Initialization

Gabriel Hope

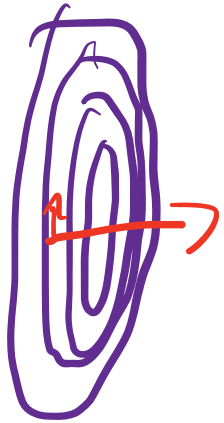
Adam

Can we combine adaptive scaling and momentum?

Adam

Update *velocity*

$$\mathbf{v}^{(k+1)} \longleftarrow \beta_1 \mathbf{v}^{(k)} + (1 - \beta_1) \nabla_{\mathbf{w}} \mathbf{Loss}(\mathbf{w}^{(k)}, \mathbf{X}, \mathbf{y})$$



Update *scaling*

$$\mathbf{s}^{(k+1)} \longleftarrow \beta_2 \mathbf{s}^{(k)} + (1 - \beta_2)(\nabla_{\mathbf{w}} \mathbf{Loss}(\mathbf{w}^{(k)}, \mathbf{X}, \mathbf{y}))^2$$

Update weights

$$\mathbf{w}^{(k+1)} \longleftarrow \mathbf{w}^{(k)} - \alpha \frac{\mathbf{v}^{(k+1)}}{\sqrt{\mathbf{s}^{(k+1)} + \epsilon}}$$

Adam

Update *velocity*

$$\mathbf{v}^{(k+1)} \longleftarrow \beta_1 \mathbf{v}^{(k)} + (1 - \beta_1) \nabla_{\mathbf{w}} \mathbf{Loss}(\mathbf{w}^{(k)}, \mathbf{X}, \mathbf{y})$$

Update *scaling*

$$\mathbf{s}^{(k+1)} \longleftarrow \beta_2 \mathbf{s}^{(k)} + (1 - \beta_2) (\nabla_{\mathbf{w}} \mathbf{Loss}(\mathbf{w}^{(k)}, \mathbf{X}, \mathbf{y}))^2$$

Modified weight update:

$$\mathbf{w}^{(k+1)} \longleftarrow \mathbf{w}^{(k)} - \alpha \frac{\frac{\mathbf{v}^{(k+1)}}{(1-\beta_1^k)}}{\sqrt{\frac{\mathbf{s}^{(k+1)}}{(1-\beta_2^k)}} + \epsilon}$$

learning rate

velocity

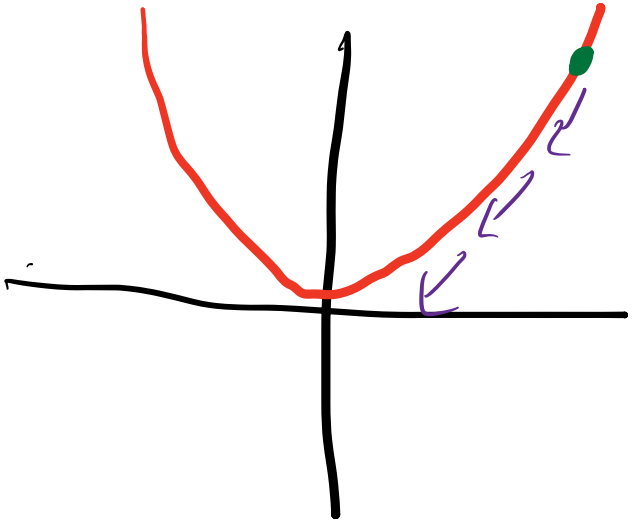
scale

Adam

At step 0:

$$\mathbf{v}^{(0)} = \mathbf{0}, \quad \mathbf{s}^{(0)} = \mathbf{0}$$

$$\frac{\mathbf{v}^{(k+1)}}{(1-\beta_1^k)} = \frac{\beta_1 \mathbf{0} + (1-\beta_1)\nabla_{\mathbf{w}}\mathbf{Loss}(\mathbf{w}^{(k)}, \mathbf{X}, \mathbf{y})}{(1-\beta_1^1)} = \nabla_{\mathbf{w}}\mathbf{Loss}(\mathbf{w}^{(k)}, \mathbf{X}, \mathbf{y})$$



$v^{(k+1)} = \beta_1 \frac{v^{(k)}}{0} + (1-\beta_1)\nabla_w$

$k=0$

as $k \to \infty$

$(1-\beta^k) \to 1$

e.g. $\beta = 0.9$

$(0.9)^{100} \approx 0$

$\to (1-\beta^{100}) \approx 1$

Summary of gradient descent issues

Updates are too slow

- Stochastic/minibatch gradient descent

SGD gradients are very noisy (high variance)

- Increase batch size, use momentum

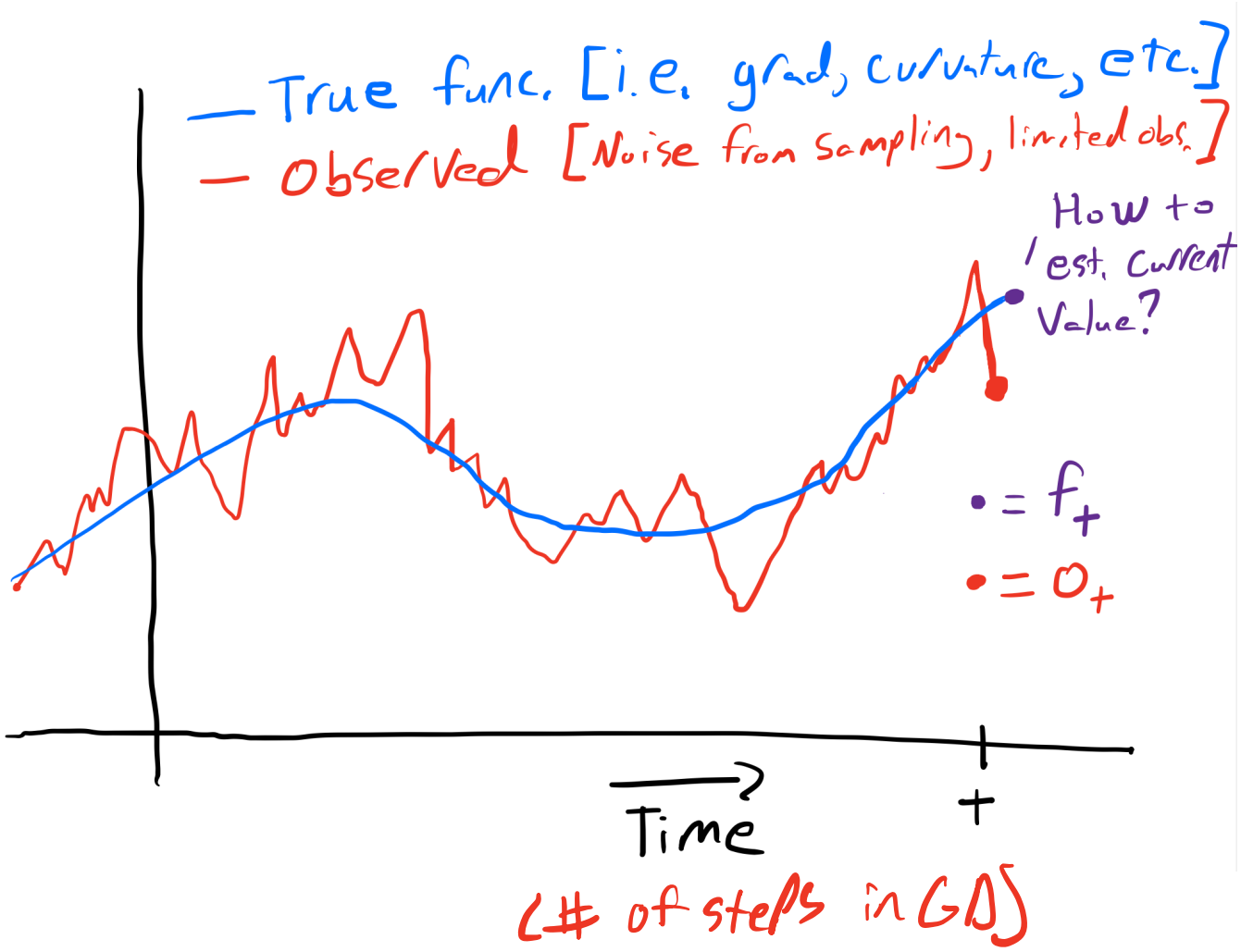
Stuck at saddle points or shallow optima

