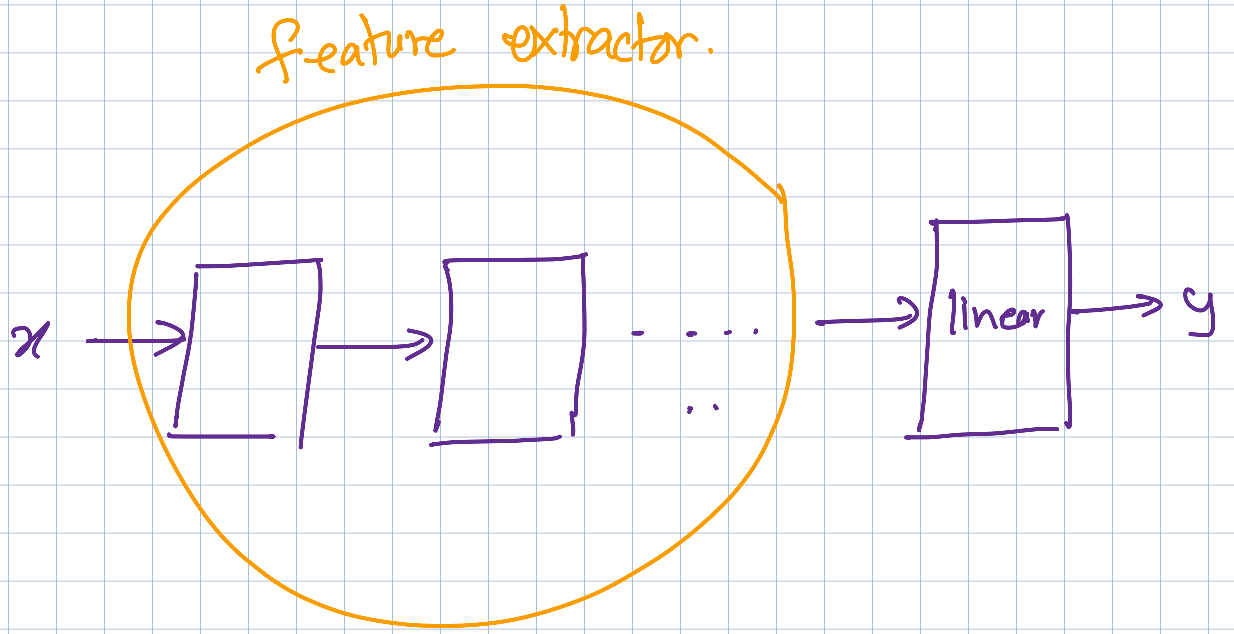
Lecture 4

- Feature perspective and Taylor expansion
- Adam revisited
- GD revisited.
- Initialization



A second perspective...

Network: $f(\vec{x}, \vec{\theta})$ $\vec{x}$: data

$\vec{\theta}$: parameters

Focus on this as a function of $\vec{\theta}$.

$\vec{\theta} \in \mathbb{R}^m$ say.

Consider $f: \mathbb{R}^m \to \mathbb{R}$ e.g. linear regression, binary classification

$$f(x, \theta + \Delta\theta) = f(x, \theta) + \boxed{\langle \nabla_\theta f, \Delta\theta \rangle} + \cdots \text{ higher order terms}$$

$\longrightarrow$ Taylor expansion.

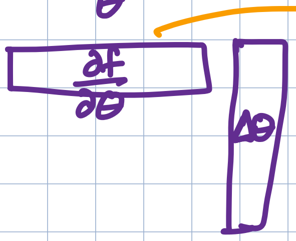
"Linear approximation of the network"

Taylor approximation only holds true in a neighbourhood

Often true in practice

Lazy Training assumption: Weights are moving by very small amounts.

$\langle \nabla_\theta f, \Delta\theta \rangle \longrightarrow$ linear term



entries are "features"!

Recall:

$$\nabla_\theta f = \left(\frac{\partial f}{\partial \theta}\right)^T$$

Gradient is transpose of the derivative.

So now if we revisit descent algorithms from this perspective ...

Revisit Adam (SignSGD).

