03/08/2023

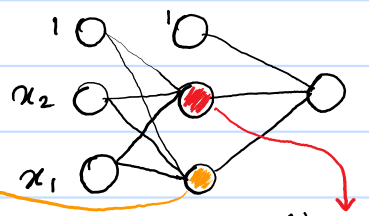
RNN ⟶ Sequential

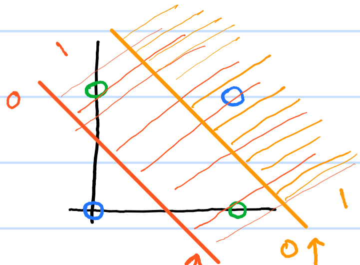
CNN

Hyper parameter ⟶ Higher level parameters of the model eg. learning rate in Perceptron Learning Algorithm

There are various kinds of hyper parameters.

2-layers neural network :



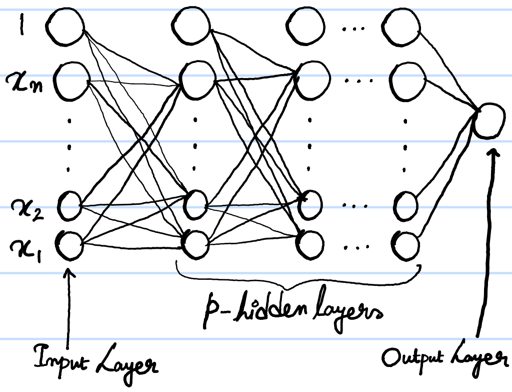


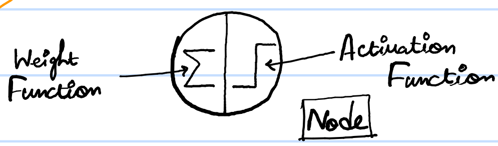
$$\omega_{11}^{(1)} x_1 + \omega_{21}^{(1)} x_2 + \omega_{31}^{(1)} = 0$$

$$\Rightarrow x_1 + x_2 - \frac{1}{2} = 0 \Rightarrow \frac{x_1}{1/2} + \frac{x_2}{1/2} = 1$$

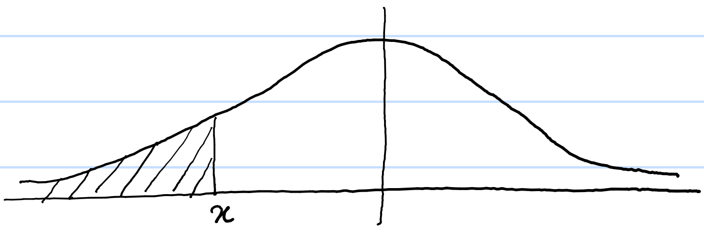
$$\omega_{12}^{(1)} x_1 + \omega_{22}^{(1)} x_2 + \omega_{32}^{(1)} = 0$$

$$\Rightarrow x_1 + x_2 - \frac{3}{2} = 0 \Rightarrow \frac{x_1}{3/2} + \frac{x_2}{3/2}$$





05/08/2023

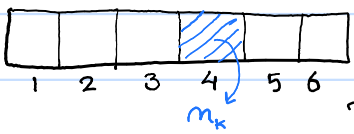


$$f(x) = \omega_1 x + \omega_2 x^2 + \ldots + \omega_{10} x^{10}$$

ω_{10} has more significance.

10/08/2023

K-Fold :

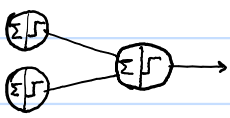


Then we find E_K.

$$\sigma(z)(1-\sigma(z))$$

$$\frac{\partial L}{\partial w_{j1}^{(2)}} = \frac{\partial\left(-y \log(y^*) - (1-y)\log(1-y^*)\right)}{\partial w_{j1}^{(2)}} = \frac{\partial L}{\partial y^*} \cdot \frac{\partial y^*}{\partial w_{j1}^{(2)}}$$

$$= \left(-\frac{y}{y^*} + \frac{(1-y)}{1-y^*}\right) \cdot \frac{\partial A\left(\sum_{j=1}^{2} w_{j1}^{(2)} z_j\right)}{\partial w_{j1}^{(2)}}$$