- Use momentum

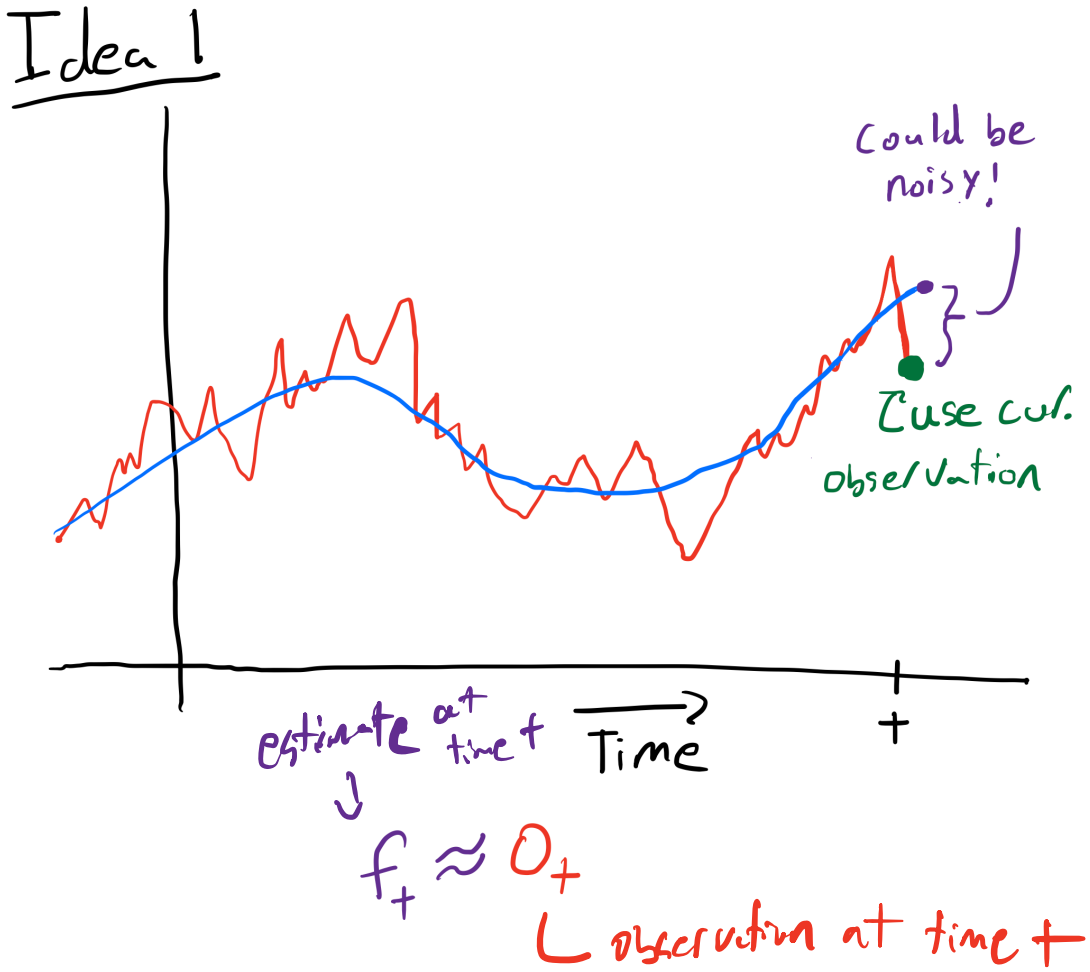
Inconsistant scaling of the gradient

- Use RMSProp scaling

Exponential moving average (EMA)

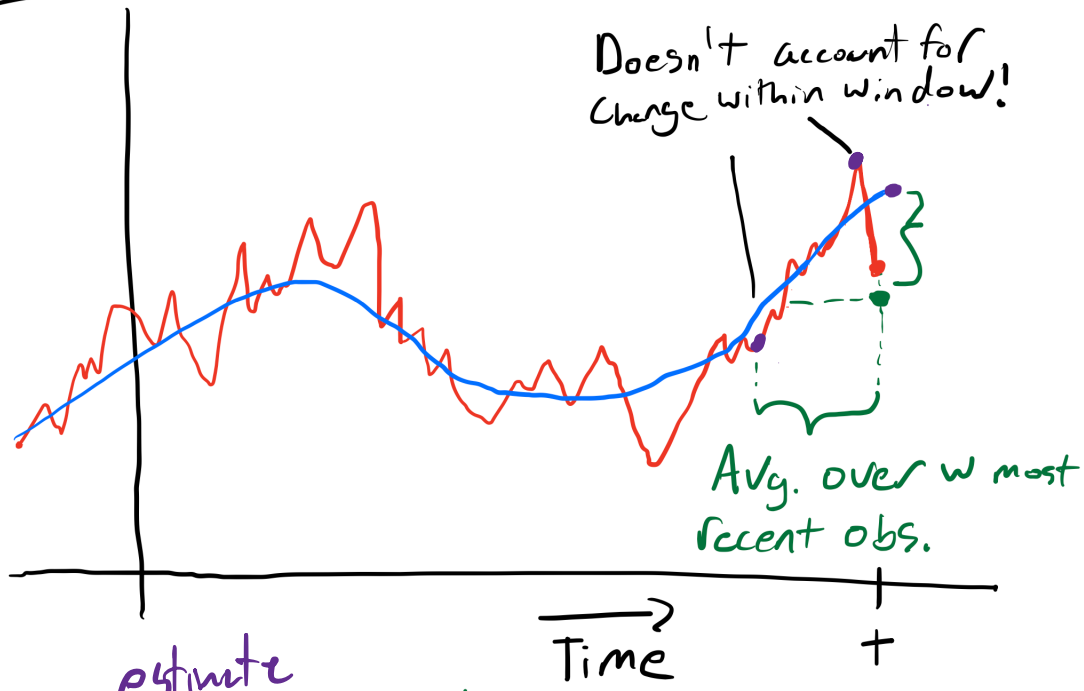


Exponential moving average (EMA)



Exponential moving average (EMA)

Idea 2 [Box avg.]



estimate

$$f_t \approx \frac{1}{w} \sum_{i=t-w}^{t} o_t$$

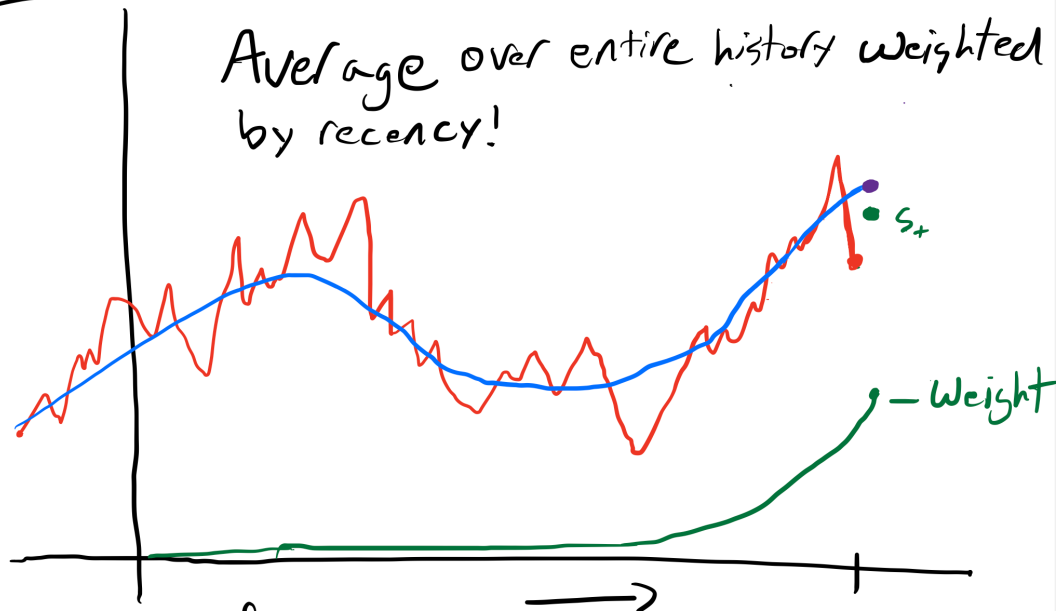
↳ Average obs. over a window of size w

Exponential moving average (EMA)

$$\sum_{t=0}^{T} \text{weight} = 1$$

$$\sum_{t=0}^{T-1} \text{weight} = \alpha$$

Idea 3 [EMA]



recursive def.

