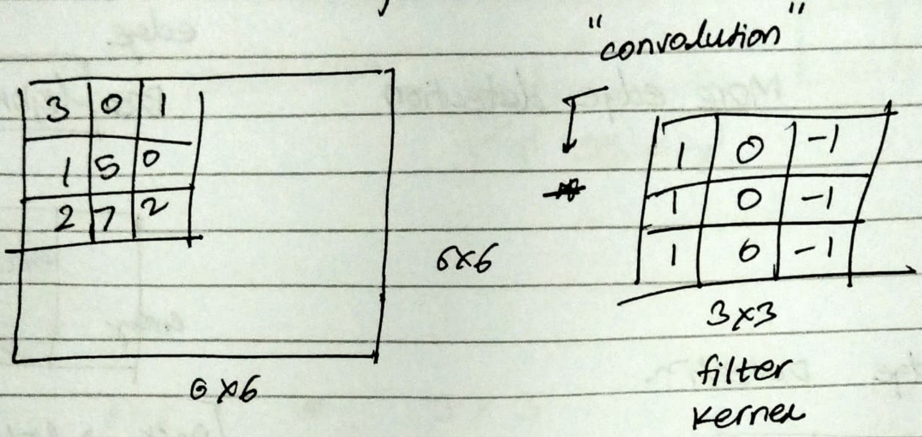
CONVOLUTIONAL NEURAL NETWORKS ·

Computer Vision Problems
→ Image ~~to~~classification
→ Object detection
→ Neural Style transfer

Cant take huge images as input as it requires very high end computations so lets ~~lea~~ learn convolution

* How to detect edges?



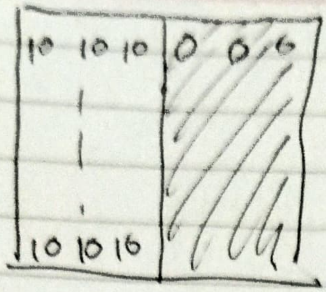
=

-5	-4	0	8
-10	-2	2	3
0	-2	-4	-7
-3	-2	-3	-16

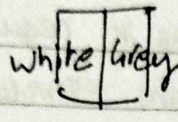
for box

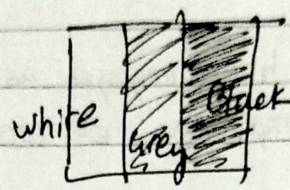
Multiply every element as per matrix and then add up these 9 numbers

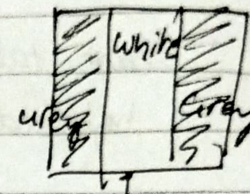
python : conv_forward
keras : conv2D



$$* \begin{bmatrix} 1 & 0 & -1 \\ 1 & 0 & -1 \\ 1 & 0 & -1 \end{bmatrix} = \begin{bmatrix} 0 & 30 & 30 & 0 \\ 0 & 30 & 30 & 0 \\ 0 & 30 & 30 & 0 \\ 0 & 30 & 30 & 0 \end{bmatrix}$$



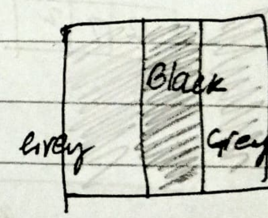




Detects one vertical edge

More edge detection

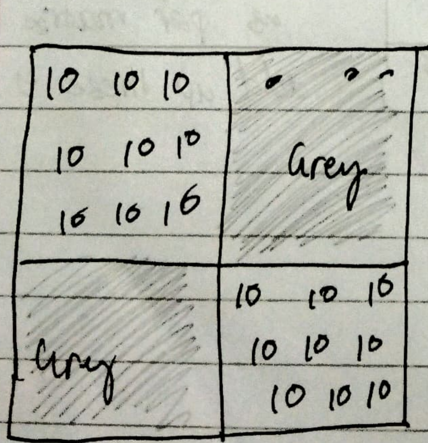
$\boxed{\text{light} \rightarrow \text{Dark}}$



