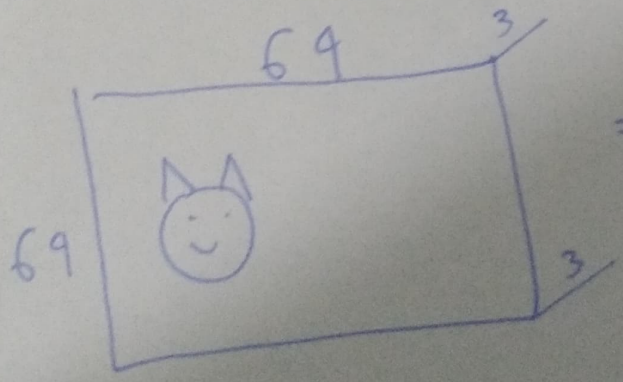
Lecture-10

Deep Learning:-

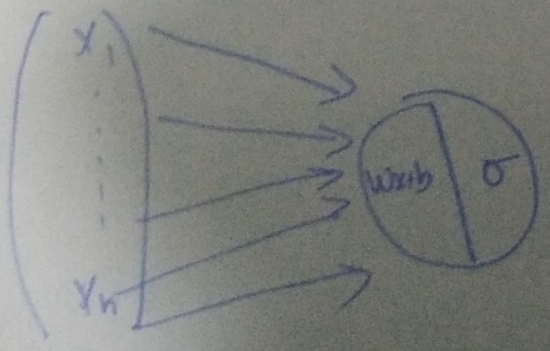
- Computational power
- Data
- Algorithms

* How can logistic regression be seen as neural network?

Goal: Find cats in images (1 → presence of cats, 0 → absence of cats)



$= \begin{pmatrix} x_1 \\ \vdots \\ x_n \end{pmatrix}$ → flattened the image.



$\rightarrow \hat{y} = \sigma(\theta^T x)$

$= \sigma(wx + b)$

$(1, 12288)$ $(12288, 1)$

1) Initialize w, b
2) Find the optimal w, b
3) Use $\hat{y} = \sigma(wx + b)$ to predict

(*) Define a loss function:

$$\min \mathcal{L} = -\left[y \log \hat{y} + (1-y) \log (1-\hat{y}) \right]$$

$$w = w - \alpha \frac{\partial \mathcal{L}}{\partial w}$$

$$b = b - \alpha \frac{\partial \mathcal{L}}{\partial b}$$

→ Gradient descent.

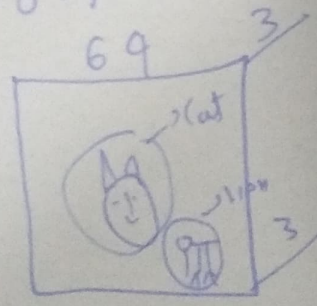
(*) Neuron = Linear + Activation

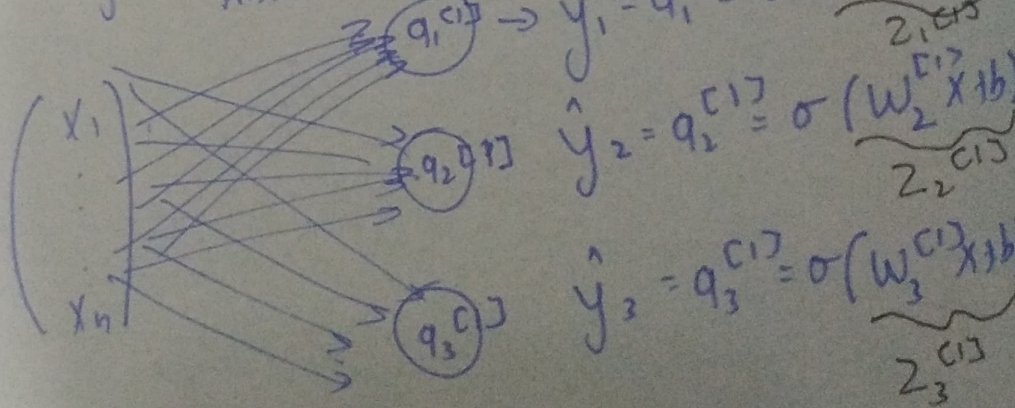
(*) Model = Architecture + Parameters

(*) New goal: Find cat/lion/iguana in images. (Multiple presence)

[ ] → Layer.