$\alpha \in [0, 1]$

$f_t \approx S_t$ — estimate at time t

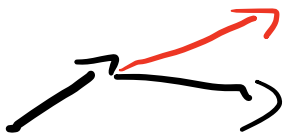
$S_t = (1-\alpha) O_t + \alpha S_{t-1}$

O_t: current observation

S_{t-1}: prev. at $t-1$

Exponential moving average (EMA)

momentum



Recursive def.

$f_t \approx S_t \qquad S_t = (1-\alpha) O_t + \alpha S_{t-1}$

$S_{t-1} = (1-\alpha) O_{t-1} + \alpha S_{t-2}$.

$\rightarrow f_t \approx (1-\alpha) O_t + (1-\alpha)\alpha O_{t-1} + (1-\alpha)\alpha^2 O_{t-2} + \ldots$

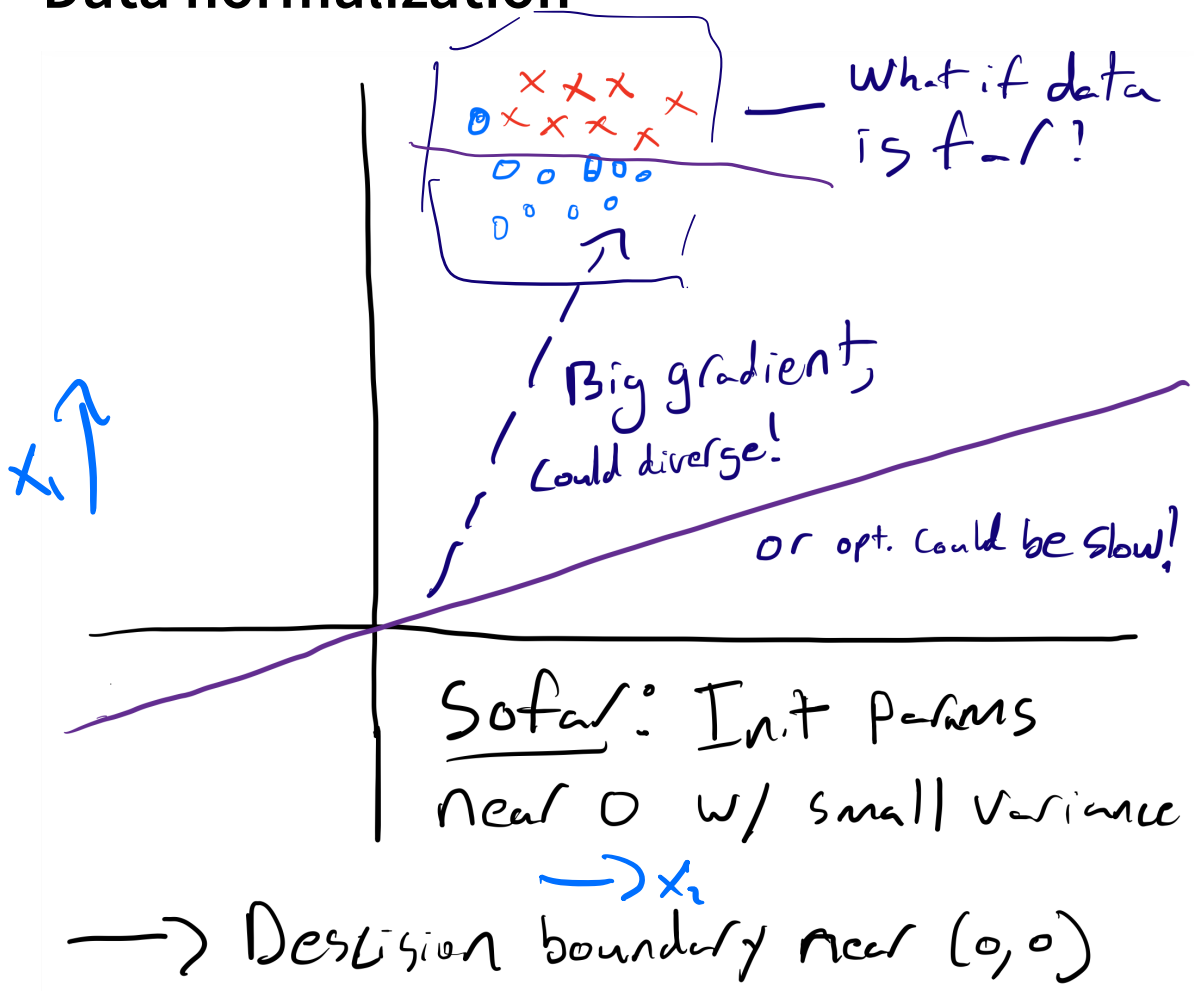
Ex. $\alpha = 0.9$

$f_t \approx 0.1\, O_t + 0.09\, O_{t-1} + 0.081\, O_{t-2} + 0.0729\, O_{t-3} + \ldots$

