Scaling Rules for Optimization

Jeremy Bernstein
jbernstein@mit.edub

6.7960 :: Lecture 7

Plan for today

introduce the optimization problem

cover some classical approaches

- Newton
- Gauss-Newton
- steepest descent

heuristic picture of scaling

- width
- depth

modularizing the theory

The machine learning puzzle

Three pieces to the puzzle:

1. Approximation — Does there exist a neural net in my model family that fits the training data?

2. Optimization — If it does exist, can I find it?

3. Generalization — Does it work well on unseen data?

This lecture will focus mainly on the second question.

The optimization problem: formal statement

neural net $\quad f(x, w)$

prediction

target

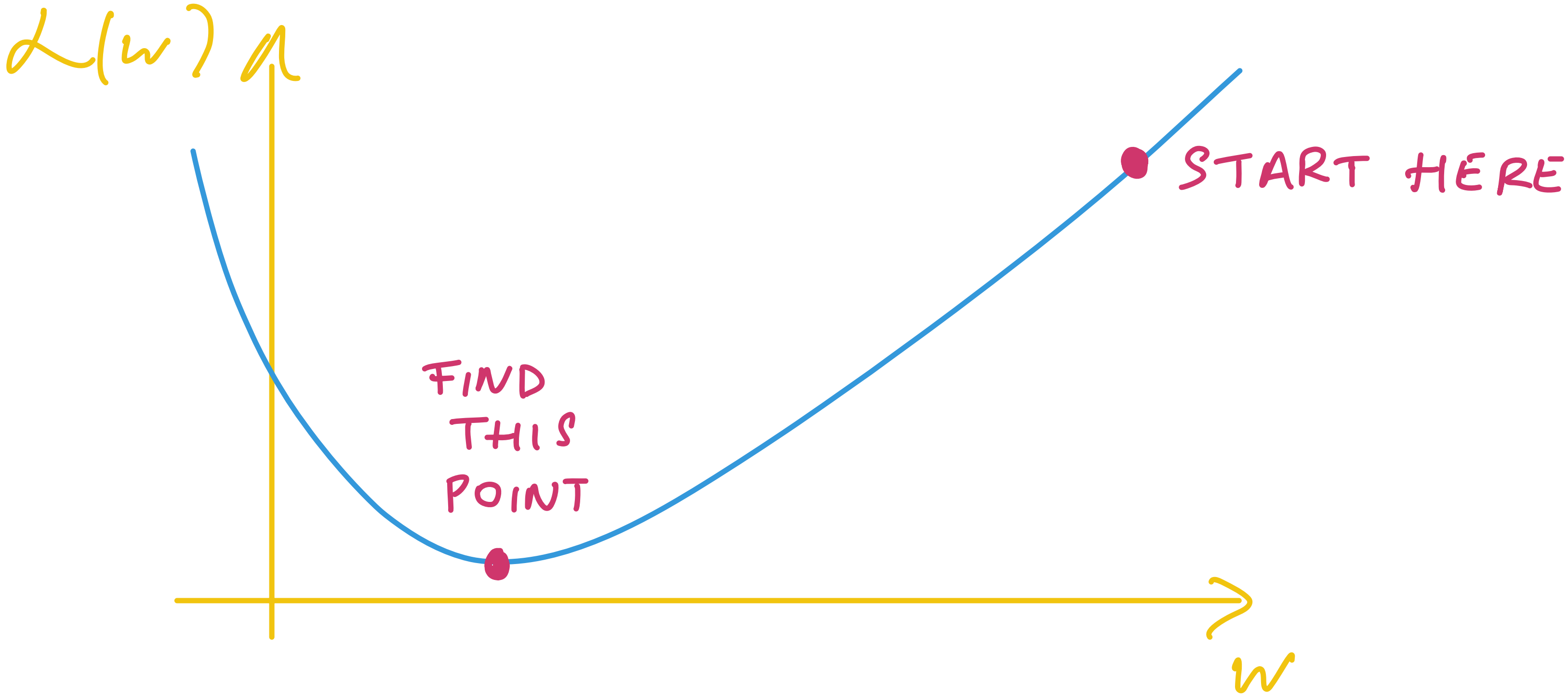
error measure $\quad \ell(\hat{y}, y)$

training data $\quad (x^{(1)}, y^{(1)}), \ldots, (x^{(N)}, y^{(N)})$

loss function $\quad \mathcal{L}(w) = \frac{1}{N} \sum_{i=1}^{N} \ell(f(x^{(i)}, w), y^{(i)})$

<u>GOAL</u> $\quad$ find w that minimizes $\mathcal{L}(w)$

The optimization problem: a picture



roughly, we just iterate

$$W \longrightarrow W - \underbrace{\eta}_{\text{learning rate}} \underbrace{\frac{\partial \mathcal{L}}{\partial W}}_{\text{gradient}}$$