Label: $\begin{pmatrix} 1 \\ 1 \\ 0 \end{pmatrix}$ (cat, lion, iguana)

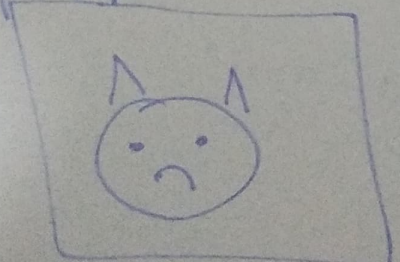


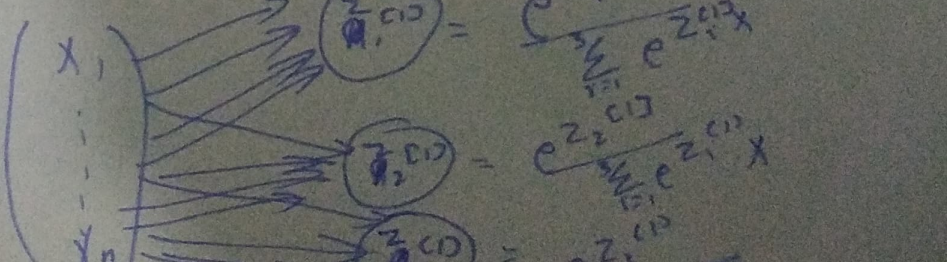


(*) New goal: Add contraint unique animal in an image

Softmax Layer.

Label: $\begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix}$ (cat, lion, iguana)





⑤ Redesign loss function for Goal 2.0 :-

$$\mathcal{L} = -\sum_{k=1}^{2} [\, y_k \log \hat{y}_k + (1 - y_k) \log (1 - \hat{y}_k) \,]$$

⑥ Redesign loss function for Goal 3.0 :-

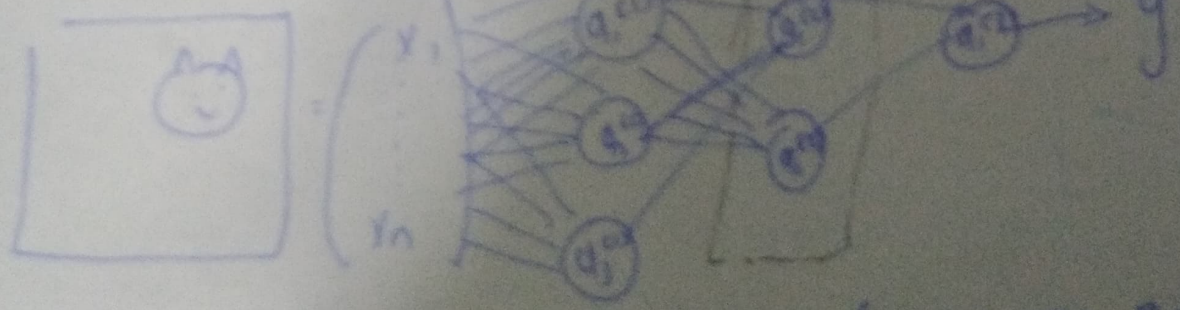
(Cross-Entropy loss)

$$\mathcal{L} = -\sum_{k=1}^{3} y_k \log \hat{y}_k$$

II Neural Networks :-

Goal :- Cat vs No cat

(1) (0)



→ Layer is a cluster of neuron not connected to each other.

→ ↳ Hidden : Output and Input are hidden from this layer.

(*)

~~First Layer are going~~

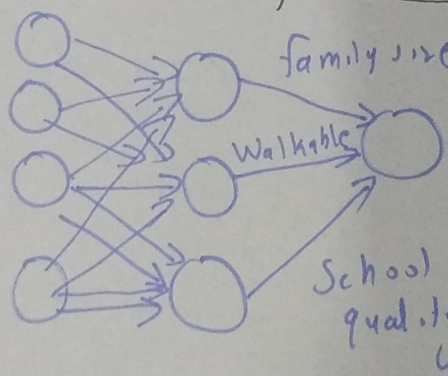
(*) Layers tend to learn different feature representations (complex when combined) which help in the final classification task

↳ Deeper layer learn more complex features.

