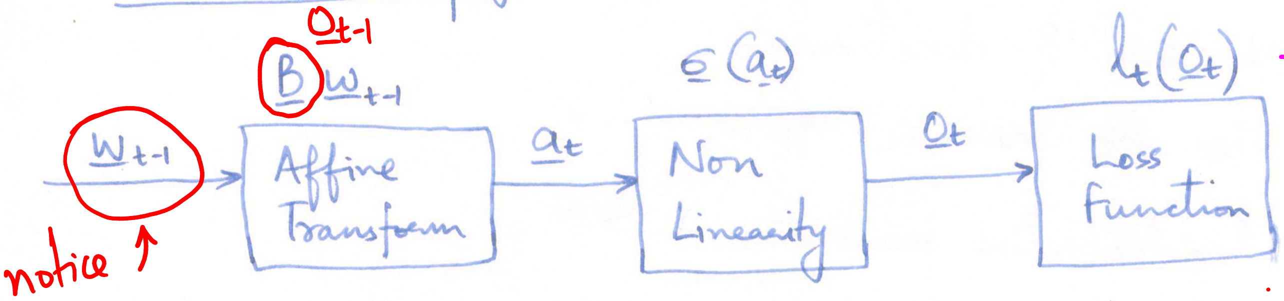
Forward Propagation



W_{t-1}

$k_{t+1} \times k_t$

$w_{t+1} =$

$k_{t+1} k_t \times 1$

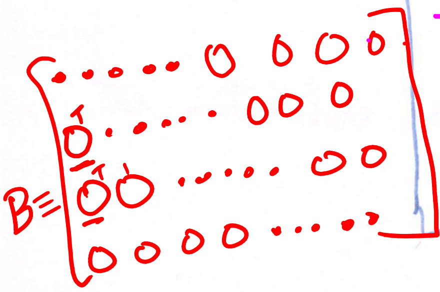
Matrix $\underline{B}$ is formulated using layer inputs O_{t-1}

Activation $a_t = \underbrace{A\, O_{t-1}}_{W_{t-1} O_{t-1}} = \underbrace{B\, w_{t-1}}_{B\, w_{t-1}}$ → vector w_{t-1}

vector w_{t-1} is the column vector obtained by concatenating the rows of matrix W_{t-1} and then taking the transpose of the resulting long row vector. $k_{t+1} k_t$.

zeros

$$B \equiv O_{t-1} = k_t \begin{bmatrix} O_{t-1}^T & 0^z & \cdots & 0^z \\ 0^z & O_{t-1}^T & \cdots & 0^z \\ \vdots & & & \\ 0^z & 0^z & \cdots & O_{t-1}^T \end{bmatrix}$$



$$W_{t-1} \qquad w_{t+1} = \begin{bmatrix} \text{row1} \\ \text{row2} \\ \vdots \end{bmatrix} \quad k_{t+1} k_t$$

Layer $t-1$ $\xrightarrow{W_{t-1}}$ layer t

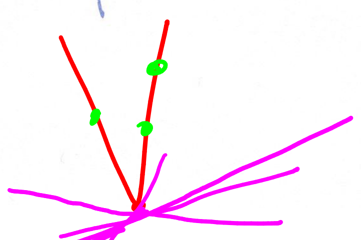
If k_t denotes the number of neurons in the layer V_t

then size of O_{t-1} is $k_t \times (k_{t-1} k_t)$

vector w_{t-1} is $(k_{t-1} k_t) \times 1$.

Training a Neural Network : Stochastic Gradient Descent (SGD).

Inputs to the SGD:
- Training examples $(\underline{x}, y)$
- Layered Graph (V, E)
- A differentiable non-linearity σ



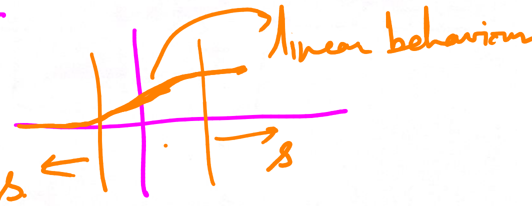
H. Hyper Parameters:

Number of iterations τ

Step size sequence $\eta_1\ \eta_2 \dots \eta_\tau$

Regularization parameter $\lambda > 0$

$0 = A\, \underline{o}_{t-1}$



Initialize:

Choose $\underline{w}^{(1)} \in \mathbb{R}^{|E|}$ at random from a distribution such that $\underline{w}^{(1)}$ is close to $\underline{0}$ zerovector.

edges

small weights ensure small activations.

for $i = 1, 2, \dots \tau$

sample $(\underline{x}, \underline{y}) \sim \mathcal{D}$

calculate gradient $\underline{v}_i$ = backpropagation $(\underline{x}, \underline{y}, \underline{w}, (V,E), \sigma)$????

update $\underline{w}^{(i+1)} = \underline{w}^{(i)} - \eta_i \left(\underline{v}_i + \lambda \underline{w}^{(i)} \right)$ $\frac{1}{2}\|\underline{w}\|^2$

SGD update step.