≈ 0

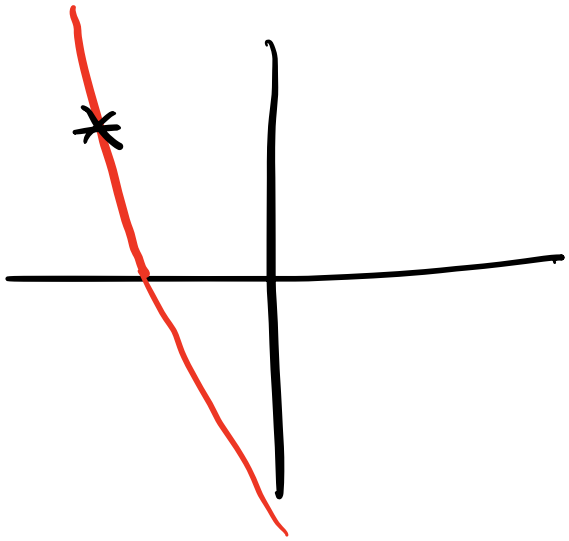
$+ \square\, O_{t-100}$

Data normalization

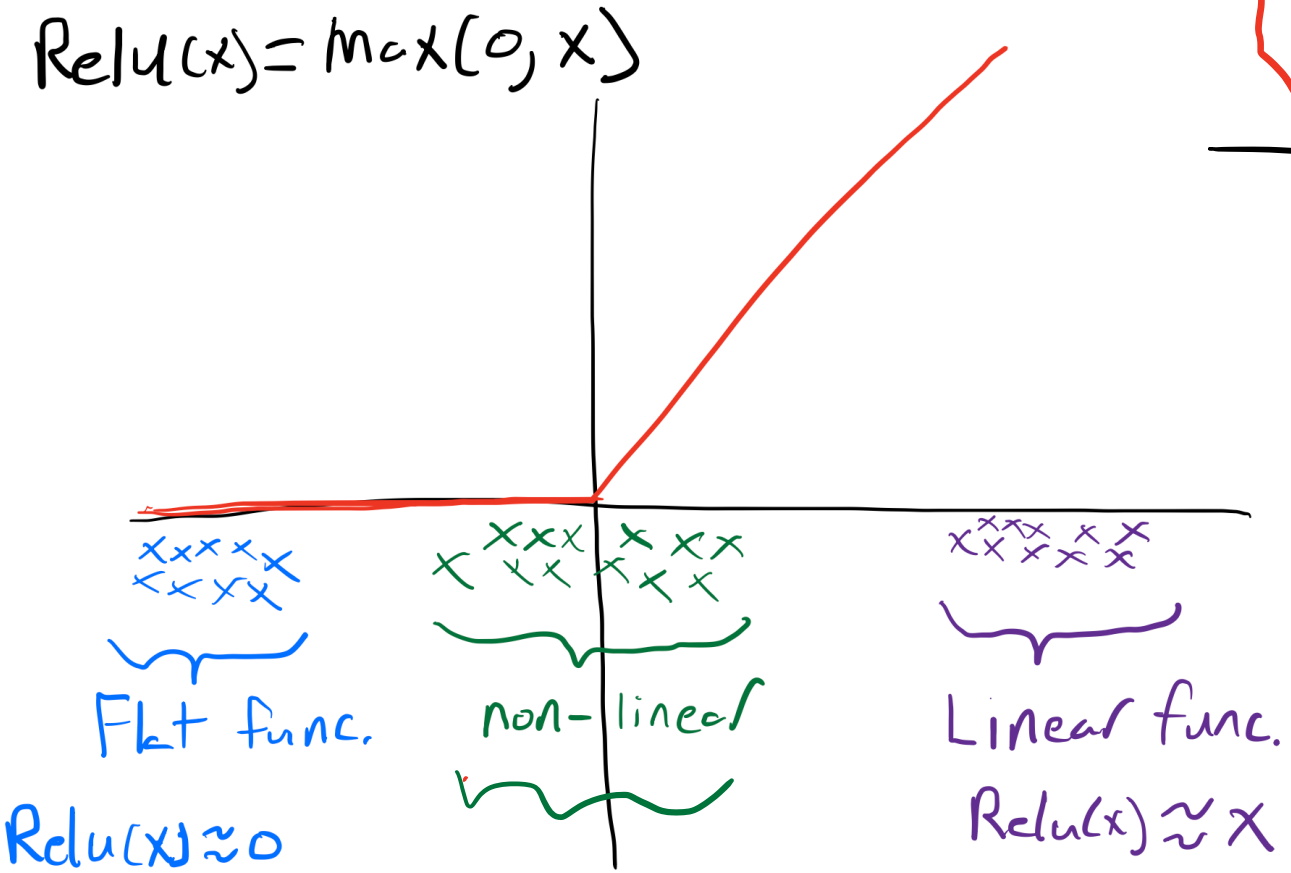


$$f(x) = \sum x_i w_i$$

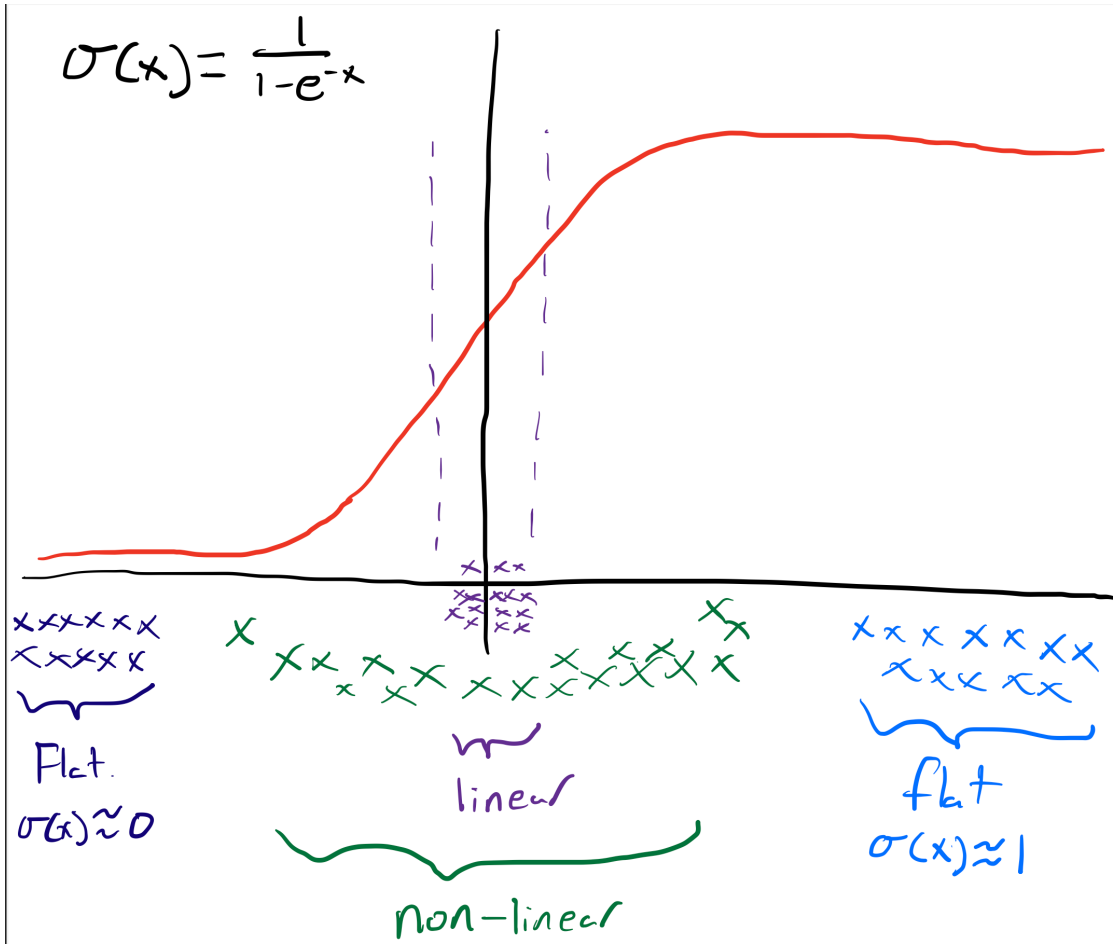
What about



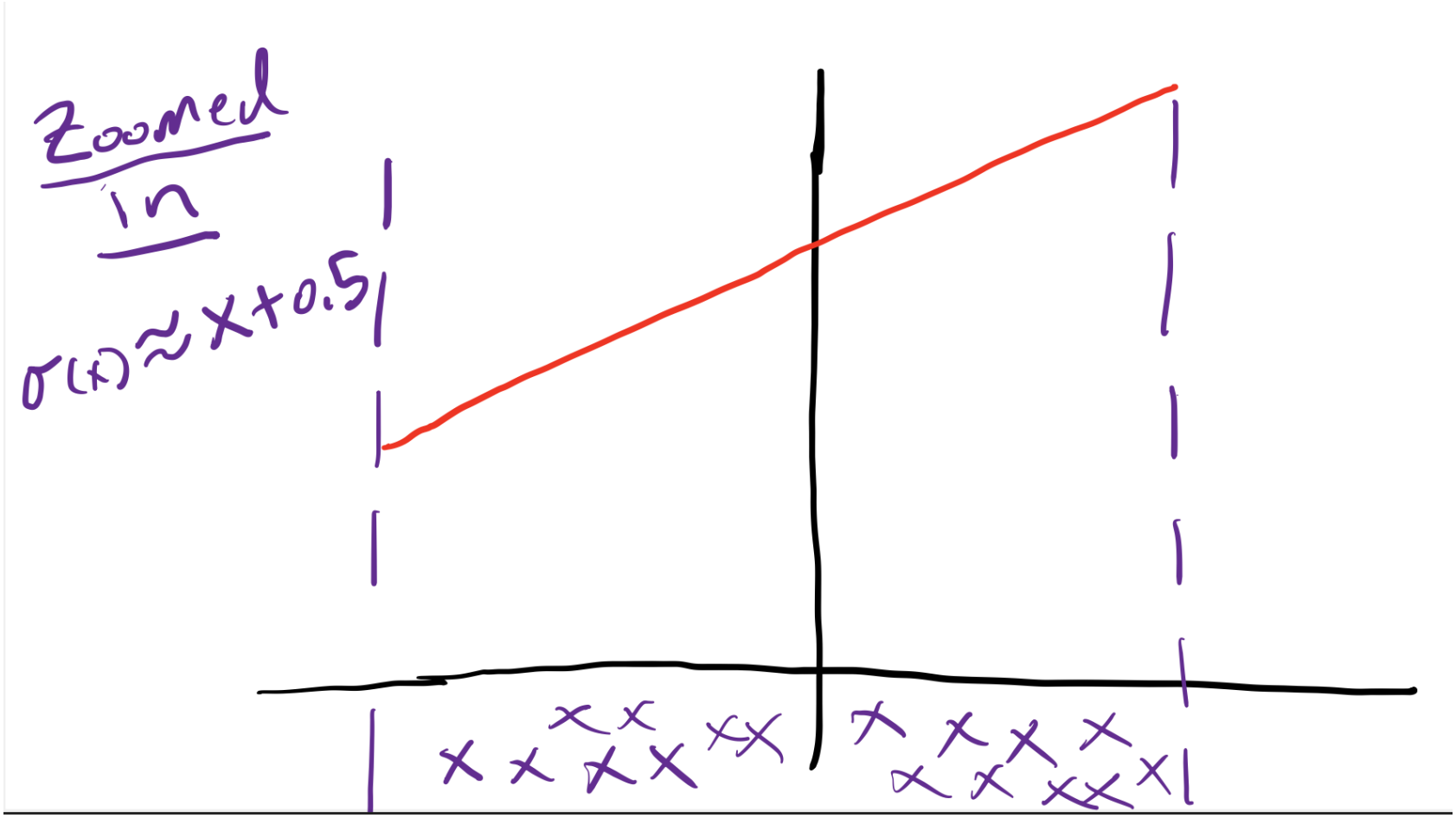
Data normalization



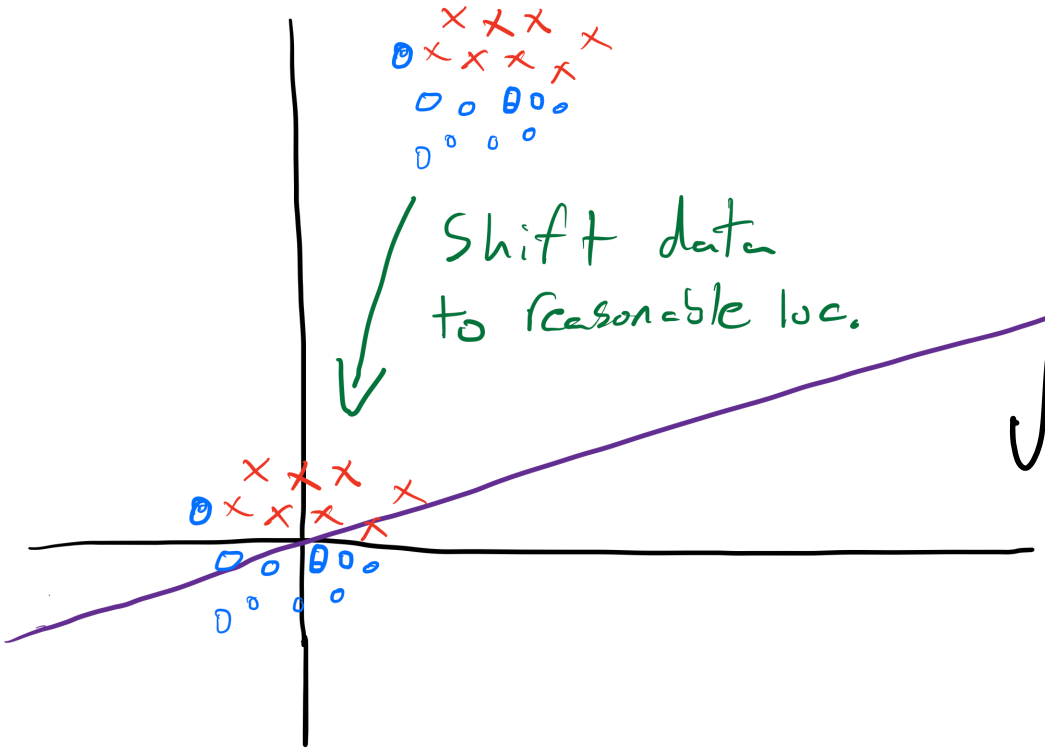
Data normalization



Data normalization



Data normalization



$X_1, X_2 \ldots, X_N$

$\frac{1}{N}\sum_{i=1}^{N} X_i \approx 0$ ← Want

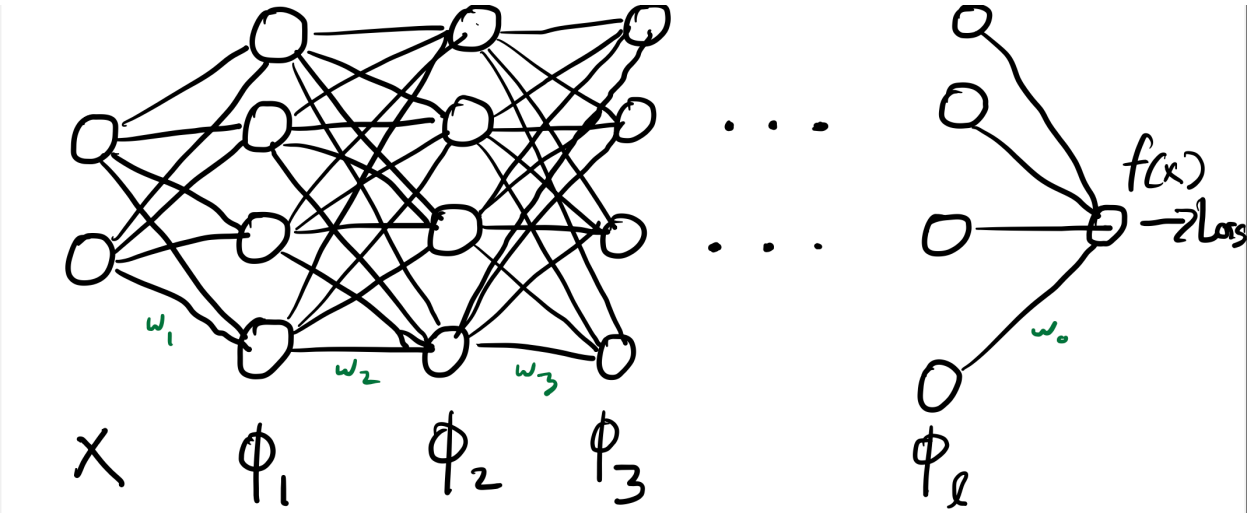
$\sqrt{\frac{1}{N}\sum_{i=1}^{N}(X_i - \bar{X})^2} \approx 1$ ← Want

Normalization