(*) House price prediction:-

#1

bedroom
Size
Zip code
Wealth



Hidden feature which had to be hand encoded in machine learning but in deep learning can be learned if give enough data.

↳ In practice we use fully connected layer which means we connect every input of a layer to every output of ~~first~~ fowrid layer.

↳ End-to-End learning:-

↳ Input → Layer → Ground Truth

Some Black Box of learned features.

⊙ Matrix = Shape

Foward Propagation equations:-

$$z^{[1]} = w^{[1]} \cdot X + b^{[1]} \rightarrow$$

$w^{[1]} = (3, n)$.
$X = (n, 1)$
$z^{[1]} = (3, 1)$.
$a^{[1]} = (3, 1)$.

$b^{[1]} = (3,1)$

$$a^{[1]} = \sigma(z^{[1]}).$$

$b^{[2]} = (2,1)$

$$z^{[2]} = w^{[2]} \cdot a^{[1]} + b^{[2]}$$

$w^{[2]} = (2, 3)$
$z^{[2]} = (2, 1)$
$a^{[2]} = (2, 1)$

$b^{[3]} = (1,1)$

$$a^{[2]} = \sigma(z^{[2]}).$$

$w^{[3]} = (1, 2)$

$$z^{[3]} = w^{[3]} \cdot a^{[2]} + b^{[3]}$$

$a^{[3]} = (1, 1)$

$$a^{[3]} = \sigma(z^{[3]}).$$

→ We put everything into matrices.

In general

$$w^{[l]} = (a^{[l]}\text{ shape}, a^{[l-1]}\text{ shape})$$

$$z^{[l]} = (\#\text{ of neurons in layer}, m)$$

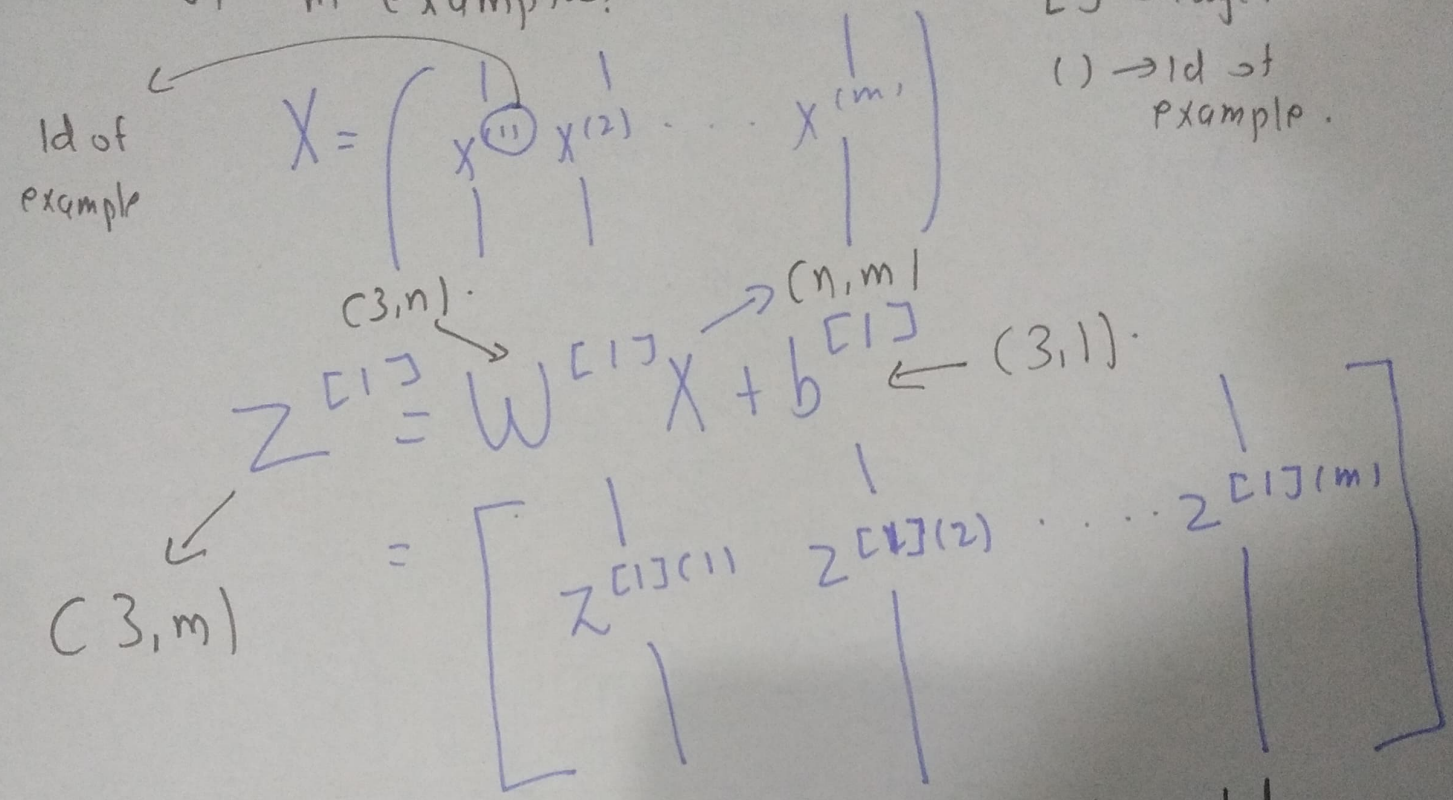
$$a^{[l]} = (\#\text{ of neurons in layer}, m)$$