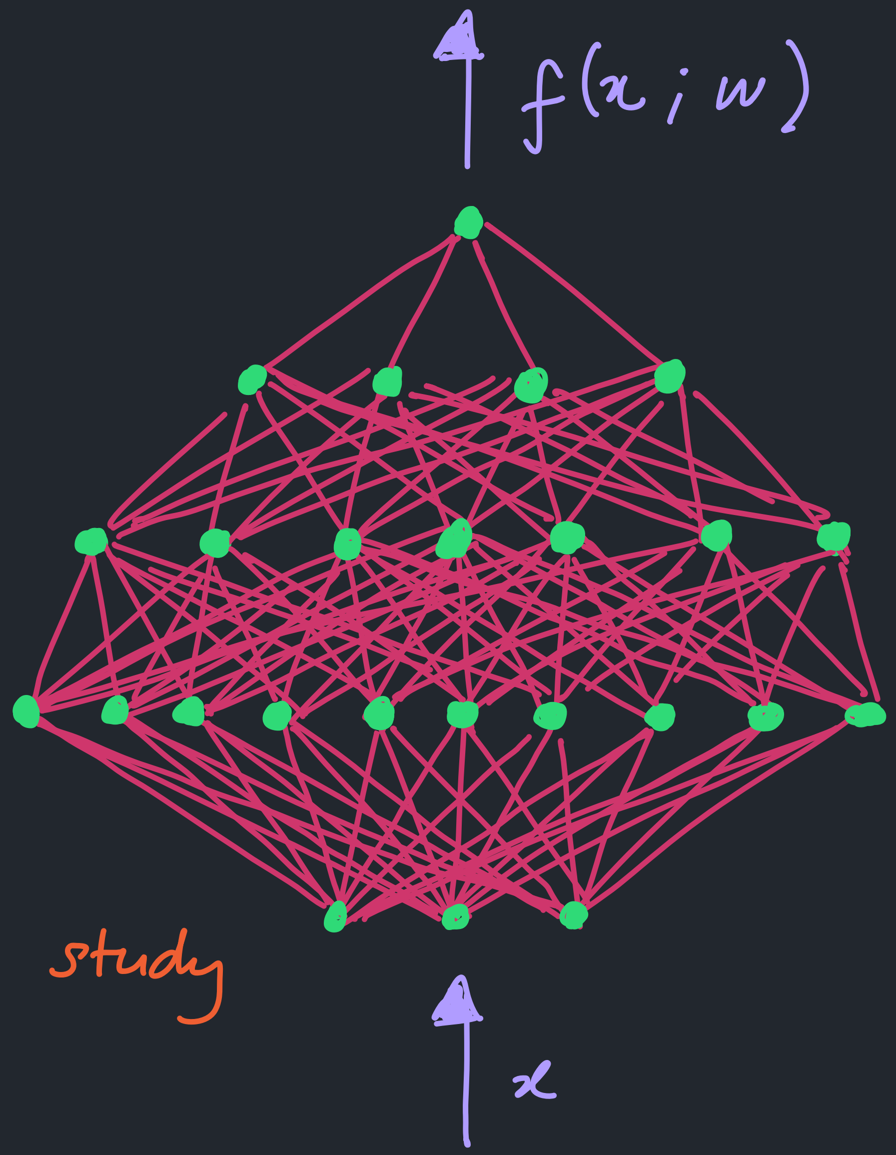
What makes optimization hard?

Some examples:

1. size: a lot of weights
2. depth: a lot of layers
3. noise: a lot of data

$\Downarrow$

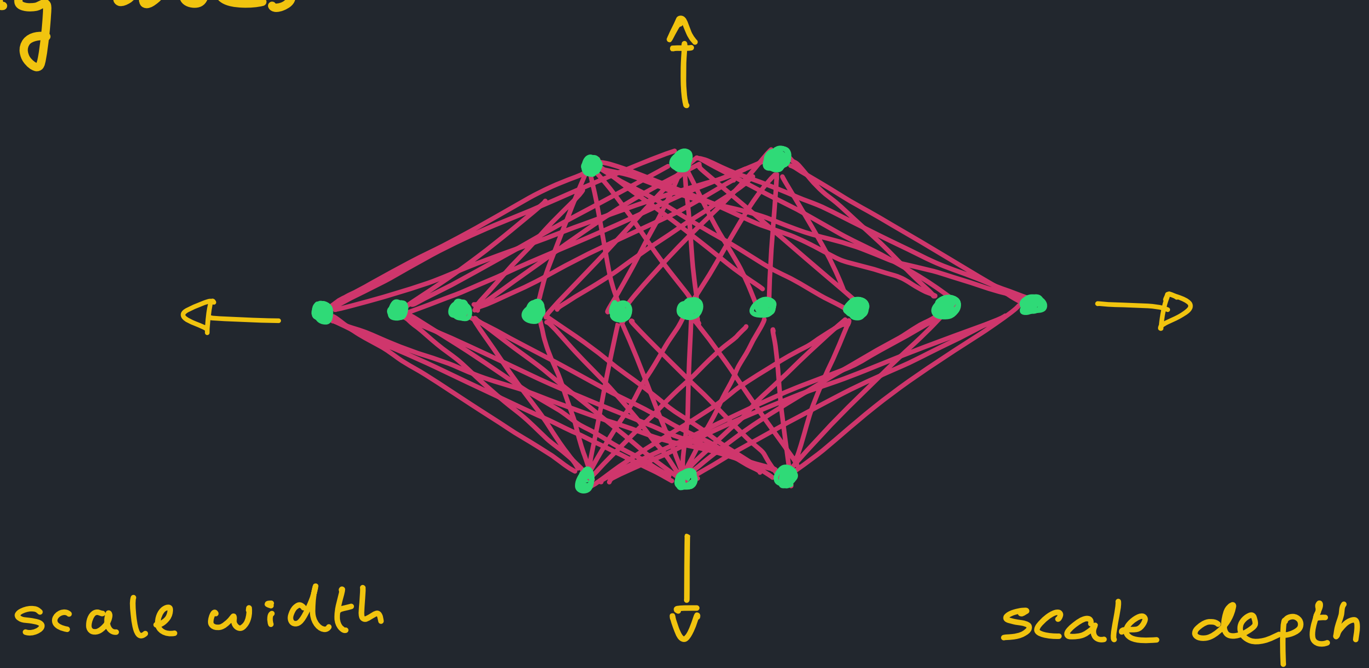
must use mini-batches

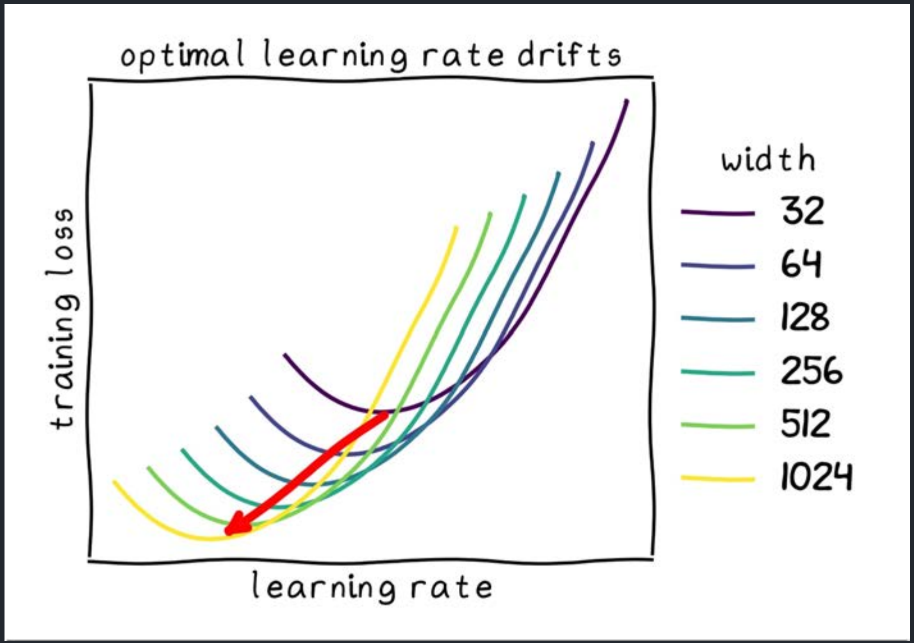


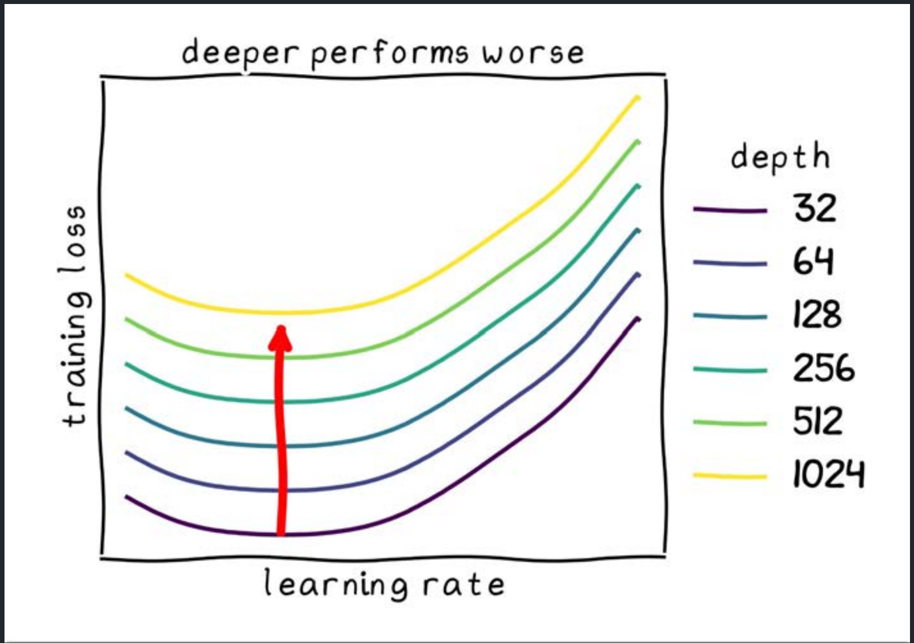
In this lecture, we will just study full-batch optimization...

... it's already interesting.

Scaling woes







Classical 1st and 2nd Order Methods

Let's survey some optimization methods

We will look at:

- first-order methods —— use 1^{st} derivatives

  gradient $g = \frac{\partial \mathcal{L}}{\partial w}$

- second-order methods —— use 2^{nd} derivatives

  Hessian $H = \frac{\partial^2 \mathcal{L}}{\partial w^2}$

Will try to highlight modelling assumptions and potential downsides of the different methods

Taylor expanding the loss

Different classical approaches to optimization take this Taylor expansion as a starting point

$$\mathcal{L}(w+\Delta w) = \underbrace{\mathcal{L}(w) + \frac{\partial \mathcal{L}}{\partial w}^T \Delta w}_{\text{"linearization"}} + \underbrace{\frac{1}{2} \Delta w \frac{\partial^2 \mathcal{L}}{\partial w^2} \Delta w}_{\text{"non-linear part"}} + \ldots$$

$$= \mathcal{L}(w) + g^T \Delta w + \frac{1}{2} \Delta w H \Delta w + \ldots$$

$g = \frac{\partial \mathcal{L}}{\partial w}$ "gradient" —— vector in $\mathbb{R}^d$

$H = \frac{\partial^2 \mathcal{L}}{\partial w^2}$ "Hessian" —— matrix in $\mathbb{R}^{d \times d}$