$X_i \longrightarrow \dfrac{X_i - \frac{1}{N}\sum_{j=1}^{N} X_j}{\sqrt{\frac{1}{N}\sum_{i=1}^{N}(X_i - \bar{X})^2}}$

Decision boundary near (0, 0)

Vanishing and exploding gradients



$\phi_1 = \sigma(x^T w_1 + b)$

$\phi_2 = \sigma(\phi_1^T w_2 + b)$

$\phi_3 = \sigma(\phi_2^T w_2 + b)$

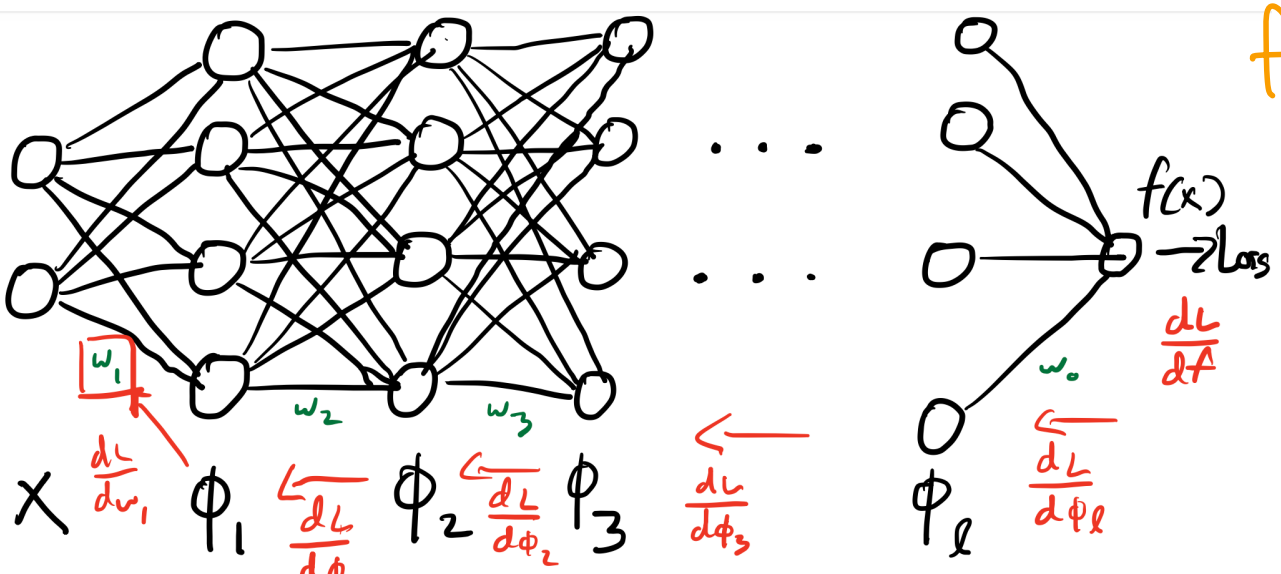
$\vdots$

many layers

$f = \phi_\ell^T w_o + b \quad L = \text{Loss}(f)$

ℓ Layers

Vanishing and exploding gradients



$$f_2(f_1(x)) = \frac{df_2}{df_1}\cdot\frac{df_1}{dx}$$

$$\frac{dL}{dw_1} = \frac{d\phi_1}{dw_1}\cdot\frac{d\phi_2}{d\phi_1}\cdot\frac{d\phi_3}{d\phi_2}\cdots\frac{d\phi_\ell}{d\phi_{\ell-1}}\frac{df}{d\phi_\ell}\cdot\frac{dL}{df}$$