$$b^{[l]} = (a^{[l]}\text{ shape}, 1)$$

where m is size of batch.

(*) What happens for an input batch of m examples?

[] → layer
() → id of example.



↳ (*) Even though for b to add to add to $W^{[1]}X$ which is of shape (3,m), b has to be also (3,m) but we were using (3,1). The reason is that we will use broadcasting. It is a fact that parameter remain same but operation is parallelized so b would automatically through broadcasting would become (3,m) where every every row value has been repeated m times.

Optimizing $W^{[1]}, W^{[2]}, W^{[3]}, b^{[1]}, b^{[2]}, b^{[3]}$:-

(Continuing same example)

Define loss / Cost function
- ↙ single example
- ↘ m examples

$$J(\hat{y}, y) = \frac{1}{m} \sum_{i=1}^{m} \mathcal{L}^{(i)}$$ → Batch gradient descent.

with $\mathcal{L}^{(i)} = -[y^{(i)} \log \hat{y}^{(i)} + (1-y^{(i)}) \log (1-\hat{y}^{(i)})]$

Backward Propagation

$\forall\, l = 1 \ldots 3$

$$W^{[l]} = W^{[l]} - \alpha \frac{\partial J}{\partial W^{[l]}}$$ ← We have to compute this.

$$b^{[l]} = b^{[l]} - \alpha \frac{\partial J}{\partial b^{[l]}}$$

→ We will start with later layers now: and then use chain rule

$\hat{y} = a^{[3]}$ ← Will go into detail in next lecture

① $$\frac{\partial J}{\partial W^{[3]}} = \frac{\partial J}{\partial \hat{y}} \cdot \frac{\partial \hat{y}}{\partial z^{[3]}} \cdot \frac{\partial z^{[3]}}{\partial W^{[3]}}$$

② $$\frac{\partial J}{\partial W^{[2]}} = \frac{\partial J}{\partial z^{[3]}} \frac{\partial z^{[3]}}{\partial a^{[2]}} \frac{\partial a^{[2]}}{\partial z^{[2]}} \cdot \frac{\partial z^{[2]}}{\partial W^{[2]}}$$

↳ ~~Later~~ Early layers with chain rule

(3) $$\frac{\partial J}{\partial W^{[1]}} = \boxed{\frac{\partial J}{\partial z^{[2]}}} \cdot \frac{\partial z^{[2]}}{\partial a^{[1]}} \cdot \frac{\partial a^{[1]}}{\partial z^{[1]}} \cdot \frac{\partial z^{[1]}}{\partial W^{[1]}}$$

(132) → from $\frac{\partial J}{\partial w^{[2]}}$