42028: Deep Learning and Convolutional Neural Network

Week-3 Lecture

Feature Extraction and Neural Network Basics

Outline

- Image Gradient
- Histogram of Oriented Gradient (HoG)
- Local Binary Pattern
- ANN Basics
- ANN Learning Process
- Logistic Regression using ANN
- Gradient Descent

Features Extraction

HoG (Histogram of Oriented Gradient*)

What is an Image Gradient?

- It is a directional change in the intensity or color in an Image.
- Can be used to extract valuable information from images.
- Commonly used in edge detection.

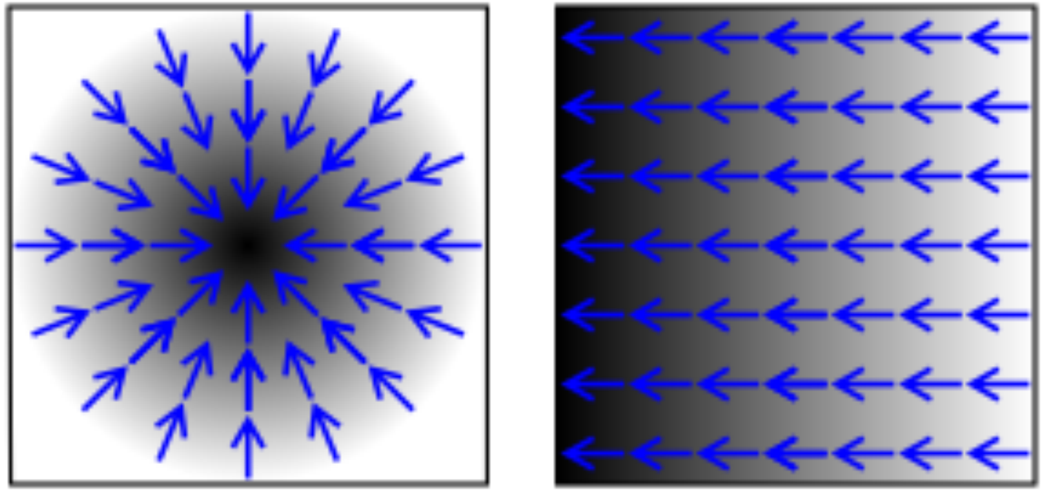
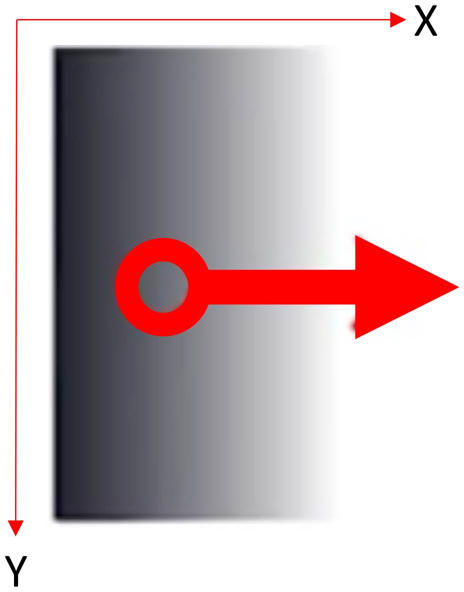


Image source: https://en.wikipedia.org/wiki/Image_gradient

Reference: http://lear.inrialpes.fr/people/triggs/pubs/Dalal-cvpr05.pdf

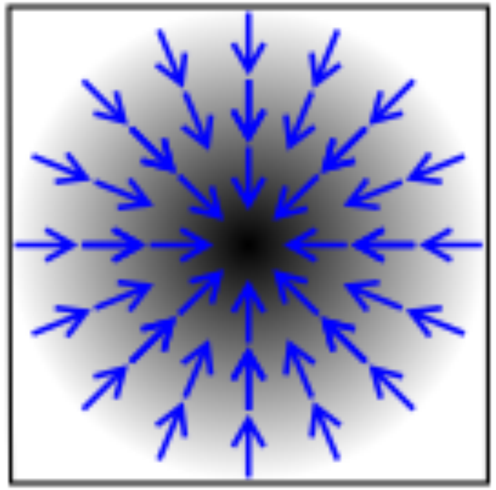
HoG (Histogram of Oriented Gradient*)

What is an Image Gradient?



Change is X-directions

Change is Y-directions



Combining both X and Y direction to estimate if changes are in both directions

Image source: https://en.wikipedia.org/wiki/Image_gradient

Reference: http://lear.inrialpes.fr/people/triggs/pubs/Dalal-cvpr05.pdf

HoG (Histogram of Oriented Gradient)

Step -1: Computing Image *Gradient*:

1. Use the horizontal and vertical filters to compute gradient values

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

Horizontal filter

Gradient is X-directions

$$g_y = \begin{bmatrix} -1 \\ 0 \\ 1 \end{bmatrix} * I$$

Gradient is y-directions

Vertical filter

Image source: https://en.wikipedia.org/wiki/Image_gradient

HoG (Histogram of Oriented Gradient)