$\boxed{\text{Dark} \rightarrow \text{light}}$

Horizontal Edge Detection

$$\begin{bmatrix} 1 & 1 & 1 \\ 0 & 0 & 0 \\ -1 & -1 & -1 \end{bmatrix}$$

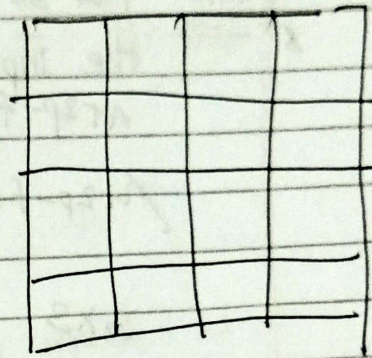


6x6

$$* \begin{bmatrix} 1 & 1 & 1 \\ 0 & 0 & 0 \\ -1 & -1 & -1 \end{bmatrix} = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 30 & 10 & -10 & -30 \\ 30 & 10 & -10 & -30 \\ 0 & 0 & 0 & 0 \end{bmatrix}$$

$$\begin{matrix} 1 & 0 & -1 \\ 1 & 0 & -1 \\ 1 & 0 & -1 \end{matrix} \qquad \begin{bmatrix} 1 & 0 & -1 \\ 2 & 0 & -2 \\ 1 & 0 & -1 \end{bmatrix} \qquad \begin{bmatrix} 3 & 0 & -3 \\ 10 & 0 & -10 \\ 3 & 0 & -3 \end{bmatrix}$$

Sobel filter   Scharr filter

6x6 Image $\ast$ $\begin{bmatrix} w_1 & w_2 & w_3 \\ w_4 & w_5 & w_6 \\ w_7 & w_8 & w_9 \end{bmatrix}$ = 

convolution

Backpropagation (find ~~using~~ using)

Padding

6x6 $\ast$ 3 x 3 = ~~6x~~ 4x4

n x n   f x f

8x8

6x6

$n-f+1 \ast n-f+1$

$6-3+1=4$

→ Shrinking output

→ throwing away info from edges

p = padding = 1

$n+2p-f+1 \ast n+2p-f+1$

$6+2-3+1 \times$ —— $= 6 \times 6$

Valid and same covolutions

"valid": $n \times n$ * $f \times f$ $\rightarrow$ $n-f+1 \times n-f+1$

6×6 * 3×3 $\rightarrow$ 4×4

Same: Pad so that output size is same as the input size.

$n+2p-f+1 \times n+2p-f+1$

$\cancel{n}+2p-f+1=\cancel{n} \Rightarrow p = \frac{f-1}{2}$

$3 \times 3 \quad p = \frac{3-1}{1} = 1$ | $5 \times 5 \quad p=2$, $f=5$

f is usually odd

1×1
3×3
5×5
7×7.

Strided convolutions

Taking big stride skipping over one row & one column

7×7 * $3 * 3$ = 3×3

$n \times n$ $\quad f \times f$

padding p stride s

$s=2$

$\left\lfloor \frac{n+2p-f}{s} + 1 \right\rfloor \times \left\lfloor \frac{n+2p-f}{s} + 1 \right\rfloor$

$\frac{7+0-3}{2} + 1 = \frac{4}{2} + 1 - 3$

$\lfloor 2 \rfloor$ = floor (2)

Ignore

~~nxn image~~

~~padding p~~ s'

cross - corelation vs convolution

we are using this from last few videos

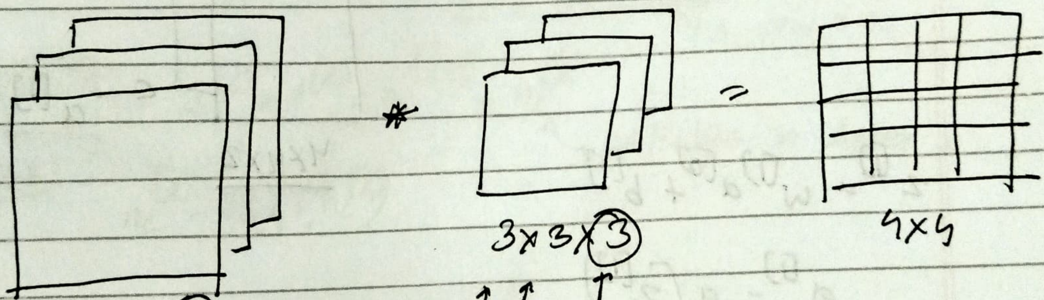
this is convolution and in deep learning so no issue.