Recall

$$\ell(\vec{x}, \vec{\theta_i} + \vec{\Delta\theta_i}) = \ell(\vec{x}, \vec{\theta_i}) + \frac{\partial \ell}{\partial \theta}\Big|_{\theta=\theta_i} \vec{\Delta\theta}$$

$$\vec{\theta}_{i+1} = \vec{\theta_i} - \eta \cdot \text{step}$$

Here ℓ is the loss

Tension: Want large steps to be able to converge fast.

Want small steps to make sure approximation still holds.

Idea: Bound the size of your step: $\|\Delta\theta\|_\infty \leq \eta$

$\hookrightarrow \max_j \|\Delta\theta[j]\| \leq \eta$

Want:

$$\underset{\|\vec{\Delta\theta}\|_\infty \leq \eta}{\text{argmin}} \frac{1}{n}\sum_{\ell=1}^{n}\left[\cancel{\ell(\vec{x_\ell}, \vec{\theta_i})} + \left\langle \nabla_\theta \ell\Big|_{\substack{x=x_\ell \\ \theta=\theta_i}}, \vec{\Delta\theta} \right\rangle\right]$$

$$= \underset{\|\vec{\Delta\theta}\|_\infty \leq \eta}{\text{argmin}} \left\langle \underbrace{\left(\frac{1}{n}\sum_{\ell=1}^{n} \nabla_\theta f\Big|_{\substack{x=x_\ell \\ \theta=\theta_i}}\right)}_{\nabla_\theta f_{\text{batch}}}, \vec{\Delta\theta} \right\rangle$$

$$= \underset{\forall j\ |\Delta\theta[j]| \leq \eta}{\operatorname{argmin}} \sum_j \left(\underbrace{\left(\frac{1}{n} \sum_{l=1}^{n} \nabla_\theta f \Big|_{\substack{x = x_l \\ \theta = \theta_i}} \right)}_{\nabla_\theta f_{batch}} [j],\ \Delta\vec{\theta}[j] \right)$$

Do each optimization individually

$$\underset{|\Delta\theta[j]| \leq \eta.}{\operatorname{argmin}} \quad \boxed{\nabla_\theta f_{batch}} \ \Delta\theta[j].$$

$$-\eta \leq \Delta\theta[j] \leq \eta$$

$$\Delta\theta_j = +\eta \qquad \text{if } \nabla_\theta f_{batch}[j] < 0$$

$$= -\eta \qquad \text{if } \nabla_\theta f_{batch}[j] > 0.$$

$$\Rightarrow \quad \vec{\Delta\theta}^* = -\eta \cdot \operatorname{sign}(\nabla_\theta f_{batch}).$$

Two-norm constraint: $\|\Delta\theta\|_2 \leq \eta$

Stepping back

Why do we standardize data in ML?

$\rightarrow$ Want features on similar orders of magnitude. $\rightarrow$ else large features will dominate.

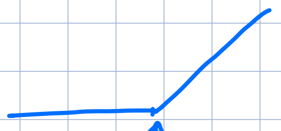
↳ Better conditioning

$\rightarrow$ Numerical issues and numerical precision.

$\rightarrow$ Want to be able to "use" the non-linearity for the expressive power of neural nets.

Consider ReLU. Scalar case

$$\begin{tikzcd}
 & w \arrow[d] & b \arrow[d] & & \\
x \arrow[r] & \odot \arrow[r] & \oplus \arrow[r] & \boxed{\text{ReLU}} \arrow[r] & {}
\end{tikzcd}$$



$\max(0, wx+b)$

$\uparrow$ elbow @ $-\frac{b}{w}$

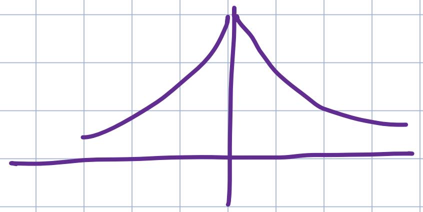
Say we initialize

$b \sim N(0,1)$

$w \sim N(0,1)$

then $-\frac{b}{w} \sim$ Cauchy distribution.