2. Compute the strength/magnitude and direction of gradient.

Strength/Magnitude(g) = $\sqrt{g_y^2 + g_x^2}$

Direction $\theta = \tan^{-1}\left[\frac{g_y}{g_x}\right]$

Example

X	100	X
70	60	120
X	50	X

$g_x = |-70 + 120| = 50$
$g_y = |-100 + 50| = 50$

Gradient Magnitude = ~70.7
Direction/Angle = 45°

Image source: https://en.wikipedia.org/wiki/Image_gradient

HoG (Histogram of Oriented Gradient)

Step -2: Create orientation histogram:

- Divide the image into small connected regions called *Cells* which is a 8 X 8 patch
- Create cell histogram based on gradient direction and magnitude
- 64 (8 X 8) gradient vectors are put into a 9-bin histogram
- The bins are the gradient directions (θ) quantized into 9-bins

Reference: https://tanasecucliciu.wordpress.com/2016/06/08/programming-histogram-of-oriented-gradients-hog-explained/
https://www.learnopencv.com/histogram-of-oriented-gradients/

HoG (Histogram of Oriented Gradient)

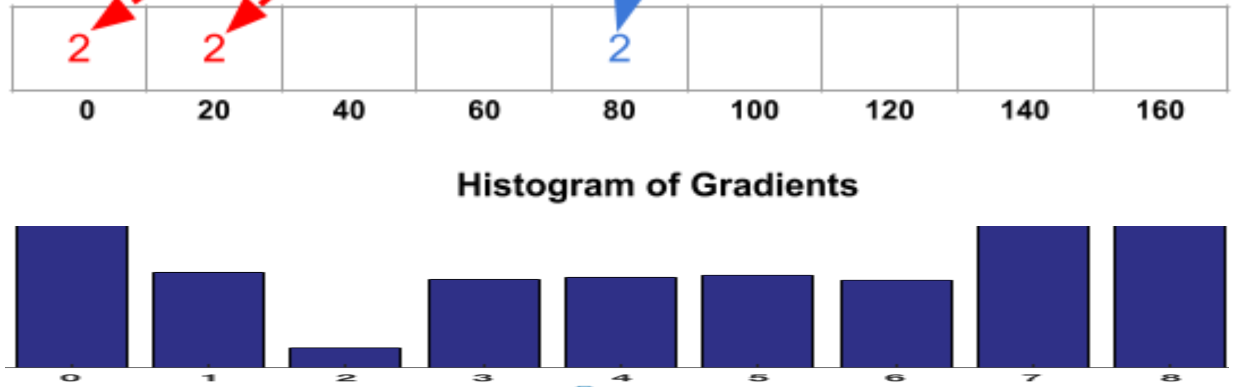
80	36	5	10	0	64	90	73
37	9	9	179	78	27	169	166
87	136	173	39	102	163	152	176
76	13	1	168	159	22	125	143
120	70	14	150	145	144	145	143
58	86	119	98	100	101	133	113
30	65	157	75	78	165	145	124
11	170	91	4	110	17	133	110

Gradient Direction

2	3	4	4	3	4	2	2
5	11	17	13	7	9	3	4
11	21	23	27	22	17	4	6
23	99	165	135	85	32	26	2
91	155	133	136	144	152	57	28
98	196	76	38	26	60	170	51
165	60	60	27	77	85	43	136
71	13	34	23	108	27	48	110

Gradient Magnitude

Pixel with blue circle has an angle of 80 degrees and magnitude of 2



Reference: https://tanasecucliciu.wordpress.com/2016/06/08/programming-histogram-of-oriented-gradients-hog-explained/
Image source: https://www.learnopencv.com/histogram-of-oriented-gradients/

HoG (Histogram of Oriented Gradient)

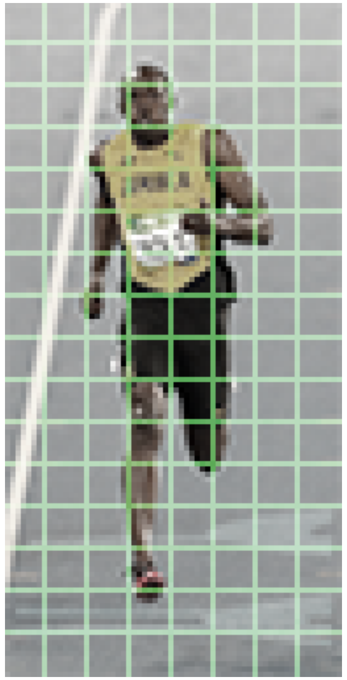
Step -3: Block Normalization:

- 16 X 16 pixels blocks or 2X2 cells are used for normalization, which has 4 histograms.
- Normalization will make it scale/multiplication invariant
- Each block will represent 36 X 1 element vector

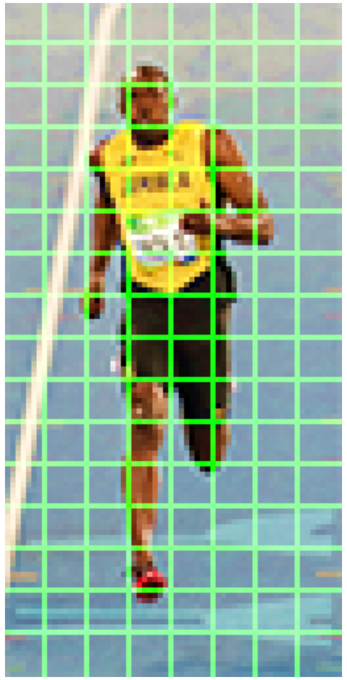
Reference: https://tanasecucliciu.wordpress.com/2016/06/08/programming-histogram-of-oriented-gradients-hog-explained/
https://www.learnopencv.com/histogram-of-oriented-gradients/

