#nodes in hidden layers
#hidden layers
activation fns
} examples of "hyperparameters"

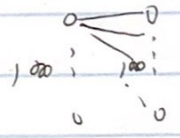
↓

not trainable parameters θ.

- NN basically repeated matrix multiplications
  + vectorized activations
  → perfect for GPU acceleration!!

- Count parameters

  Let's say $d = 100$ θ, $L = 10$, $n_h = 1000$



$\dim W^{(n)} \sim (10^3)^2 \sim 10^6$

θ $\times 10$ hidden layers

$\sim 10^7$ parameters

→ # of NN parameters can easily get very large!

↳ can still be fit to data using powerful algorithms & hardware

→ 1st demo nb: constructing a simple MLP in Keras & in pytorch

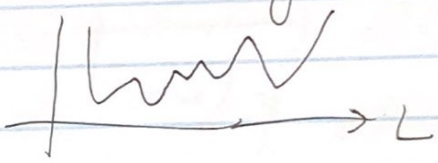
Training NNs

How to fit $f(x;\theta)$ w/ many (eg 1M) parameters?

Want to minimize $L(\theta)$ wrt θ

$$\left.\frac{\partial L}{\partial \theta}\right|_{\theta_*} = 0 \quad + \text{(ideally) global minimum}$$

L might have many local minima, or saddles



↓ don't want to get stuck.

Many approaches...

— one idea: Newton method

$\theta = \theta_0$ initial guess

$$L(\theta) = L(\theta_0) + \delta\theta_i \left.\frac{\partial L}{\partial \theta_i}\right|_{\theta_0} + \frac{1}{2}\delta\theta_i \delta\theta_j \left.\frac{\partial^2 L}{\partial\theta_i \partial\theta_j}\right|_{\theta_0} + \cdots$$

- Minimize 2nd order approx $\rightarrow \delta\theta$
- Repeat for $\theta + \delta\theta$
- Hopefully converge to ~~true~~ correct θ_*.

— Why is this infeasible!?

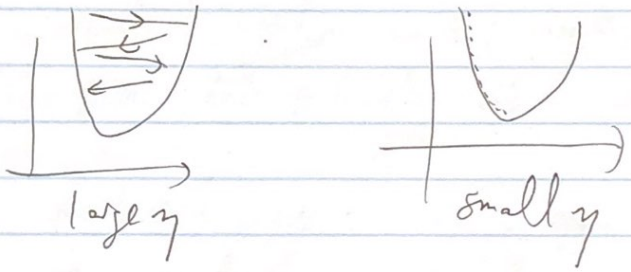
$\rightarrow \frac{\partial^2 L}{\partial\theta_i \partial\theta_j}$ too high dim'l & have to invert it!

- Instead, NNs are optimized w/ 1st order methods:

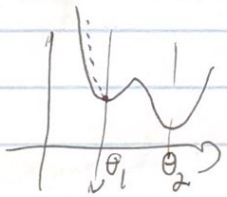
$$\delta\theta = -\eta \left.\frac{\partial L}{\partial \theta}\right|_{\theta_0}$$ "gradient descent"

idea: move in direction where L decreases

η = "learning rate" — important hyperparameter!
often need to tune for best perf.



- Problem: GD can get stuck in local minima



$\left.\frac{\partial L}{\partial \theta}\right|_{\theta_1} = 0 \Rightarrow \delta\theta = 0$ no update
get stuck at $\theta = \theta_1$

→ "stochastic gradient descent"
noisy gradients are better than true gradients!
how to get noisy gradients?