$$\frac{d}{dw_1}\sigma(x^T w_1 + b) = x\,\sigma'(xw_1+b)$$

$2, -5, 3, -6, \ldots$

$= 180$

$$\frac{dL}{dw_1} = x\sigma'(xw_1+b)\left(\prod_{i=2}^{\ell}\frac{d\phi_i}{d\phi_{i-1}}\right)\frac{df}{d\phi_\ell}\cdot\frac{dL}{df}$$

for all $\frac{d\phi_i}{d\phi_{i-1}}$

$$\left|\frac{d\phi_i}{d\phi_{i-1}}\right| \approx M$$

Scale ↓

$$\left|\frac{dL}{dw_1}\right| \approx |x|\,|\sigma'(xw_1+b)|\prod_{i=2}^{\ell}\left|\frac{d\phi_i}{d\phi_{i-1}}\right|$$

Vanishing and exploding gradients

$\approx M^{20}$

$.9^{20} \approx 0$

2^{20}

If $M > 1 \longrightarrow \left|\frac{dL}{dw_1}\right| >> 1$

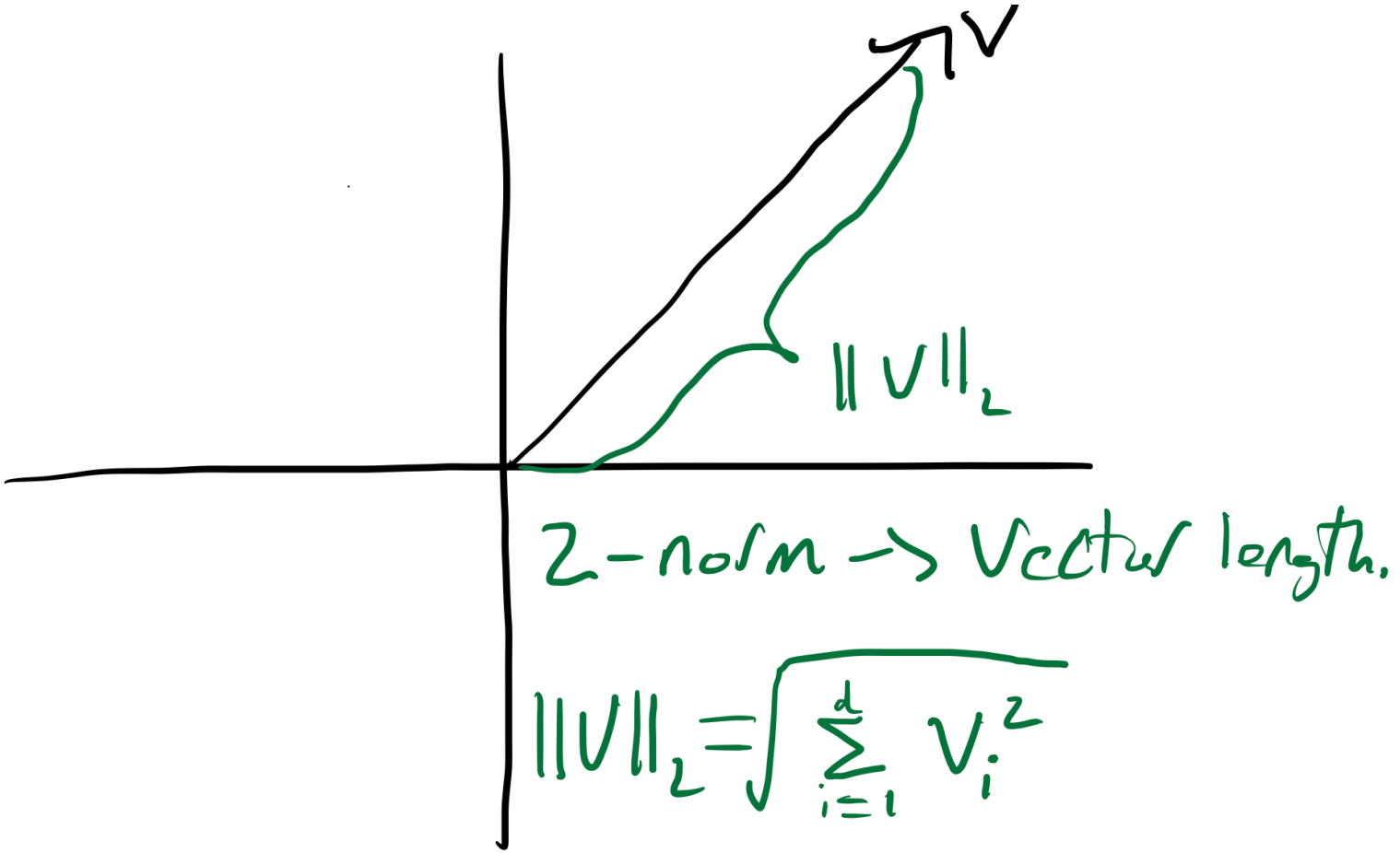
Gradient explodes!

If $M < 1 \longrightarrow \left|\frac{dL}{dw_1}\right| \approx 0$

Gradient Vanishes

Both bad for gradient descent!

Gradient clipping



Gradient clipping

Explicitly clip the gradient to prevent it form becoming too large.

ϵ = limit

Clip by value

$$\mathbf{clip}_{\text{value}}\left(\mathbf{x}, \epsilon\right) = \begin{bmatrix} \min(\max(x_1, -\epsilon), \epsilon) \\ \min(\max(x_2, -\epsilon), \epsilon) \\ \vdots \end{bmatrix}$$