HoG (Histogram of Oriented Gradient)

Step -3: Block Normalization:


Brightness reduced


Original image


Brightness increased

Normalization example:

(3, 9) → $\sqrt{3^2 + 9^2} = 9.48$

(3/9.48 , 9/9.48) = (0.32, 0.95)

Multiple (3, 9) by 2 to increase brightness
(6, 18) → $\sqrt{6^2 + 18^2}$ = 18.97

(6/18.97, 18/18.97) = (~0.32, ~0.95)

Reference: https://tanasecucliciu.wordpress.com/2016/06/08/programming-histogram-of-oriented-gradients-hog-explained/
https://www.learnopencv.com/histogram-of-oriented-gradients/

HoG (Histogram of Oriented Gradient)

Step -4: Calculate the HOG feature vector:

- Each of the 36 X 1 vectors in each blocks are concatenated into one big vector.
- Size of the vector will be:

Number of blocks X 36

Example: For an Image size: 64 X 128, will have 8 X 16 cells,

and 7 X 15 block (with 50% overlap),

hence size of HOG feature vector: 7 X 15 X 36 = 3,780

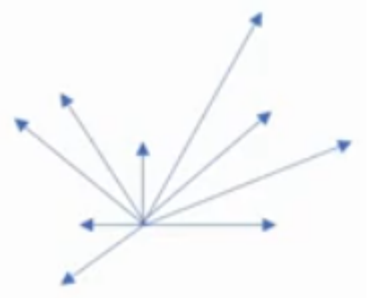
Reference: https://tanasecucliciu.wordpress.com/2016/06/08/programming-histogram-of-oriented-gradients-hog-explained/
https://www.learnopencv.com/histogram-of-oriented-gradients/

HoG (Histogram of Oriented Gradient)

Example:


```python
from skimage.feature import hog
from skimage import data, color, exposure
import cv2

import matplotlib.pyplot as plt
image = cv2.imread('new_image.png')
image = color.rgb2gray(image)

fd, hog_image = hog(image, orientations=8, pixels_per_cell=(16, 16),
                    cells_per_block=(1, 1), visualise=True)

plt.figure(figsize=(8, 4))

plt.subplot(121).set_axis_off()
plt.imshow(image, cmap=plt.cm.gray)
plt.title('Input image')

# Rescale histogram for better display
hog_image_rescaled = exposure.rescale_intensity(hog_image, in_range=(0, 0.02))

plt.subplot(122).set_axis_off()
plt.imshow(hog_image_rescaled, cmap=plt.cm.gray)
plt.title('Histogram of Oriented Gradients')
plt.show()
```

Visualisation of the histogram (Magnitude and direction)



Reference: http://scikit-image.org/docs/0.6/auto_examples/plot_hog.html

Local Binary Pattern (LBP)

- An efficient texture operator which labels each pixels of an image by thresholding their neighbours.

- A powerful feature for texture classification

- The idea behind the LBP operator is to describe the image textures using two measures namely, local spatial patterns and the gray scale contrast of its strength.

Reference: https://ieeexplore.ieee.org/abstract/document/6889906

Local Binary Pattern (LBP)

- The basic $LBP_{P,R}$ operator is defined as follows:

$$LBP_{P,R}(x_c, y_c) = \sum_{p=0}^{P-1} S(g_p - g_c)2^P$$

$$S(x) = \begin{cases} 1, if x >= 0 \\ 0, otherwise \end{cases}$$

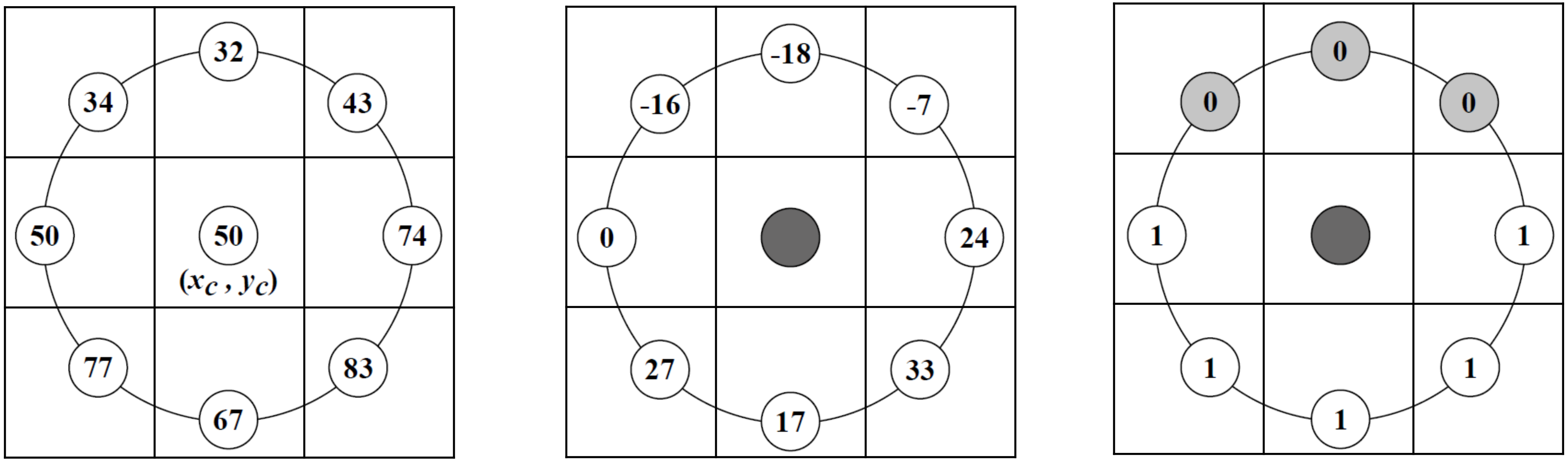
Where,

S(x) → a thresholding function
(x_c , y_c) → the centre pixel in the 8 pixel neighbourhood,
g_c → gray level of the centre pixel
g_p → gray value of a sampling point in an equally spaced circular neighbourhood of P sampling points and radius R around the point (x_c , y_c)

Reference: https://ieeexplore.ieee.org/abstract/document/6889906

Local Binary Pattern (LBP)

An Example of LBP Computation:



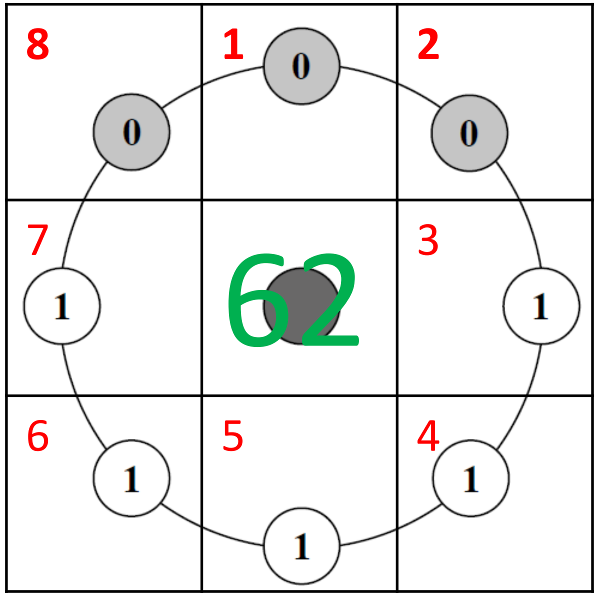
(a) Sample pixel neighbourhood (b) Difference result (c) Thresholding result

Reference: https://ieeexplore.ieee.org/abstract/document/6889906

Local Binary Pattern (LBP)

An Example of LBP Computation:

An 8-digit binary number is obtained by considering the thresholding result, starting from pixel 1 to 8, as marked in red.



- There can be $2^8 = 256$ possible values
- Hence, the LBP histogram will have **256 bins →feature vector**

00111110 =
$(0 \times 2^7) + (0 \times 2^6) + (1 \times 2^5) + (1 \times 2^4) + (1 \times 2^3) + (1 \times 2^2) + (1 \times 2^1) + (0 \times 2^0) = 62$

Reference: https://ieeexplore.ieee.org/abstract/document/6889906

Local Binary Pattern (LBP)

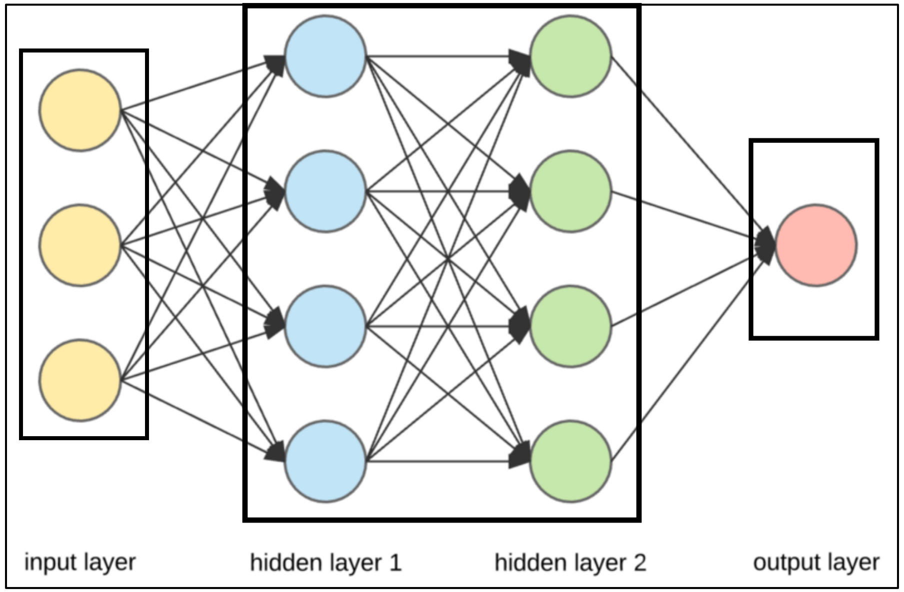
An Example of LBP Computation:

Reference: https://ieeexplore.ieee.org/abstract/document/6889906

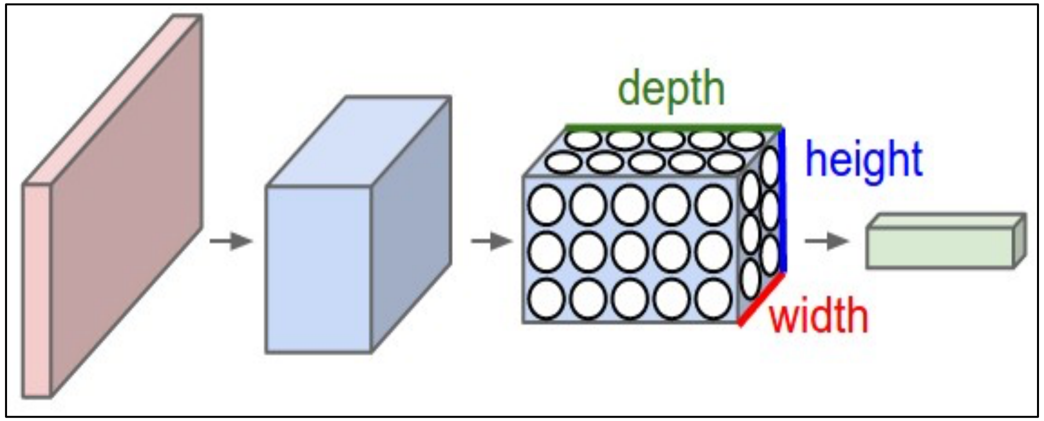
Neural Network Basics

What is Artificial Neural Network (ANN)?

- Artificial Neural Networks (ANN) are multi-layered fully-connected neural networks.
- It has an input layer, multiple hidden layers and an output layer