Second-order optimization: Newton's method

Take the Taylor expansion to second-order:

$$\mathcal{L}(w+\Delta w) \approx \mathcal{L}(w) + g^T \Delta w + \frac{\lambda}{2} \Delta w^T H \Delta w$$

and minimize the RHS with respect to Δw

Take derivative and set to zero $\quad g + \lambda H \Delta w = 0$

$$\Rightarrow \boxed{\Delta w = -H^{-1} g}$$

Newton's method

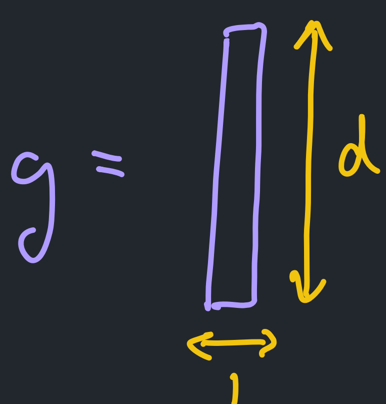
"pre-condition the gradient with the inverted Hessian"

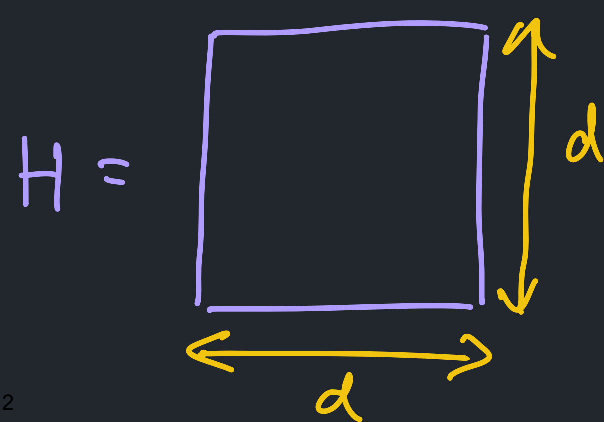
Second-order optimization: Problems w/ Newton

Newton's method $\boxed{\Delta w = -H^{-1} g}$ "pre-condition the gradient with the inverted Hessian"

Problems with Newton's method

- might converge to local max
  - can fix with "cubic regularization"
- d parameters $\Rightarrow$ Hessian is $d \times d$
  - too expensive even for "small" networks





Composite optimization: the GN decomposition

Gauss! Newton!

Suppose we have a "composite" objective $\mathcal{L} = \ell \circ f$ — e.g. error ℓ composed with neural net f

$$\frac{\partial \mathcal{L}}{\partial w} = \frac{\partial \ell}{\partial f} \cdot \frac{\partial f}{\partial w}$$

chain rule

$$\frac{\partial^2 \mathcal{L}}{\partial w^2} = \frac{\partial f}{\partial w} \cdot \frac{\partial^2 \ell}{\partial f^2} \cdot \frac{\partial f}{\partial w} + \frac{\partial \ell}{\partial f} \frac{\partial^2 f}{\partial w^2}$$

product rule + chain rule

we call the second result the "Gauss-Newton" decomposition of the Hessian

Composite optimization: the GN method

Given composite loss function $\mathcal{L} = \ell \circ f$

$$\frac{\partial^2 \mathcal{L}}{\partial w^2} = \frac{\partial f}{\partial w} \cdot \frac{\partial^2 \ell}{\partial f^2} \cdot \frac{\partial f}{\partial w} + \frac{\partial \ell}{\partial f} \frac{\partial^2 f}{\partial w^2}$$

full Hessian H — curvature of error — curvature of model

① for square loss, $\ell = \frac{1}{2}(f-y)^2$, we have $\frac{\partial^2 \ell}{\partial f^2} = 1$

② ignore the curvature of the model 🙃

$\Rightarrow$ Newton's method becomes

$$\boxed{\Delta w = -\left[\frac{\partial f}{\partial w}^{\top} \frac{\partial f}{\partial w}\right]^{-1} g}$$

Gauss Newton method

Composite optimization: Problems w/ GN method

$$\boxed{\Delta w = -\left[\frac{\partial f}{\partial w}\frac{\partial f}{\partial w}\right]^{-1} g}$$

Gauss Newton method

① requires computing extra derivatives $\frac{\partial f}{\partial w}$

② is it safe to ignore curvature of the model?

$$\underbrace{\frac{\partial^2 L}{\partial w^2}}_{\text{full Hessian } H} = \frac{\partial f}{\partial w} \cdot \underbrace{\frac{\partial^2 L}{\partial f^2}}_{\text{curvature of error}} \cdot \frac{\partial f}{\partial w} + \cancel{\frac{\partial L}{\partial f}\underbrace{\frac{\partial^2 f}{\partial w^2}}_{\text{curvature of model}}}$$

First-order optimization: Steepest descent

Take the Taylor expansion:

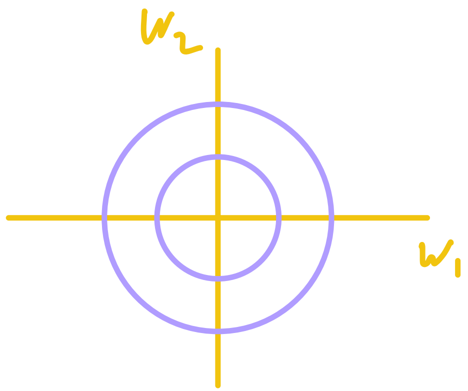
$$\mathcal{L}(w+\Delta w) = \mathcal{L}(w) + g^T \Delta w + \underbrace{\tfrac{1}{2} \Delta w^T H \Delta w + \dots}_{\text{non-linear part}}$$

$\Leftrightarrow$ model with $\frac{\lambda}{2} \|\Delta w\|^2$

model $\left\{ \boxed{\mathcal{L}(w+\Delta w) \approx \mathcal{L}(w) + g^T \Delta w + \frac{\lambda}{2} \|\Delta w\|^2} \right.$