e.g. gradient

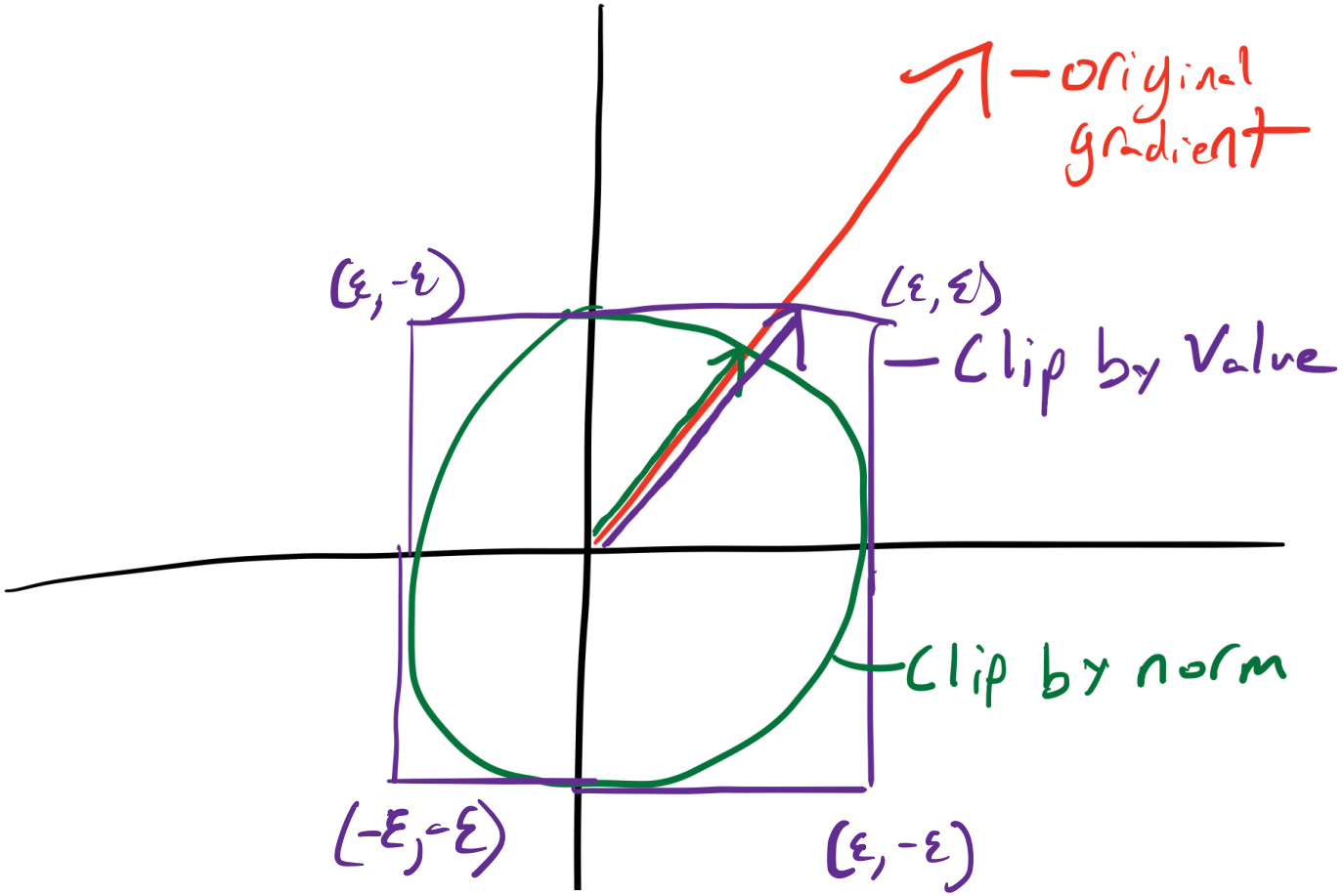
$$\mathbf{clip}_{\text{norm}}\left(\mathbf{x}, \epsilon\right) = \begin{cases} \frac{\epsilon \mathbf{x}}{\|\mathbf{x}\|_2} & \textbf{if: } \|\mathbf{x}\|_2 > \epsilon \\ \mathbf{x} & \textbf{if: } \|\mathbf{x}\|_2 \leq \epsilon \end{cases}$$

– enforce that x has length ϵ

Clipped gradient descent

$$\mathbf{w}^{(k+1)} \longleftarrow \mathbf{w}^{(k)} - \alpha\, \mathbf{clip}\left(\nabla_{\mathbf{w}} \mathbf{Loss}(\mathbf{w}^{(k)}, \mathbf{X}, \mathbf{y})\right)$$

Gradient clipping

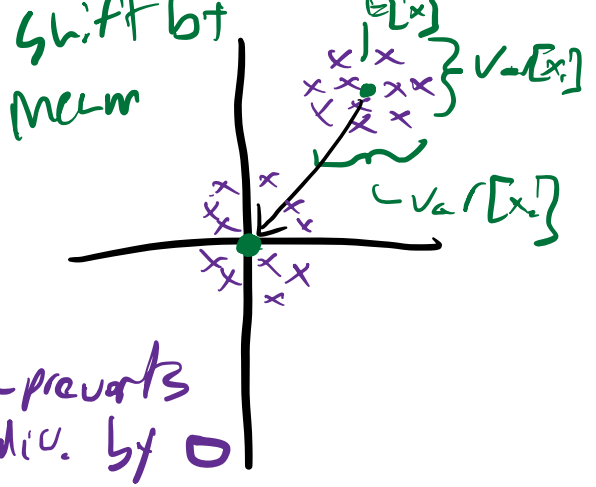


Normalization

Batch normalization

Normalize over the batch:

$$\text{BatchNorm}(x) = \frac{x - \mathbb{E}[x]}{\sqrt{\text{Var}[x] + \epsilon}}$$

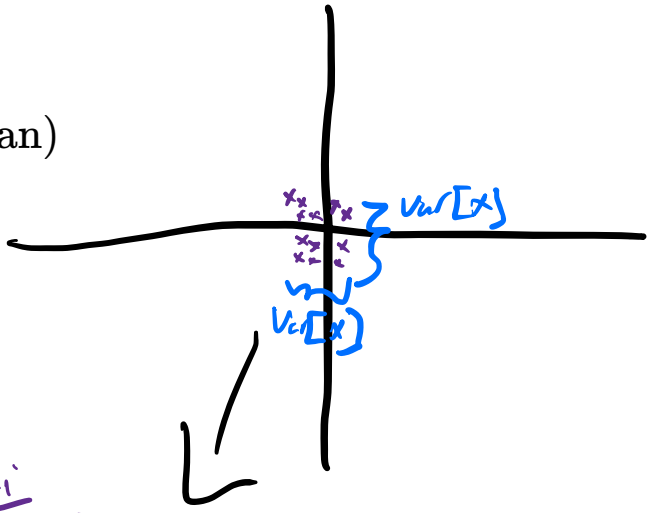


Training time:

$$\text{Batch: } \{x_1, x_2, \ldots, x_B\}$$

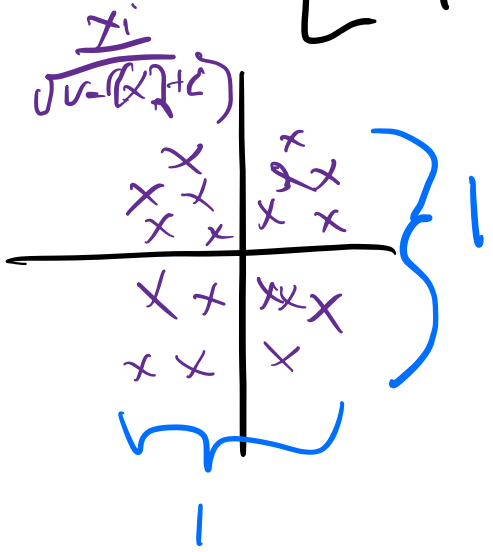
$$\mathbb{E}[x] \approx \bar{x} = \frac{1}{B}\sum_{i=1}^{B} x_i \quad \text{(sample mean)}$$

our current estimate of the mean [E[x]]