$$6 \times 6 \quad * \quad \begin{bmatrix} 3 & 4 & 5 \\ 1 & 0 & 2 \\ -1 & 9 & 6 \end{bmatrix} \quad = \quad 4 \times 4$$

$$\begin{bmatrix} 7 & 9 & -1 \\ 2 & 0 & 1 \\ 5 & 4 & 3 \end{bmatrix}$$

Multiply elements of this flipped matrix

convolutions over volume



6x6x3 — height, width, chanels

3x3x3 — h, w, c

4x4

Just multiply & add in 3d

Multiple filters

6x6x3 * vertical edge 3x3x3 = 4x4

* horizontal edge 3x3x3 = 4x4

$\boxed{4 \times 4 \times 2}$

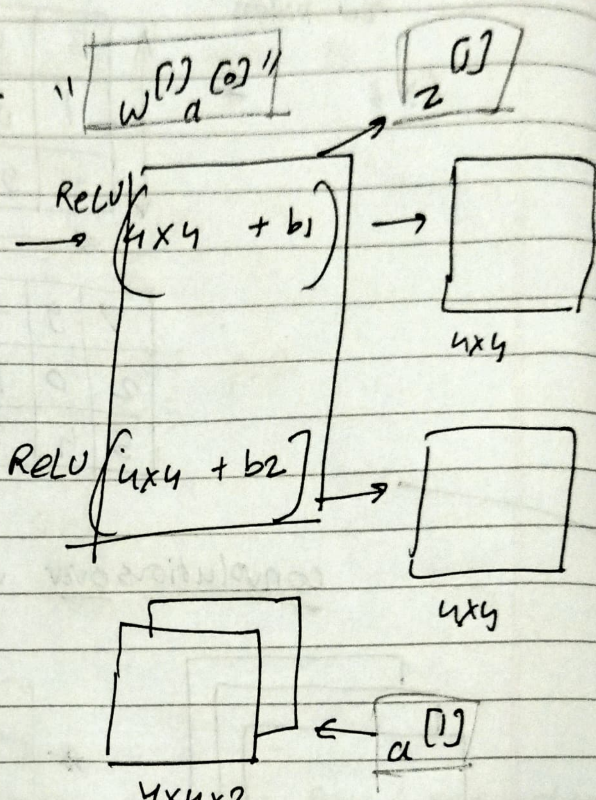
Summary $n \times n \times \boxed{n_c} * f \times f \times \boxed{n_c} \rightarrow n-f+1 \times n-f+1 \times n_c'$

$6 \times 6 \times 3 \qquad 3 \times 3 \times 3 \qquad 4 \quad 4 \quad 27$

filters used

☆ One layer of CONN

" $w^{[1]} a^{[0]}$ "



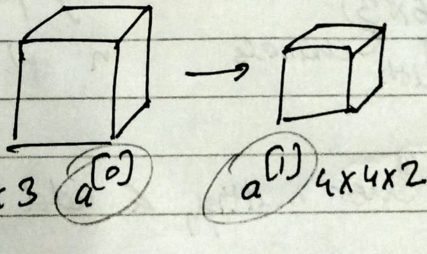
$6 \times 6 \times 3$ $\underbrace{\qquad}_{a^{[0]}}$

$* \; 3 \times 3 \times 3 \rightarrow$

$* \; \underbrace{3 \times 3 \times 3}_{w^{[1]}} \rightarrow$

$$z^{[1]} = w^{[1]} a^{[0]} + b^{[1]}$$

$$a^{[1]} = g(z^{[1]})$$



Q) 10 filters that are $3 \times 3 \times 3$ in one layer a neural network, how many parameters does that layer have?

$3 \times 3 \times 3$
27
+ bias $\quad \times 10$

$\boxed{280}$

summary of notation

If layer l is a convolutional layer:

$f^{[l]}$ = filter size
$p^{[l]}$ = padding
$s^{[l]}$ = stride

Input: $n_H^{[l-1]} \times n_w^{[l-1]} \times n_c^{[l-1]}$

Output: $n_H^{[l]} \times n_w^{[l]} \times n_c^{[l]}$

$$n_H^{[l]} = \left\lfloor \frac{n_H^{[l-1]} + 2p^{[l]} - f^{[l]}}{s^{[l]}} + 1 \right\rfloor$$

Each filter is: $f^{[l]} \times f^{[l]} \times n_c^{[l-1]}$

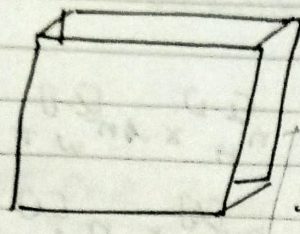
Activations: $a^{[l]} \rightarrow n_H^{[l]} \times n_w^{[l]} \times n_c^{[l]}$

$$A^{[l]} \rightarrow m \times \underbrace{n_H^{[l]} \times n_w^{[l]} \times n_c^{[l]}}_{n_c \times n_H \times n_w}$$