First-order optimization: ℓ_2 steepest descent

model $\left\{ \boxed{\mathcal{L}(w+\Delta w) \approx \mathcal{L}(w) + g^T \Delta w + \underbrace{\frac{\lambda}{2}\|\Delta w\|_2^2}_{\text{Euclidean norm}}} \right.$



Euclidean balls

Minimize RHS of model wrt Δw

$\Rightarrow$ differentiate and set derivative to zero

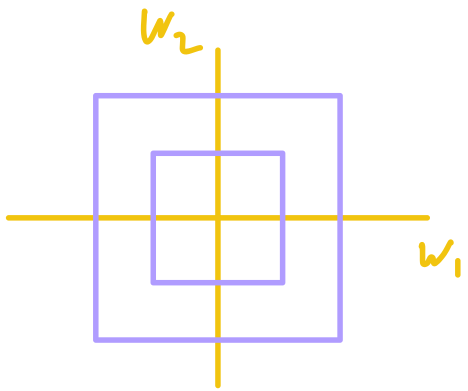
$\Rightarrow \quad g + \lambda \Delta w = 0$

$$\Rightarrow \quad \boxed{\Delta w = -\frac{1}{\lambda} g}$$

vanilla gradient descent!

First-order optimization: ℓ_∞ steepest descent

model $\left\{ \boxed{\mathcal{L}(w+\Delta w) \approx \mathcal{L}(w) + g^T \Delta w + \frac{\lambda}{2} \underbrace{\|\Delta w\|_\infty^2}_{\text{infinity norm}}} \right.$



infinity balls

Minimize RHS of model wrt Δw

$\Rightarrow$ <do homework>

$$\Rightarrow \boxed{\Delta w = -\frac{\|g\|_1}{\lambda}\,\mathrm{sign}(g)}$$

sign gradient descent!

First-order optimization: General steepest descent

$$\text{model}\left\{\ \boxed{\mathcal{L}(w+\Delta w) \approx \mathcal{L}(w) + g^T \Delta w + \underbrace{\frac{\lambda}{2}\|\Delta w\|^2}_{\text{general norm}}}\right.$$

minimizing the right hand side with respect to Δw has a "dual formulation"

$$\underset{\Delta w}{\operatorname{argmin}} \left[g^T \Delta w + \frac{\lambda}{2}\|\Delta w\|^2 \right] = \frac{\|g\|^\dagger}{\lambda} \underbrace{\underset{t:\|t\|=1}{\operatorname{argmax}}\; g^T t}_{\text{step direction}}$$

step size

where $\|\cdot\|^\dagger$ is the "dual norm" to $\|\cdot\|$

Heuristics for Scaling

How large should the weight updates be?

We want the "Goldilocks" update size...

... not too big, not too small

But always ask: $\boxed{\text{in which norm?}}$

<u>observation</u> a neural net is built out of weight matrices

perhaps we could try a matrix norm?