Output:

$\underline{w}^*$ is the best performing $\underline{w}^{(i)}$ on the validation set

---

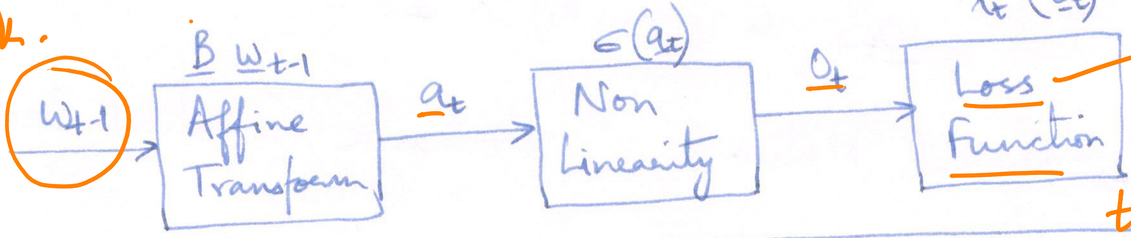
Computing the Gradients

Backpropagation

no space

back ~~propagation~~

think.



is modeled by the subsequent net layers $t+1$ onwards

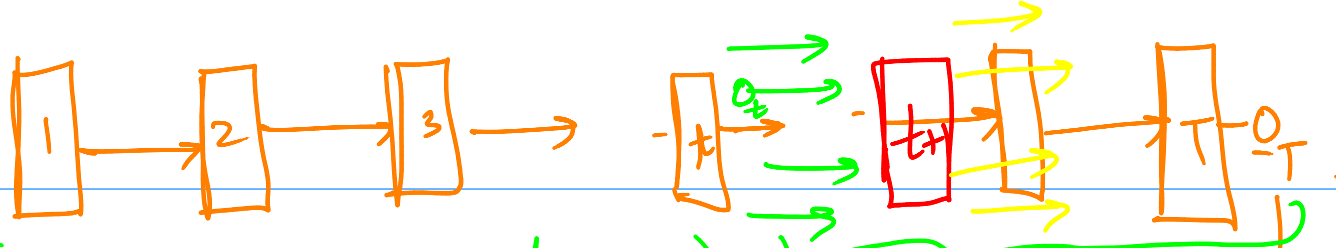
Composite function $g_t = (\text{Affine} \circ \text{Nonlinearity} \circ \text{Loss})$

$g_t(w_{t-1})$

nested function

$h(g(f(x)))$

$f \circ g \circ h$ composition of function



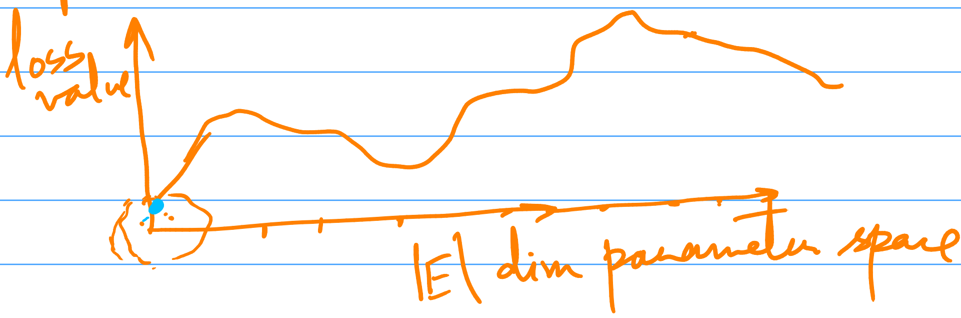
$$l_t(\underline{O}_t) = l_{t+1}(\underline{O}_{t+1}) = l_{t+2}(\underline{O}_{t+2}) = l_T(\underline{O}_T)$$

$l_t(\underline{O}_t)$ loss

$l_{t+1}(\underline{O}_{t+1})$

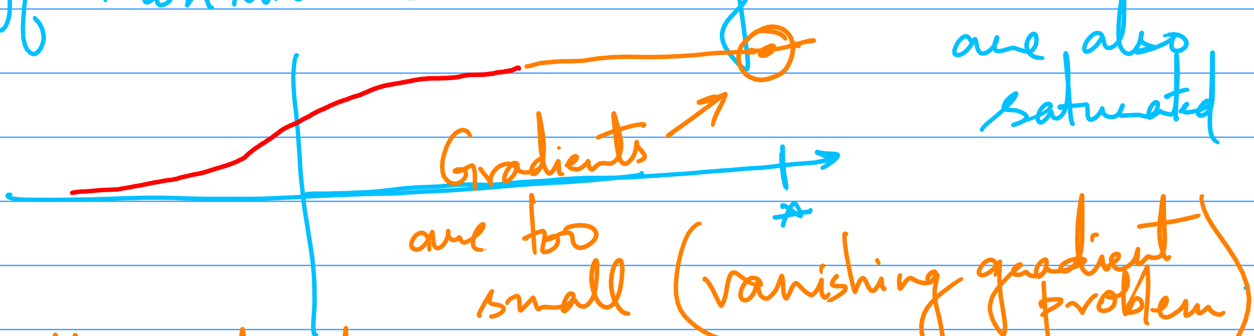
Parameter space: $|E|$

loss



large $\underline{w}$ ⇒ large values for activations.