Weights: $f^{[l]} \times f^{[l]} \times n_c^{[l-1]} \times n_c^{[l]}$ ← # filters in layer l

bias: $n_c^{[l]}$ - $(1, 1, 1, n_c^{[l]})$

A Simple convolutional network example

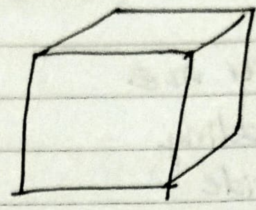


$39 \times 39 \times 3$

$n_H^{[0]} = n_W^{[0]} = 39$

$n_c^{[0]} = 3$

$f^{[1]} = 3$
$s^{[1]} = 1$
$p^{[1]} = 0$
10 filters

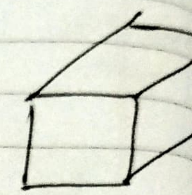


$37 \times 37 \times 10$

$n_H^{[1]} = n_W^{[1]} = 37$

$n_c^{[1]} = 16$

$f^{[2]} = 5$
$s^{[2]} = 2$
$p^{[2]} = 0$
20 filters



$17 \times 17 \times 2$

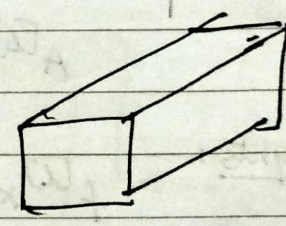
$n_H^{[2]} = n_W^{[2]} = 17$

$n_c^{[2]} = 20$

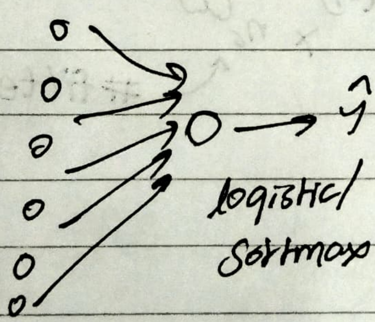
$$\frac{n+2p-f}{s} + 1$$

$$\frac{39+0-3}{1} + 1 = 37$$

$f^{[3]} = 5$
$s^{[2]} = 2$
40 filters



$7 \times 7 \times 40$



Types of layer in a convolutional network:

- convolution (conv)
- Pooling (POOL)
- Fully connected (FC)

Pooling layers

Max pooling

Max of each box

$$\begin{bmatrix} 1 & 3 & 2 & 1 \\ 2 & 9 & 1 & 1 \\ 1 & 3 & 2 & 3 \\ 5 & 6 & 1 & 2 \end{bmatrix} \longrightarrow \begin{bmatrix} 9 & 2 \\ 6 & 3 \end{bmatrix}$$

max of every 4 boxes

2x2

Hyperparameters:
$f = 2$
$s = 2$

No parameters!