e.g. Frobenius norm

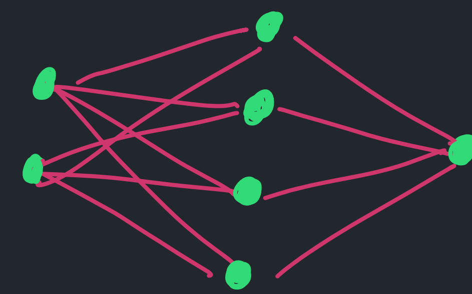
spectral norm

nuclear norm

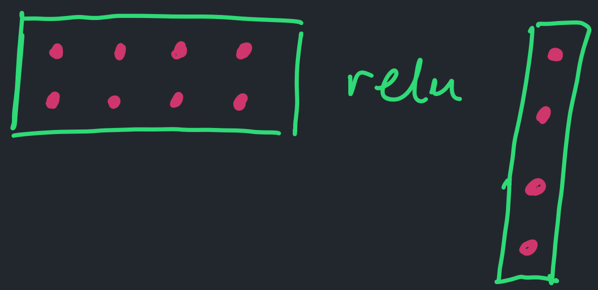
⋮

Perspectives on neural computation

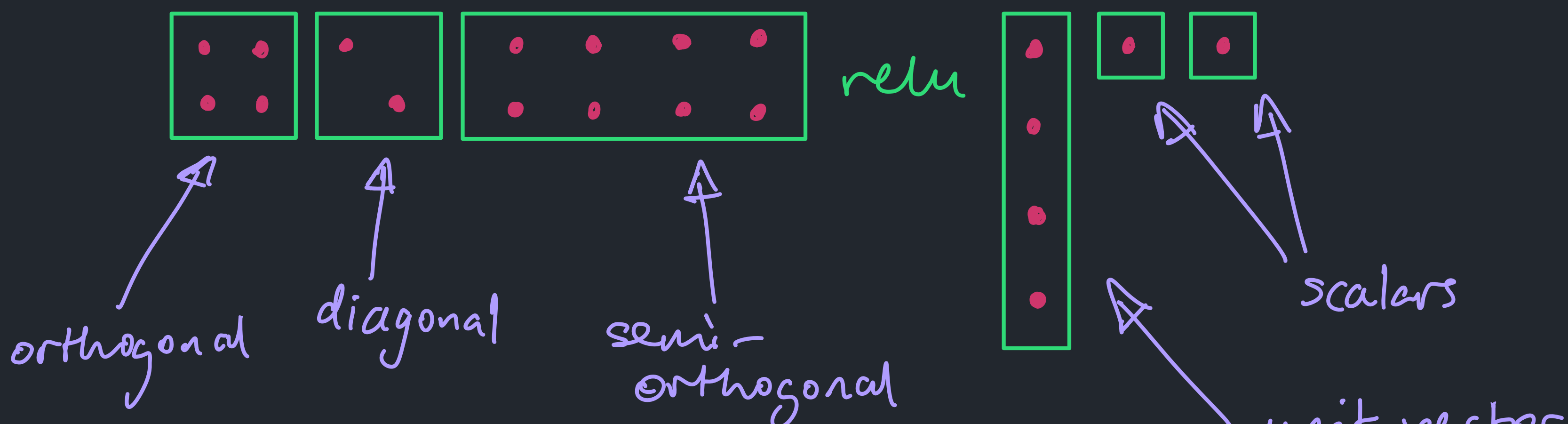
① NEURAL PERSPECTIVE



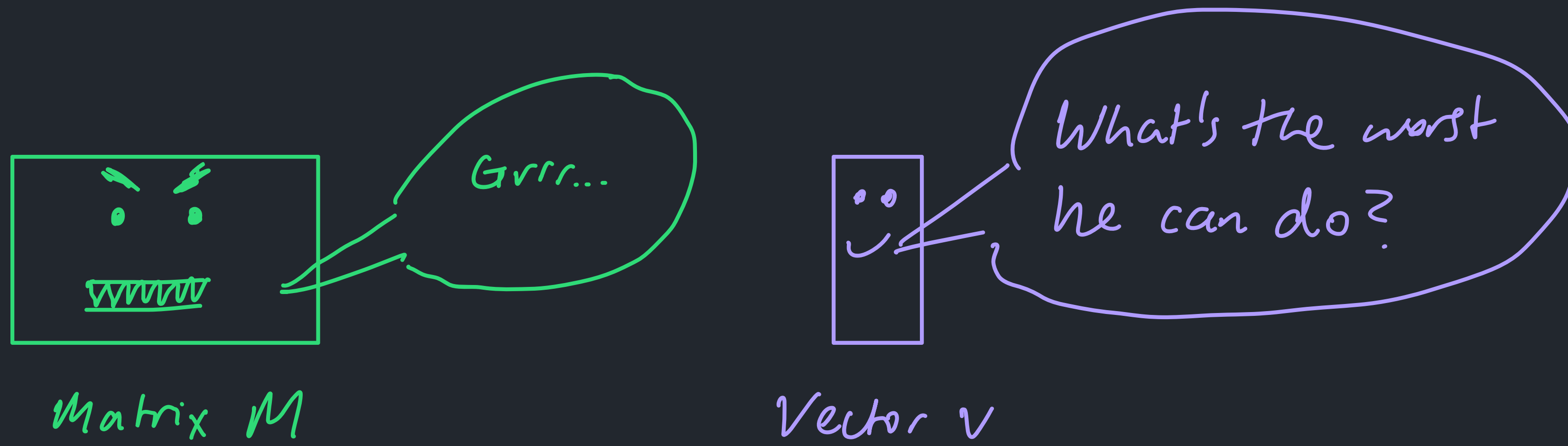
② TENSOR PERSPECTIVE



③ SPECTRAL PERSPECTIVE ...... using SVDs



The spectral norm

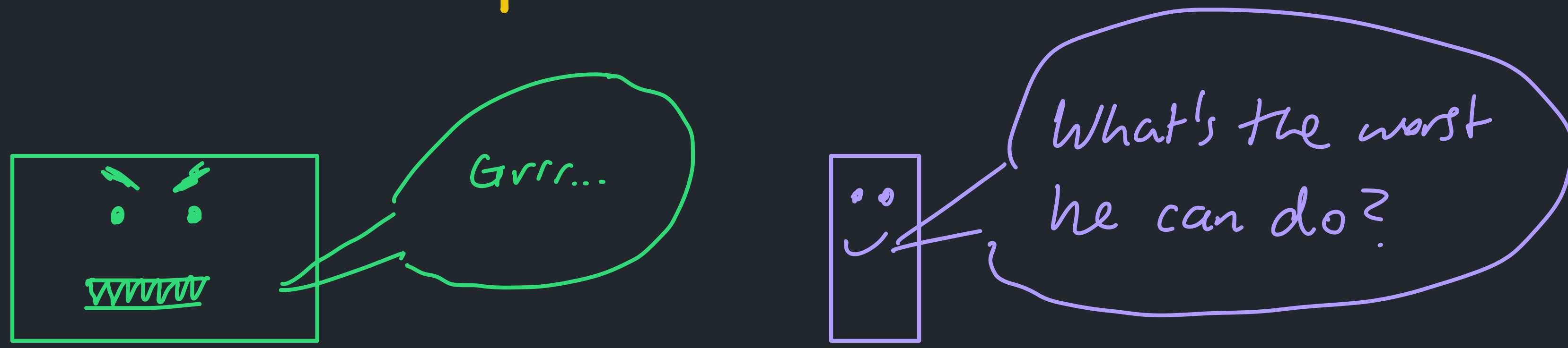


Spectral norm $\|M\|_* = \max\limits_{v \neq 0} \dfrac{\|Mv\|_2}{\|v\|_2}$

Answers question: how much can a matrix scale up the Euclidean norm of a vector?