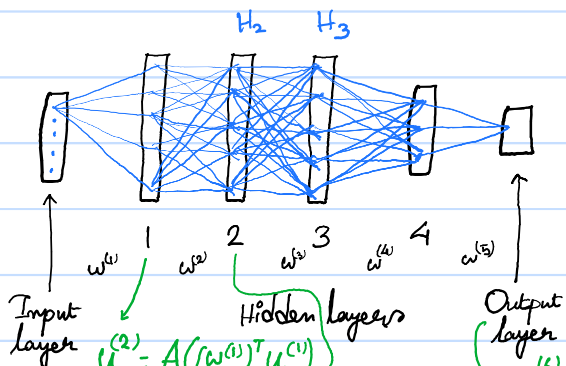
$$\frac{\partial}{\partial x} f\left(a_1 h(y) + a_2 f(x)\right) = f'(\,)\, a_2 f'(x)$$

4 hidden layer :

2 layer ⟶ 3 layer

H_2 $\quad$ H_3

$$\begin{array}{cc} & H_3 \\ H_2 & \begin{bmatrix} W_{11}^{(3)} & \cdots & W_{1H_3}^{(3)} \\ \vdots & & \vdots \\ W_{H_2 1}^{(3)} & \cdots & W_{H_2 H_3}^{(3)} \end{bmatrix} \end{array}$$



$$u^{(2)} = A\left((\omega^{(1)})^T u^{(1)}\right)$$

$$u^{(3)} = A\left((\omega^{(2)})^T u^{(2)}\right)$$

$$u^{(6)} = A\left((\omega^{(5)})^T u^{(5)}\right)$$

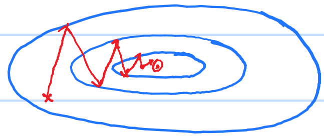
12/08/2023

Computational Resource needed ⟶ ① Storage, ② Time

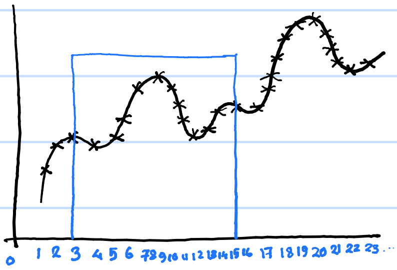
Mini Batch Gradient Decent ⟶ Many updates for a batch at a time.

Batch Gradient Decent ⟶ Updates are made after processing the whole batch of data.

We can't reach any global minima in Deep Learning in Reality.
The problems of saddle point (Since geometry is complex).



Gradient Decent



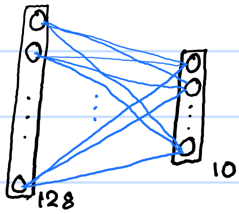
14/08/2023

$$W^{(t+1)} = W^{(t)} - \xi \nabla_{W^{(t)}} \mathcal{L}(y^*(\omega), y)$$ ⟵ Updating the weight for Gradient Decent

Tensorflow

$$\begin{bmatrix} \cdot & \cdot & \cdots & \vdots \\ \vdots & \vdots & & \vdots \\ \cdot & \cdot & \cdots & \cdot \end{bmatrix}_{28\times 28} = [\;\cdots\;]_{784}$$

flattening



17/08/2023

PyTorch (PyTorch FMNIST).

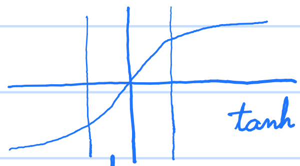
$$z_1 = A(W_{11}^{(1)}X_1 + W_{21}^{(1)}X_2) = A(W_0X_1 + W_0X_2)$$

$$z_2 = A(W_{21}^{(1)}X_1 + W_{22}^{(1)}X_2) = A(W_0X_1 + W_0X_2) = z_1$$

$$y^* = A(W_{11}^{(2)}z_1 + W_{21}^{(2)}z_2) = A(W_0z_1 + W_0z_2)$$

$$\frac{\partial y^*}{\partial W_{11}^{(2)}} = A'(\cdot)\, z_1 \quad \text{and} \quad \frac{\partial y^*}{\partial W_{21}^{(2)}} = A'(\cdot)\, z_2$$

$$z_1 = A(W_0X_1 + W_0X_2) \Rightarrow \frac{\partial z_1}{\partial W_{11}^{(1)}} = A'(\cdot)\, X_1$$



tanh(x)