Standard ANNs



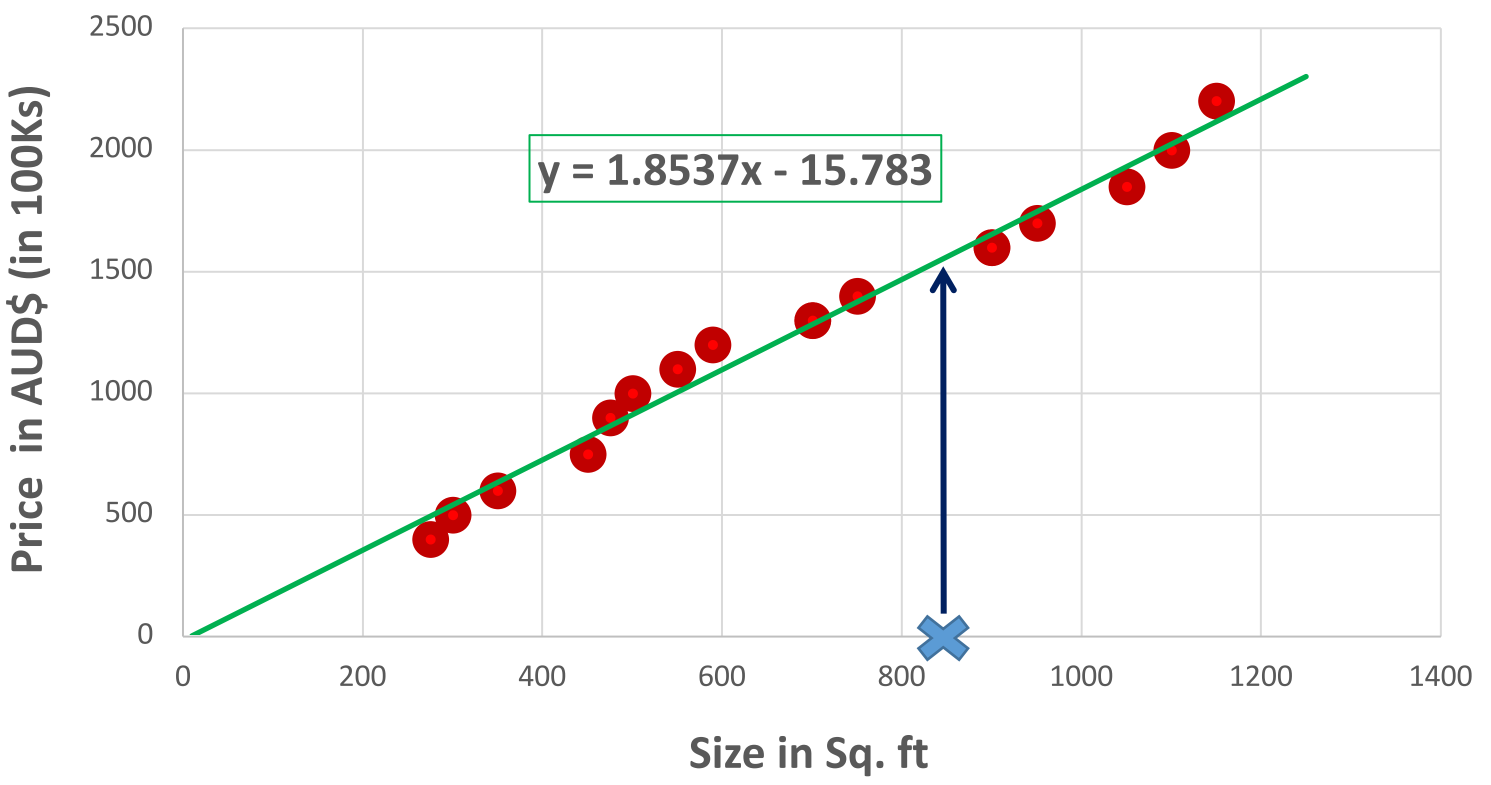
CNNs

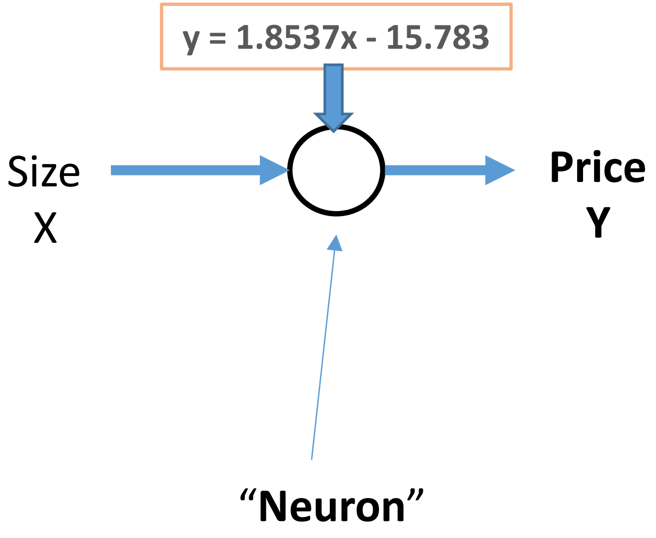
Reference & Image source: https://towardsdatascience.com/applied-deep-learning-part-1-artificial-neural-networks-d7834f67a4f6

