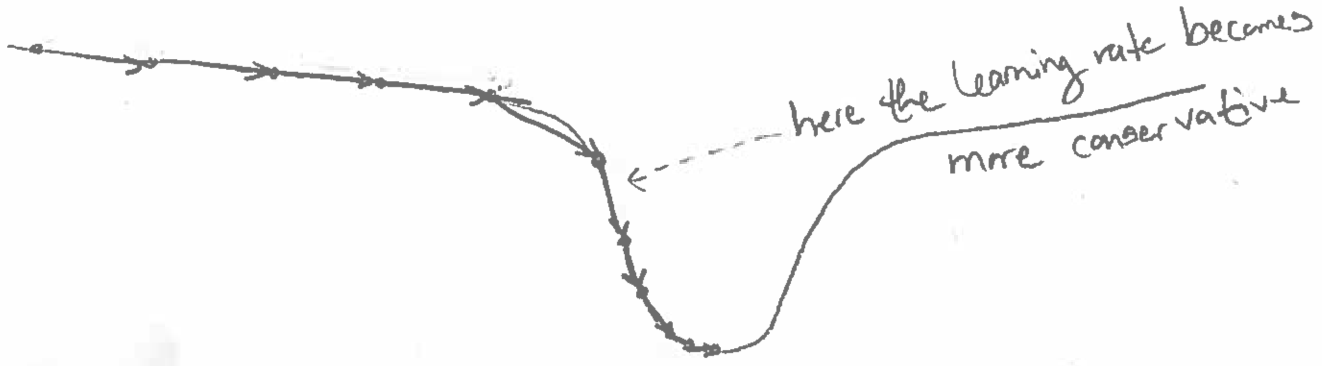
For instance, we could start with an aggressive (high) learning rate, and then gradually make it more conservative (lower it) as time goes on. Something like:

```
GDWITHTIMEBASEDDECAY(loss $L$, learning rate $\alpha$, decay rate $\beta$):
    initialize $\theta^{(0)} \in \mathbb{R}$; $t \leftarrow 0$; $\alpha_0 \leftarrow \alpha$
    repeat until happy:
        - let learning rate $\alpha^{(t)} \leftarrow \frac{\alpha}{1+\beta \cdot t}$
        - let update $\sigma^{(t)} \leftarrow -\alpha^{(t)} \cdot \frac{dL}{d\theta}(\theta^{(t)})$
        - let next guess $\theta^{(t+1)} \leftarrow \theta^{(t)} + \sigma^{(t)}$
        - let $t \leftarrow t+1$
```

Thus, the algorithm first jumps around, trying to find a valley, and then becomes more conservative, in order to stay in that valley and converge to its minimum.

But this is still fairly arbitrary, and doesn't take into account the behavior of the algorithm when determining the step size.

15) Maybe it would be useful to try the following. Start out aggressively, and only become more conservative once you begin to see significant vertical progress, e.g.



One way to do this is to set the rate $\alpha^{(t)}$ at time t to be inversely proportional to the magnitudes of the previous derivatives, e.g.

$$\alpha^{(t)} \leftarrow \frac{\alpha}{\delta + \sqrt{\sum_{t'=0}^{t} \left(\frac{dL}{d\theta}(\theta^{(t')})\right)^2}}$$

a tiny constant to avoid division by zero

as we experience large derivatives (either positive or negative), this gets increasingly large, which decreases the learning rate.

16 This update gives us the AdaGrad (Adaptive Gradient) algorithm:

```
ADAGRAD (loss L, init learning rate $\alpha$, tiny delta $\delta>0$):
initialize $\theta^{(0)} \in \mathbb{R}$; $t \leftarrow 0$
repeat until happy:
    - let learning rate $\alpha^{(t)} \leftarrow \dfrac{\alpha}{\delta + \sqrt{\sum_{t'=0}^{t} \left(\frac{dL}{d\theta}(\theta^{(t')})\right)^2}}$
    - let update $\sigma^{(t)} \leftarrow -\alpha^{(t)} \cdot \frac{dL}{d\theta}(\theta^{(t)})$
    - let next guess $\theta^{(t+1)} \leftarrow \theta^{(t)} + \sigma^{(t)}$
    - let $t \leftarrow t+1$
```

17 One potential downside to setting the learning rate this way:

$$\alpha^{(t)} \leftarrow \frac{\alpha}{\delta + \sqrt{\sum_{t'=0}^{t} \left(\frac{dL}{d\theta}(\theta^{(t')})\right)^2}}$$

← this is a constant

← this grows without bound

So eventually AdaGrad will crawl to a stop, possibly before you want it to.