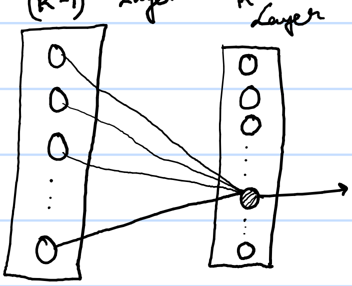
Almost Linear

$$z_1 = W^{(1)}X$$
$$z_2 = W^{(2)}z_1$$
$$\vdots$$
$$y^* = z_n = W^{(n)}W^{(n-1)}\cdots W^{(1)}X$$

for large $W^{(i)}$, $y^* \to \infty/-\infty$

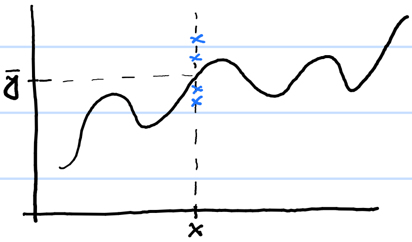
for small $W^{(i)}$, $y^* \to 0$.

Exploding Gradient or Vanishing Gradient Problem

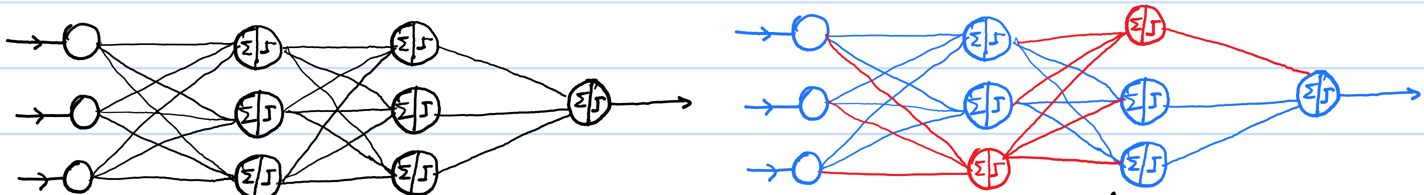


$$z^{(K)} = [z_1^{(K)}\; z_2^{(K)} \;\cdots\; z_j^{(K)} \cdots\; z_{H_K}^{(K)}]$$

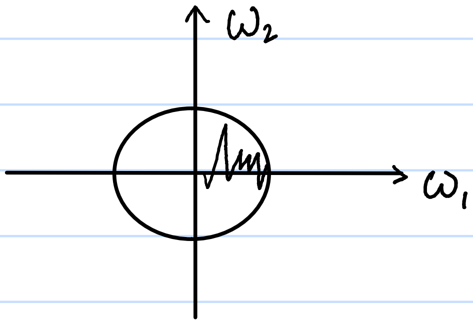
21/08/2023

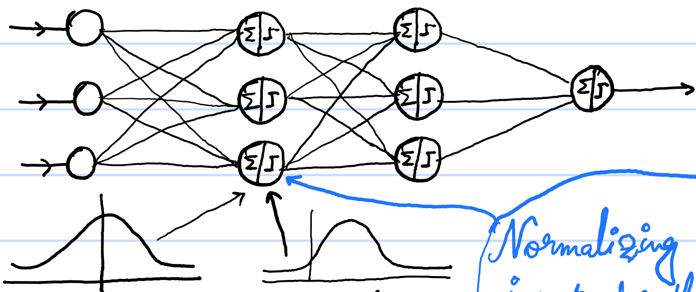


Dropout:



Red one's are the dropouts. The renewed model is forced to learn from dataset.





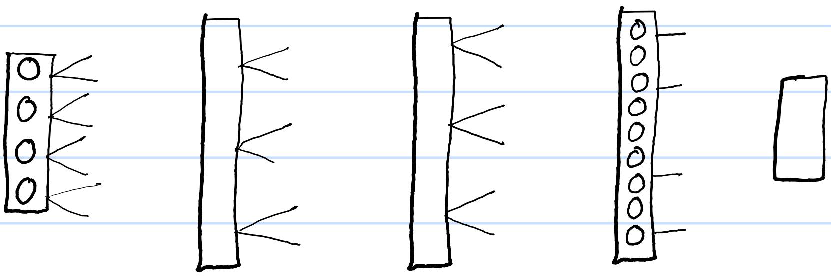
Sample 1 Sample 2

Normalizing the inputs for this layer and get output

24/08/2023

Batch Normalization:

W

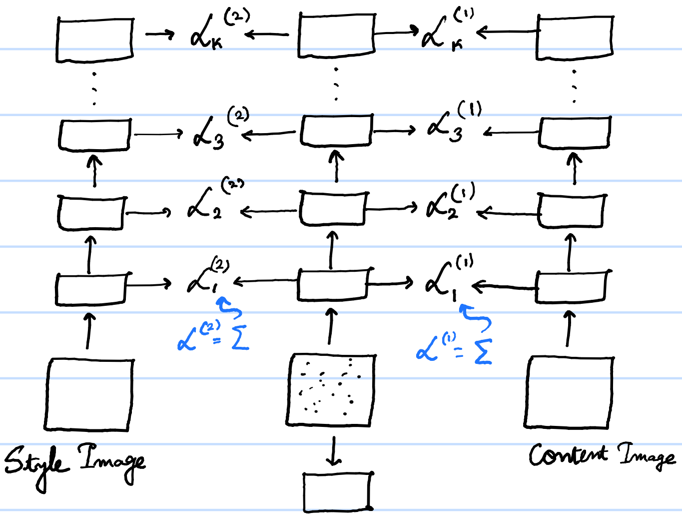


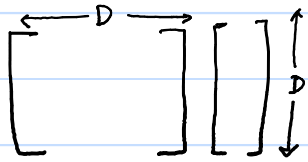
Exploding & Vanishing Gradient Solution:

1. Change Activation Function
2. Regularization.

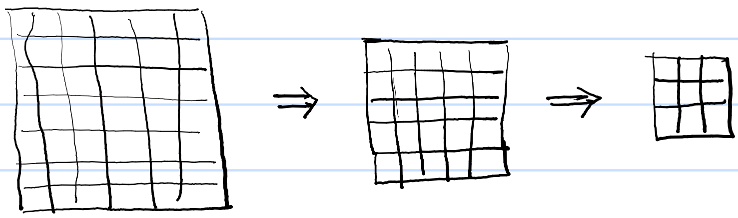
Convolutional Neural Network :

28/08/2023





$WX \to D$

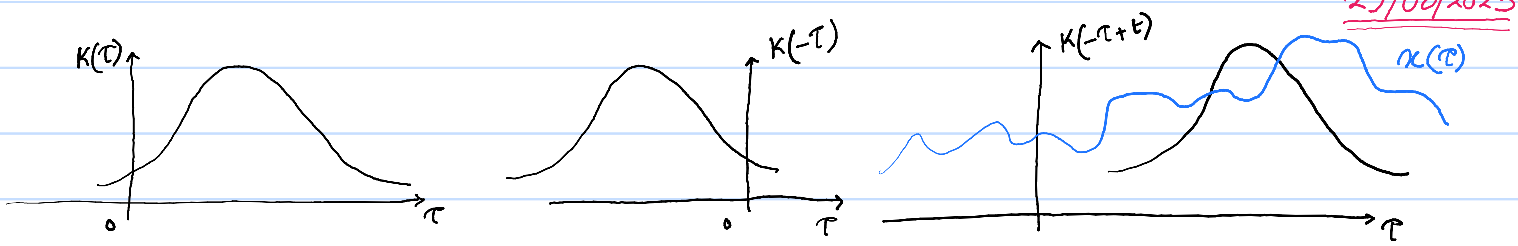


Padding and Pooling

$30 \times 30 \times 32$, $32 \times 3 \times 3 \times 3 = 288 \times 3 = 864$ } After this, $864 + 32 = 896$

$13 \times 13 \times 64$, $64 \times (3 \times 3) \times 32 + 64 = 1849$

29/08/2023

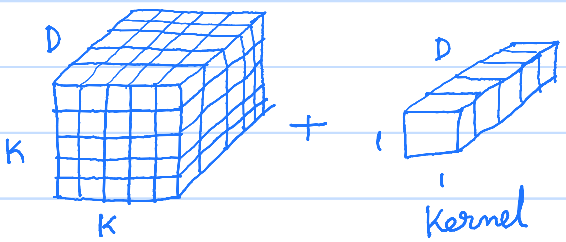


$$g(t) = \int x(\tau)\, K(t-\tau)\, d\tau = \int K(T)\, x(t-T)\, dT \quad \left[\text{Taking } t-\tau = T\right]$$

$$= \int K(\tau)\, x(t-\tau)\, d\tau \iff (K * x)(t) = (x * K)(t)$$

Again, $\sum_{\alpha} \sum_{\beta} x(\alpha, \beta)\, K(i-\alpha, j-\beta) = \sum_{\alpha} \sum_{\beta} x(i-\alpha, j-\beta)\, K(\alpha, \beta)$