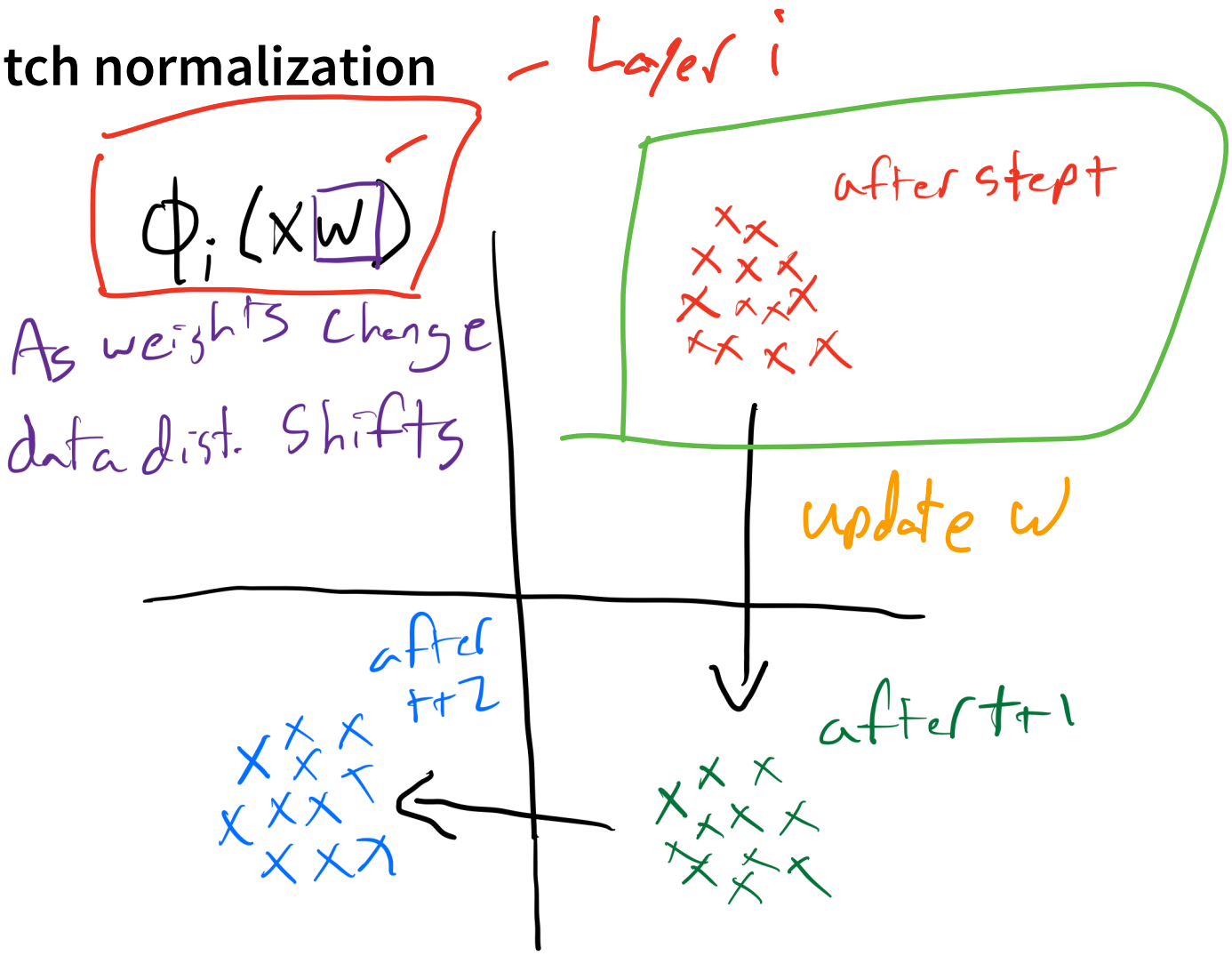


Note

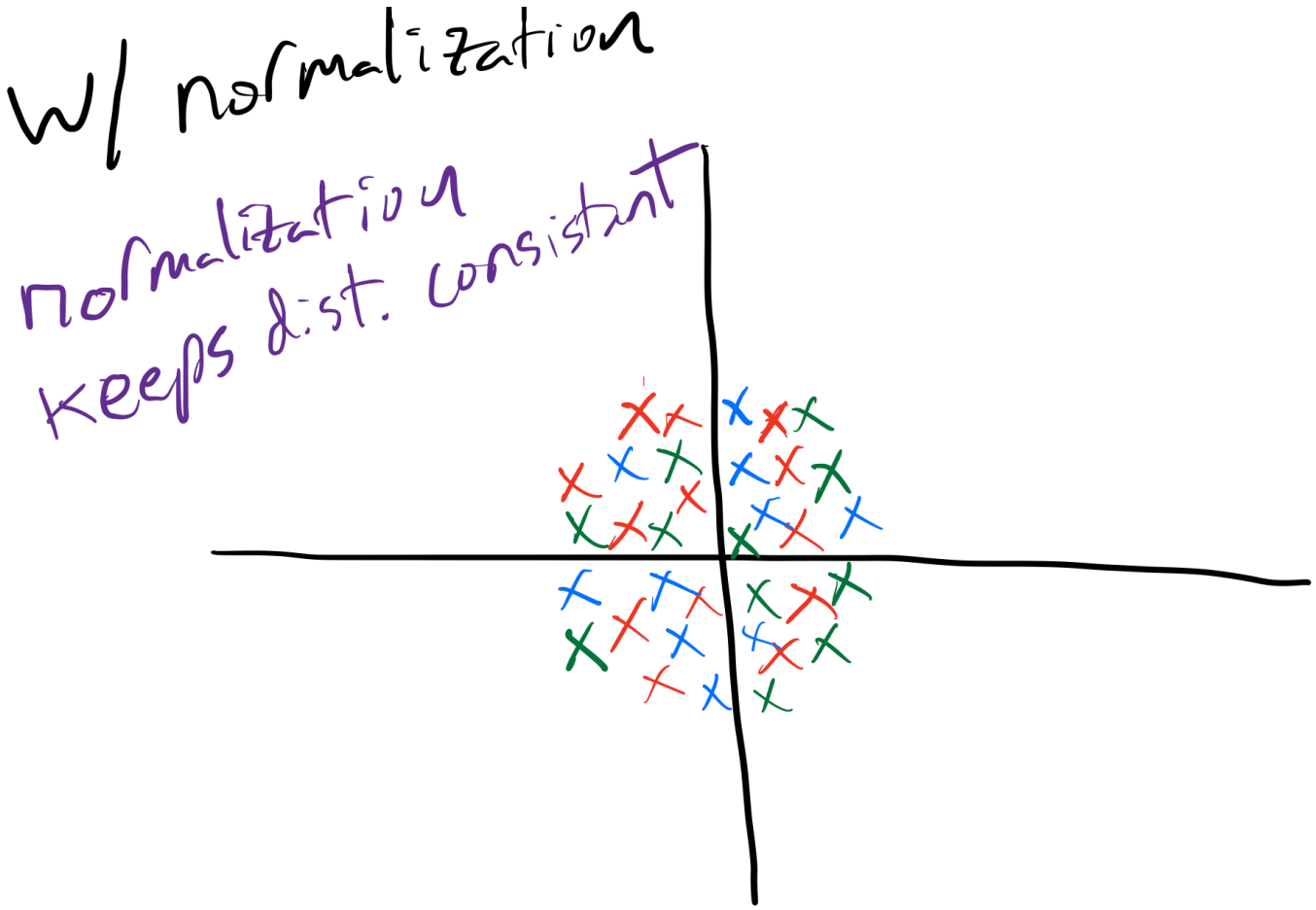
Could also use full dataset



Batch normalization



Batch normalization



Batch normalization

Biased estimator:

Sample mean

$$\mathrm{Var}[x] \approx s^2 = \frac{1}{B}\sum_{i=1}^{B}\left(x_i - \left(\frac{1}{B}\sum_{i=1}^{B} x_i\right)\right)^2 \quad \text{(sample var.)}$$

Unbiased estimator:

$$\mathrm{Var}[x] \approx s^2 = \frac{1}{B-1}\sum_{i=1}^{B}\left(x_i - \left(\frac{1}{B}\sum_{i=1}^{B} x_i\right)\right)^2 \quad \text{(sample var.)}$$

$$\underset{\text{train}}{\mathrm{BatchNorm}}(x) = \frac{x - \bar{x}}{\sqrt{s^2 + \epsilon}}$$

Batch normalization

Running estimate:

$$\bar{\mu}^{(k+1)} \longleftarrow \beta\bar{\mu}^{(k)} + (1-\beta)\bar{x}^{(k)}$$

$$\bar{\sigma}^{2(k+1)} \longleftarrow \beta\bar{\sigma}^{2(k)} + (1-\beta)s^{2(k)}$$

$$\underset{\text{test}}{\text{BatchNorm}}(x) = \frac{x - \bar{\mu}}{\sqrt{\bar{\sigma}^2 + \epsilon}}$$

Layer normalization

Normalize over the layer:

$$\text{LayerNorm}(\mathbf{x}) = \frac{\mathbf{x} - \bar{x}}{\sqrt{s^2 + \epsilon}}, \quad \mathbf{x} = \begin{bmatrix} x_1 \\ \vdots \\ x_d \end{bmatrix}$$

Training & test time:

$$\bar{x} = \frac{1}{d} \sum_{i=1}^{d} x_i \quad \text{(output mean)}$$

Layer normalization

Biased estimator:

$$s^2 = \frac{1}{d} \sum_{i=1}^{d} \left(x_i - \left(\frac{1}{d} \sum_{i=1}^{d} x_i \right) \right)^2 \quad \text{(output var.)}$$

Unbiased estimator:

$$s^2 = \frac{1}{d-1} \sum_{i=1}^{d} \left(x_i - \left(\frac{1}{d} \sum_{i=1}^{d} x_i \right) \right)^2 \quad \text{(output var.)}$$

Scaled normalization

$$\text{BatchNorm}(x) = \frac{x - \mathbb{E}[x]}{\sqrt{\text{Var}[x] + \epsilon}}\gamma + \kappa$$

$$\text{LayerNorm}(\mathbf{x}) = \frac{\mathbf{x} - \bar{x}}{\sqrt{s^2 + \epsilon}}\gamma + \kappa$$