Image source: http://cs231n.github.io/assets/cnn/cnn.jpeg

ANN Introduction

House Price prediction





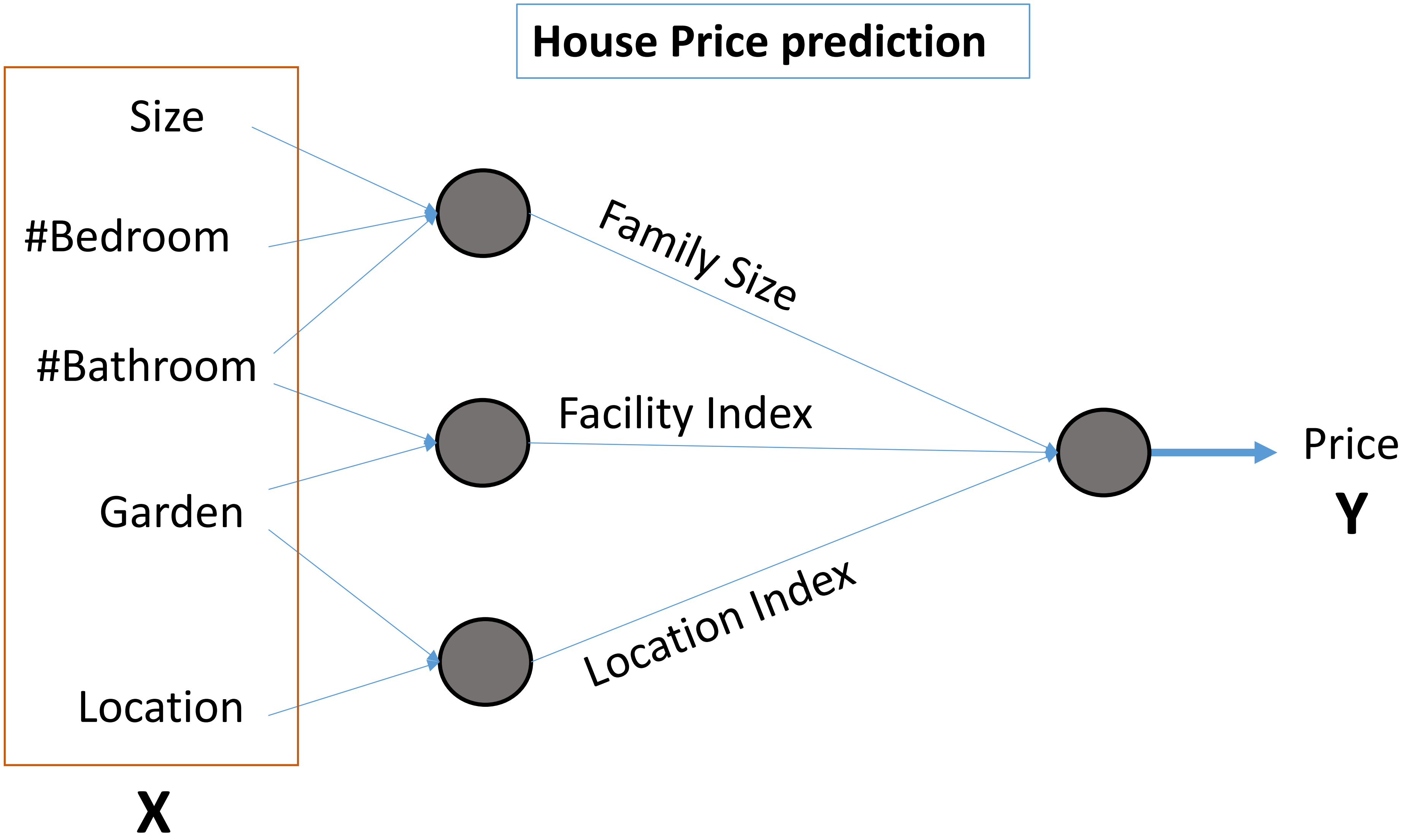
ANN Introduction

House Price prediction

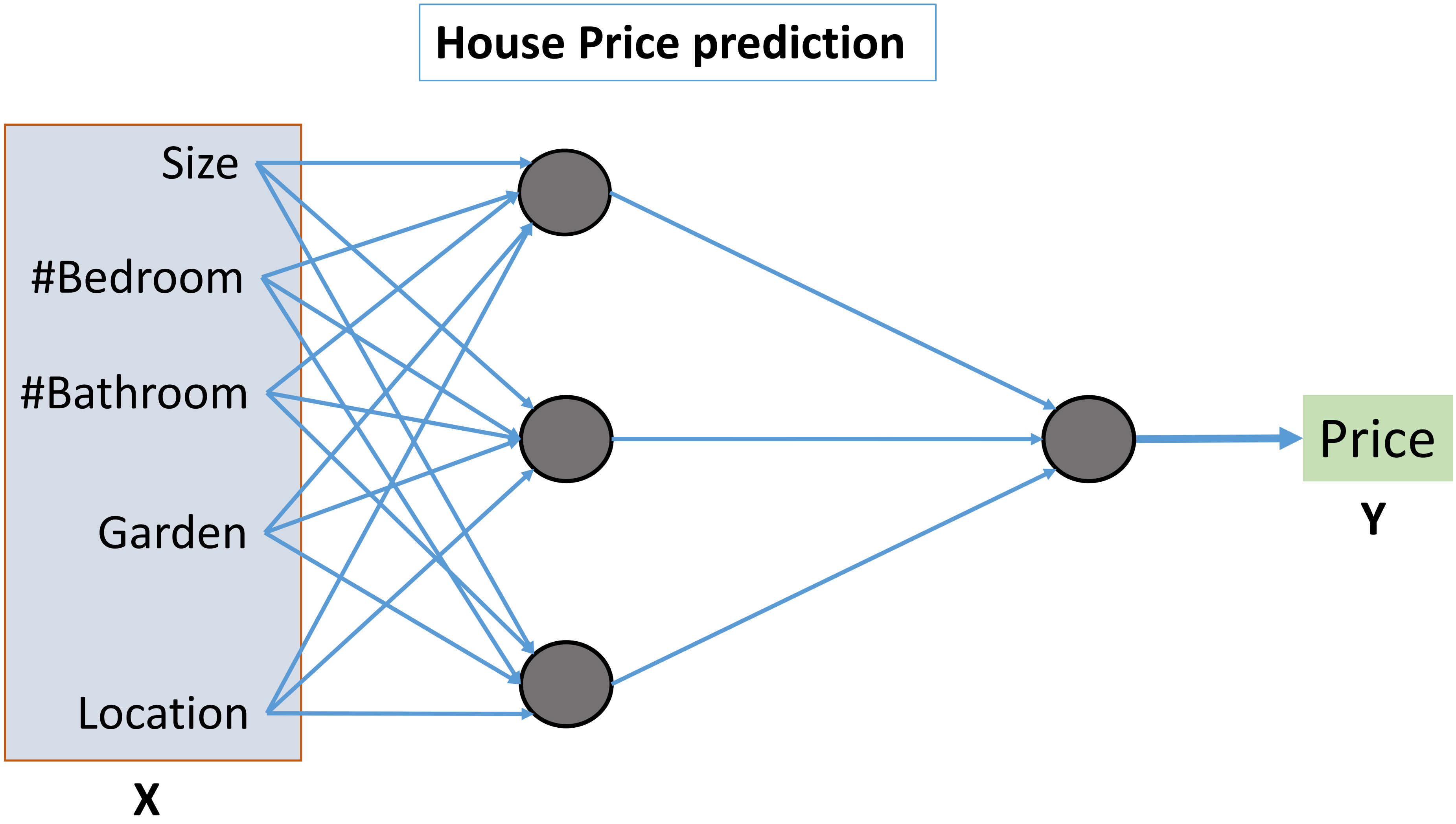
	Variables Used
Quality Index	Floors, windows type, kitchen furniture and reformations
Facilities Index	Pool, tennis, garden
Building Index	Age, lift, laundry
External data Index	Orientation, terrace
Extras Index	Garage, storage
Location Index	Geographical position within the city

Source and Reference: https://www.econstor.eu/bitstream/10419/113851/1/756619068.pdf