PDF : $\frac{1}{\pi}\left[\frac{1}{x^2+1}\right]$



Properties: "0-mean"

∞ variance.

But mostly around 0.

50%	of probability mass in	-1 to 1
70%		-2 to $+2$
80%		-3 to $+3$
90%		-7 to $+7$

We want our values to be where the non-linearity is interesting

→ True for other non-linearities too   Tanh, sigmoid

What happens if we don't do this?

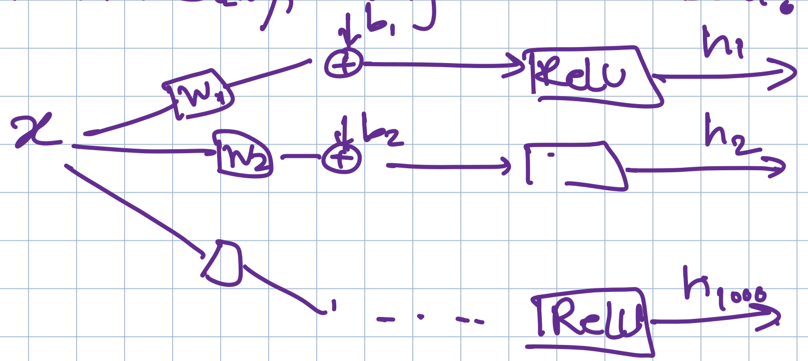
Suppose our data was in the range 100 to 200.

Probability of a corner landing in that range is about 0.16%

So only 1-2 units out of 1000 may have corners there.

Lose expressive power :(

Mathematically, why is this bad?



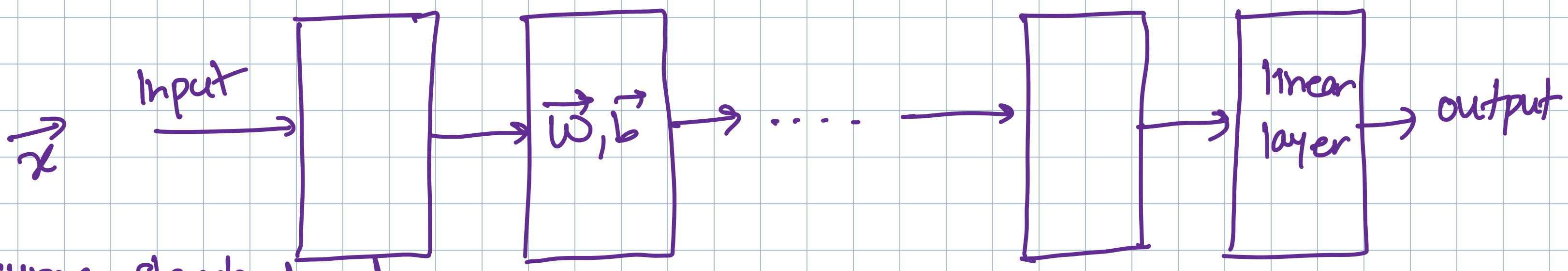
If no corners:

$$h_i = \begin{cases} 0 & \forall x \in (100,200) \\ \alpha x + \beta & \forall x \in (100,200) \end{cases}$$

$\beta_i = 0$

$$\begin{bmatrix} \alpha_1 x_1 + \beta_1 & \cdots & \alpha_{1000} x_1 + \beta_{1000} \\ \alpha_1 x_2 + \beta_1 & & \\ \vdots & & \\ \alpha_1 x_n + \beta_1 & & \alpha_{1000} x_n + \beta_{1000} \end{bmatrix}$$

Initialization



Assume standardized
mean 0
variance 1

How to initialize?