But we have, $\sum_{\alpha} \sum_{\beta} x(i+\alpha, j+\beta) \cdot K(\alpha, \beta) \longleftarrow$ Cross Correlation



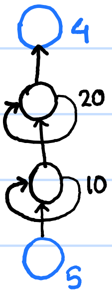
$\Rightarrow \boxed{K \times K \times 1}$

Again, for 5 $(1 \times 1 \times D)$ filter, if $D = 10$ we have, $(K \times K \times 5)$

RNN:

09/09/2023

$h_t = f(x_t, h_{t-1})$, Note that, $h_1 = f(x_1, h_0)$ then we have to put a value to h_0 such that we get a better result and faster output.



What is the dimension of $\omega_1, \omega_{11}, \omega_2$?

$X_t^{(n)} = 5 \times 1$

$\omega_1 = 10 \times 5$

$\omega_{11} = 10 \times 10$

$\omega_2 = 20 \times 10$

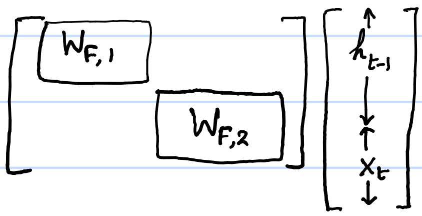
11/09/2023

For RNN we have sequential inputs and get an output per time period.

$$h_{T,i}^{(n)} = A(z_{1,T,i}^{(n)}), \quad z_{1,T,j}^{(n)} = \sum_i \omega_{1,ij} x_{T,i} + \sum_i \omega_{11,ij} h_{T-1,j} + b_j, \quad \frac{\partial z_{1,T,i}^{(n)}}{\partial \omega_{1,ij}} = x_{T,i}$$

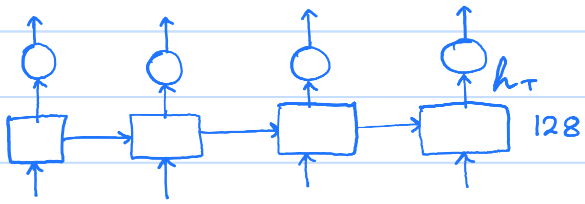
$$\frac{\partial z_{1,T,j}^{(n)}}{\partial \omega_{11,ij}} = h_{T-1,i}^{(n)}$$

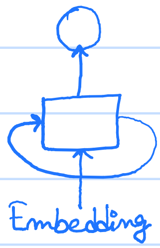
$$W_{F,1} h_{t-1} + W_{F,2} X_t + b$$

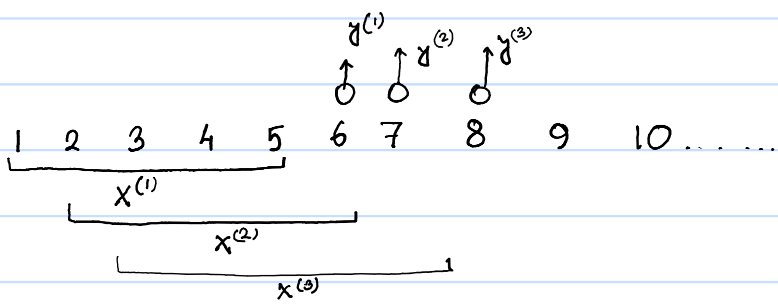


14/09/2023

Layers	Output Shape	Parameters
Embedding	(None, None, 64)	$64 \times 1000 = 64000$
LSTM	(None, 128)	$\underset{128\times64}{W_I}\, \underset{64}{X} + \underset{128\times128}{W_R}\, \underset{128}{R_{t-1}} + \underset{128}{b} = 24704\ (\times 4)$ (∵ there are 4 networks) $= 98816$







GRU → Return Sequence

LSTM → Return State

Layers

→ Bidirectional (LSTM (10))

→ Conv 1D (TCN)

→ Time Distributed

→ Time 2 Vec