$5 \times 5 \times n_c \longrightarrow 3 \times 3 \times n_c$

$f = 3$
$s = 1$

Average Pooling

Instead of max, avg of the inputs is taken for Average pooling

$7 \times 7 \times 1000 \rightarrow 1 \times 1 \times 1000$

To shrink, very rare

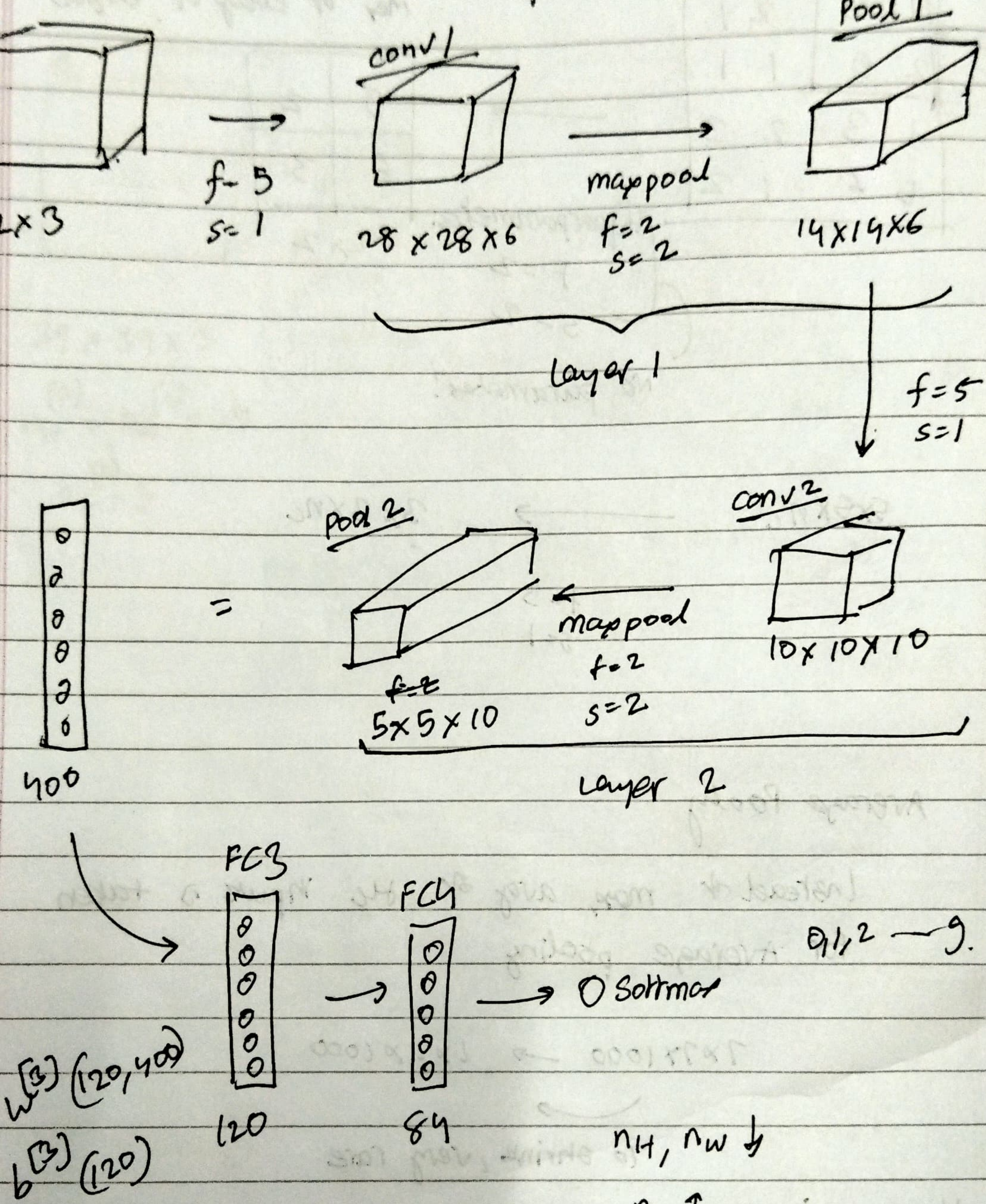
Hyperparameters:
- f: filter size
- s: stride
- Max or average pooling
- p: padding

$$n_H \times n_W \times n_C \longrightarrow \left\lfloor \frac{n_H - f}{s} + 1 \right\rfloor \times \left\lfloor \frac{n_W - f}{s} + 1 \right\rfloor \times n_C$$

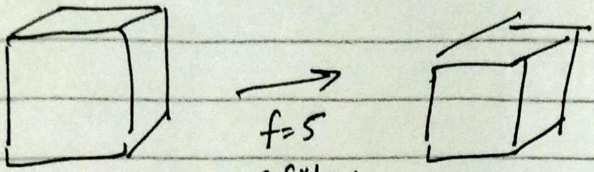
No parameters to learn

CNN Example

Neural Network Example (LeNet-5)



Why convolutions?



32×32×3   28×28×6

3702   4704

~~5×5 → 25~~ 26

~~6×26~~ = 156 parameters

$(5\times5\times3 + 1) * 6$

$3072 \times 4704 \approx 14M$

→ Parameter sharing   A feature detector in one part helpful in another part of image

→ Sparsity of connections   Depends on only a small number of inputs

Training set $(x^{(1)}, y^{(1)}) \ldots (x^{(m)}, y^{(m)})$

cost $J = \frac{1}{m}\sum_{i=1}^{m} L(\hat{y}^{(i)}, y^{(i)})$

Use Gradient descent

Test

$64 + 2p - 9 = 64$

$2p - 9 = 0$

$2p = 9$

$p = 4$

$\frac{63-7}{2} + 1$

$\frac{66 * -3}{3}$ 6 $21 + 1 = 22$

66

$63 + 2p - 86 = 0$

$2p = 6$

$66 - 3$

$\frac{64-9}{2p} + 1 =$