<u>FACT</u> spectral norm $\equiv$ largest singular value

The RMS-RMS operator norm



Matrix M Vector $v \in \mathbb{R}^d$

Equip vectors in $\mathbb{R}^d$ with the RMS norm $\|\cdot\|_{RMS} = \frac{1}{\sqrt{d}} \|\cdot\|_2$

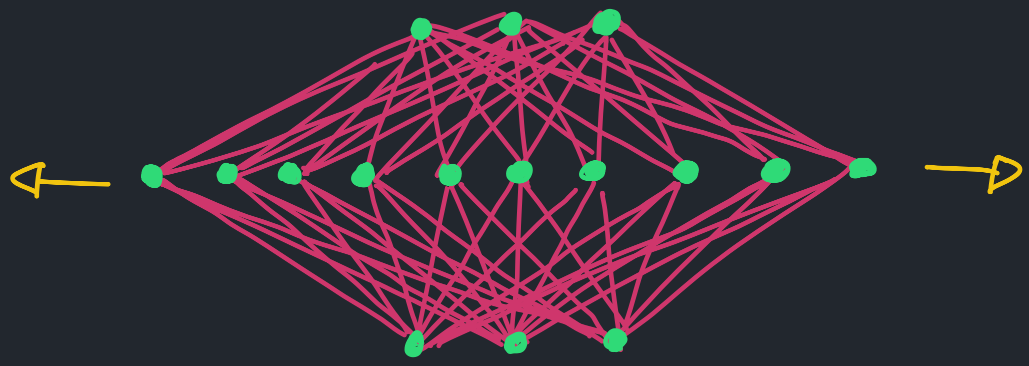
RMS-RMS operator norm $\|M\|_{RMS-RMS} = \max_{v \neq 0} \frac{\|Mv\|_{RMS}}{\|v\|_{RMS}}$

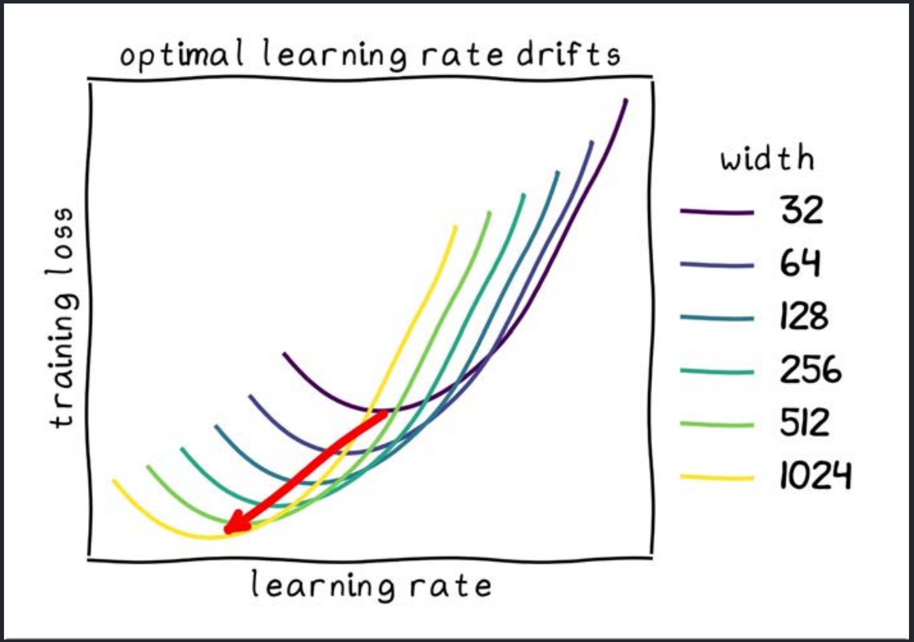
Interpretation $\|\cdot\|_{RMS-RMS}$ constrains how much a matrix can change the RMS norm of its input

Exercise show that $\|\cdot\|_{RMS-RMS} = \sqrt{\frac{d_{in}}{d_{out}}} \|\cdot\|_*$