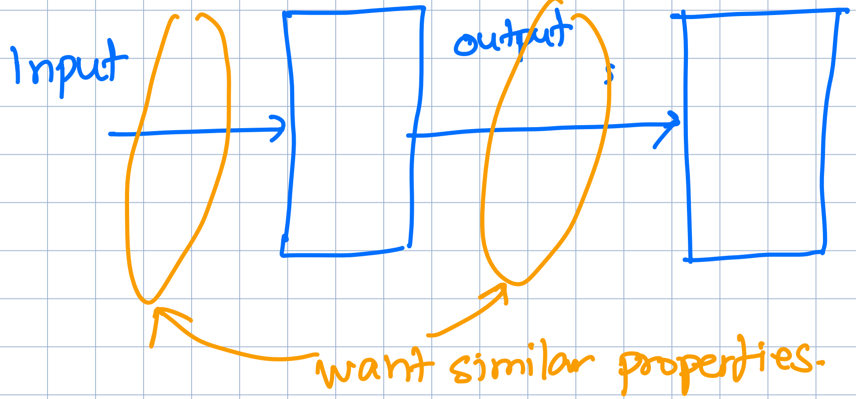
→ Set everything to 0?

But then all inputs from the previous layer get multiplied by zero

Gradients are zero , weights never more.

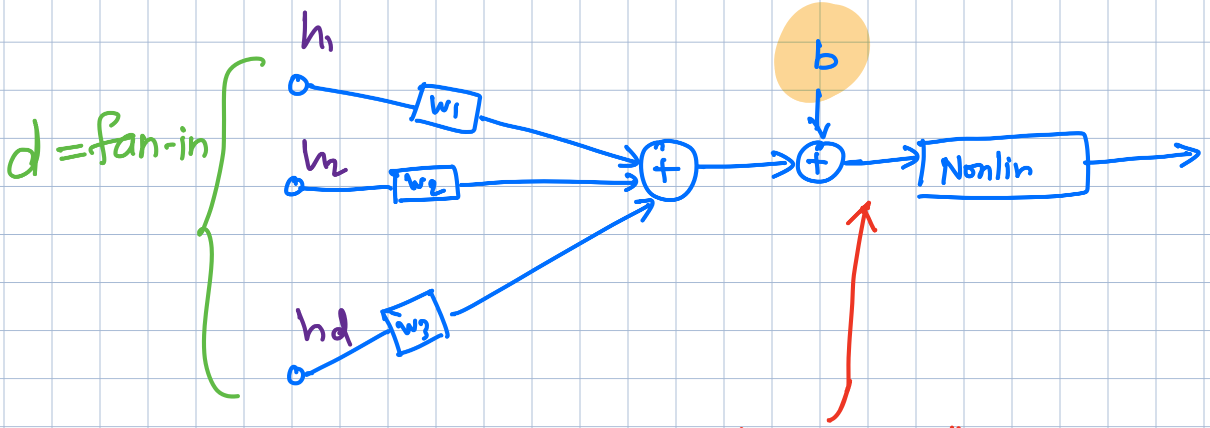
→ So what do we do?

Key observation: Output activations from one layer are inputs to the next layer.



Goal: Can each layer have inputs that start "standardized"?

Xavier initialization (weights, i.e. parameters that multiply data)



Want: input to nonlinearity is "standardized."

Approx "0-mean"
"unit-variance"

Adding d terms: $w_i \sim N(0, \underset{\uparrow \text{variance}}{\tfrac{1}{d}})$

$$\mathrm{var}\left(\sum w_i h_i\right)$$

$$= E\left[\left(\sum w_i h_i\right)^2\right] - \left(E\left(\cancel{\sum w_i h_i}\right)\right)^2 \rightarrow 0$$

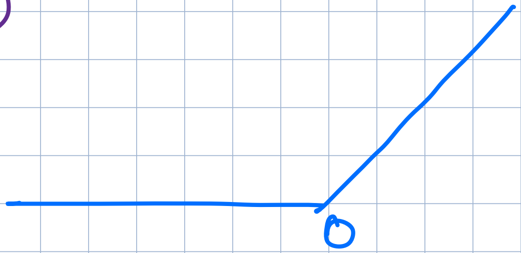
$$= \sum_d E[w_i^2] \cdot E[h_i^2]$$ because independence and zero-mean.

He Initialization

ReLU non-linearities. on previous layer

Observation: Output of a ReLU is zero half the time.

$$w_i \sim N\left(0, \frac{1}{d/2}\right) = N\left(0, \frac{2}{d}\right)$$



Bias initialization

(a) Initialize at 0.

(b) Use other small number 0.01.

(c) Treat as Xavier with initialization $d+1$.