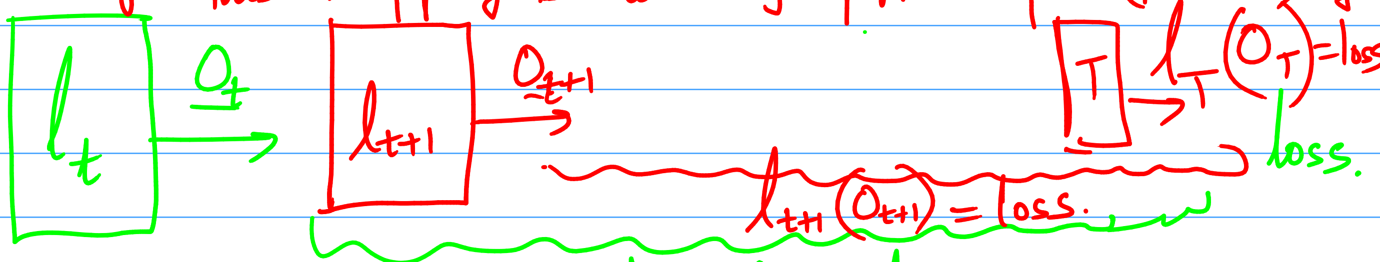
If non lin are saturating → out values are also saturated



small gradients ⇒ small updates to $\underline{w}$ ⇒ slow learning.

outputs of each layer can be mapped to the loss val

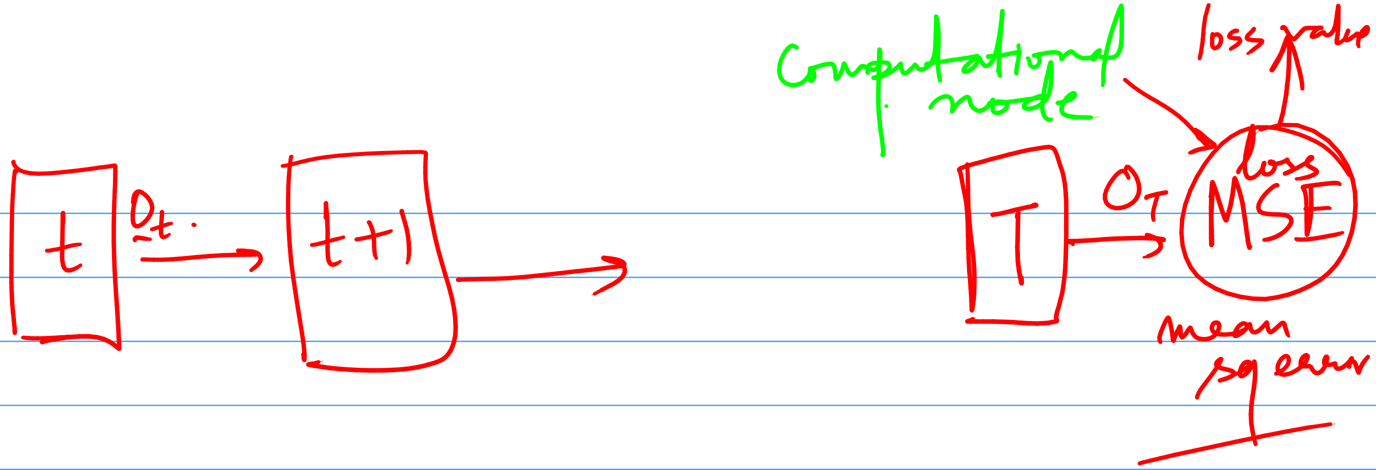
This mapping is done by the subsequent (further) layers



$l_{t+1}(O_{t+1}) = \text{loss}.$

$l_t(\underline{O}_t) = \text{loss}.$

$$\text{loss} = l_T(\underline{O}_T) = \cdots = \underline{l_{t+1}(\underline{O}_{t+1})} = \underline{l_t}(\underline{O}_t)$$



Cross Entropy.

$$loss = MSE(O_T)$$

$$MSE(O_t)$$

$$MSE(\text{layers}_{t+1 \text{ onwards}} \text{ that process } O_t \text{ \& map it to } O_T)$$

$$MSE(\text{Subsequent layers after layer } t\,(O_t))$$

$$loss = MSE(l_t(O_t))$$

$$= MSE(l_{t+1}(O_{t+1}))$$

$$= \underbrace{MSE(O_T)}_{(y_T - O_T)^2}$$