Spectrally controlled weight updates

recall our scaling woes



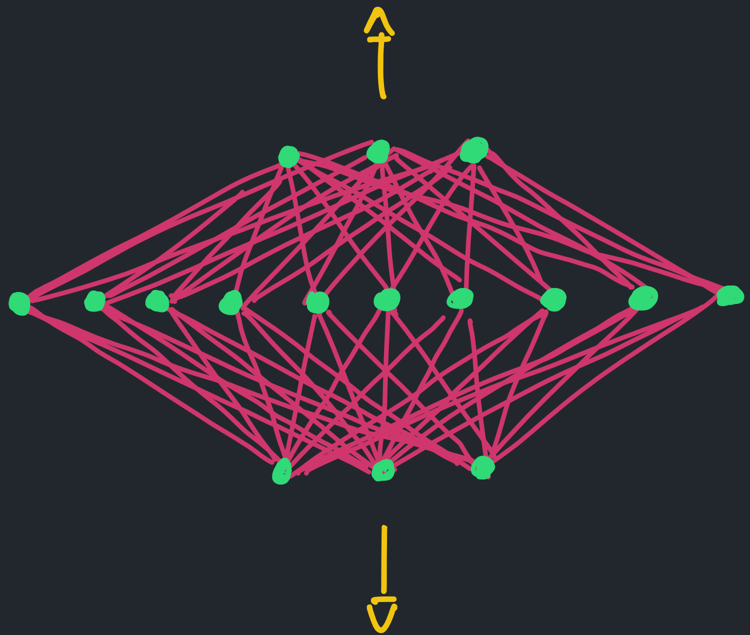


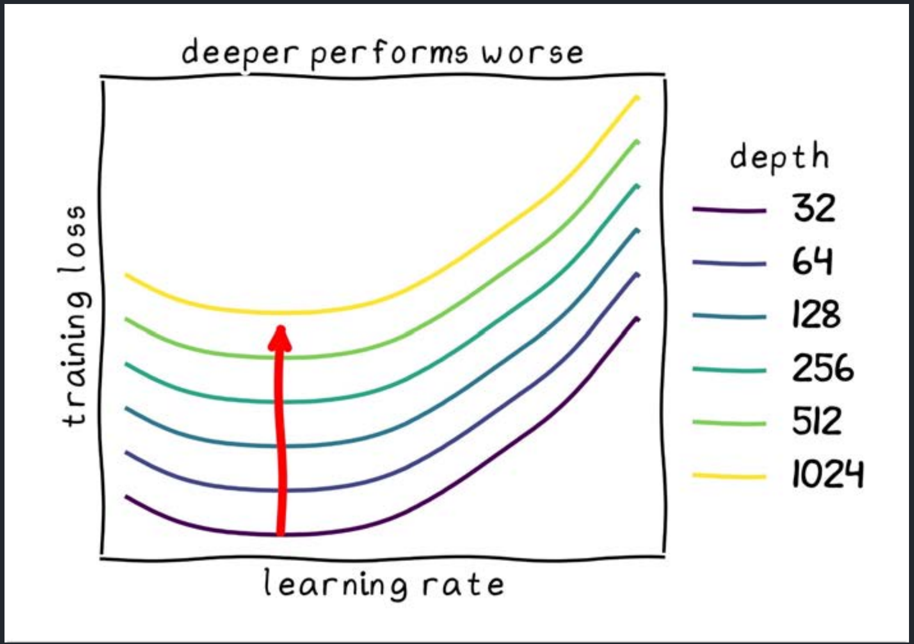
<u>claim</u>: to remove drift in optimal learning rate as width is varied, for all layers $l = 1, \ldots, L$, do:

1. initialize weights so that $\|W_l\|_{RMS-RMS} \sim 1$.
2. scale updates so that $\|\Delta W_l\|_{RMS-RMS} \sim 1$.

See homework!

Depth scaling





the trick seems to be to parameterize your residual block the "right" way

recall that $\lim_{L \to \infty} \left(1 + \frac{x}{L}\right)^L = \exp(x)$

so build your residual block like

$x \longrightarrow x + \frac{1}{L}\,\text{layer}(x)$?

needs more research

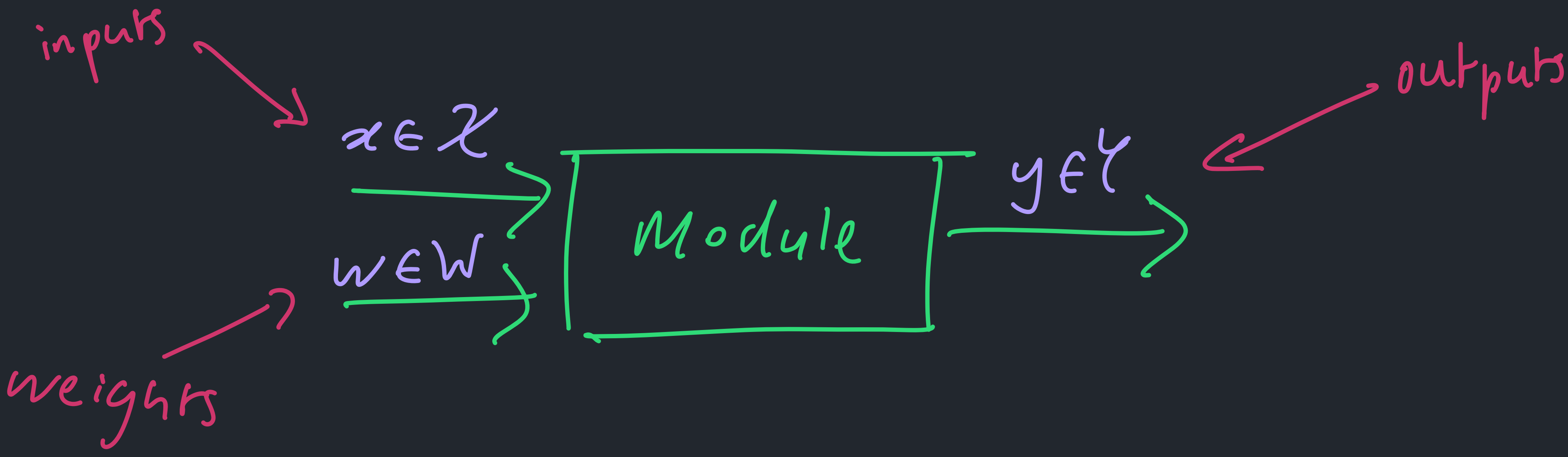
Building a Theory: Modularization

my research, so be skeptical

Building a modular theory

IDEA: if you want an optimization theory that handles complicated neural networks

... build the theory with the neural net



Write a library of "atomic modules"

<u>Definition</u> module M

M.forward $\quad \mathcal{W} \times \mathcal{X} \to \mathcal{Y}$

M.backward $\quad \mathcal{Y} \times \mathcal{W} \times \mathcal{X} \to \mathcal{W} \times \mathcal{X}$

M.norm $\quad \mathcal{W} \to \mathbb{R}$

Linear

Embedding

Conv2D

ReLU

} all with hand-specified forward, backward and norms

Write combination rules

e.g. module composition $M = M_2 \circ M_1$

M.forward

just compose M_2.forward with M_1.forward

M.backward

do the chain rule

M.norm

how should we combine M_2.norm and M_1.norm?

References

📕 Feature Learning in Infinite Width Neural Networks
Yang & Hu (2020)

📒 Infinite Limits of Multi-Head Transformer Dynamics
Bordelon, Chaudhury & Pehlevan (2024)

📙 Scalable Optimization in the Modular Norm
Large et al (2024)

📓 Universal Majorization-Minimization Algorithms
Streeter (2023)