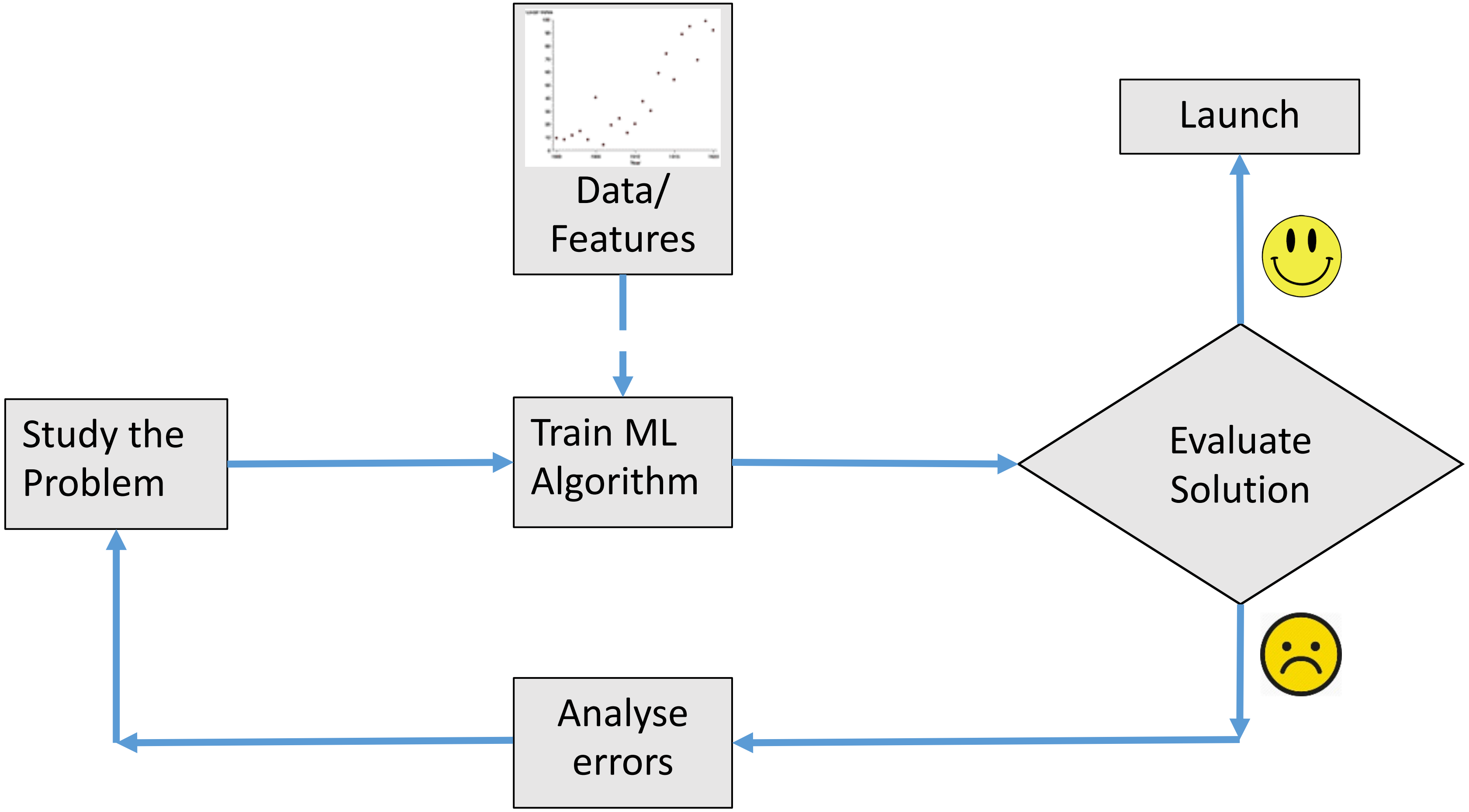
ANN Introduction



ANN Introduction



ANN Introduction – Learning Process



ANN Introduction – Learning Process

Problem of Binary Classification:

Shark ? → 1

Not Shark? → 0

Mechanism to reduce the loss in the model

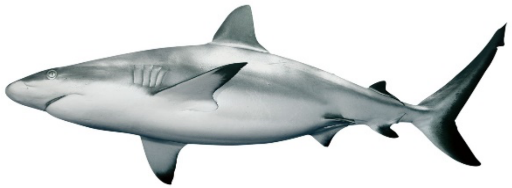
Gradient

Function to calculate the loss/error

Error/ Loss

1

Target



Input (x)

Model

ANN Architecture

+

Parameters

0

Output (y)

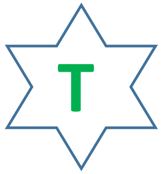
ANN Introduction – Learning Process: Example

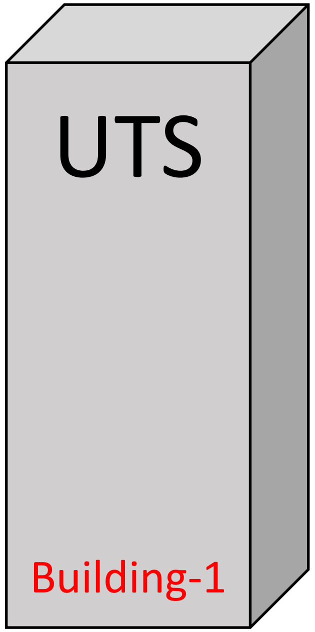
S

Position: (x, y)

Position: (x+dx, y+dx)

Target





d

D

Distance need to cover
to reach target

Distance remaining = (D – d)
(Error/Loss to minimize)

Update Position (parameter):
x = x + dx
y = y + dy

Logistic Regression as Neural Network

Problem of Binary Classification → Logistic Regression (Shark ? → 1 | Not Shark? → 0)

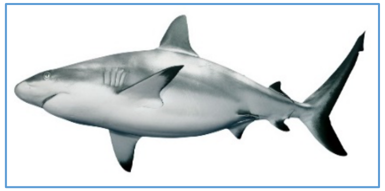
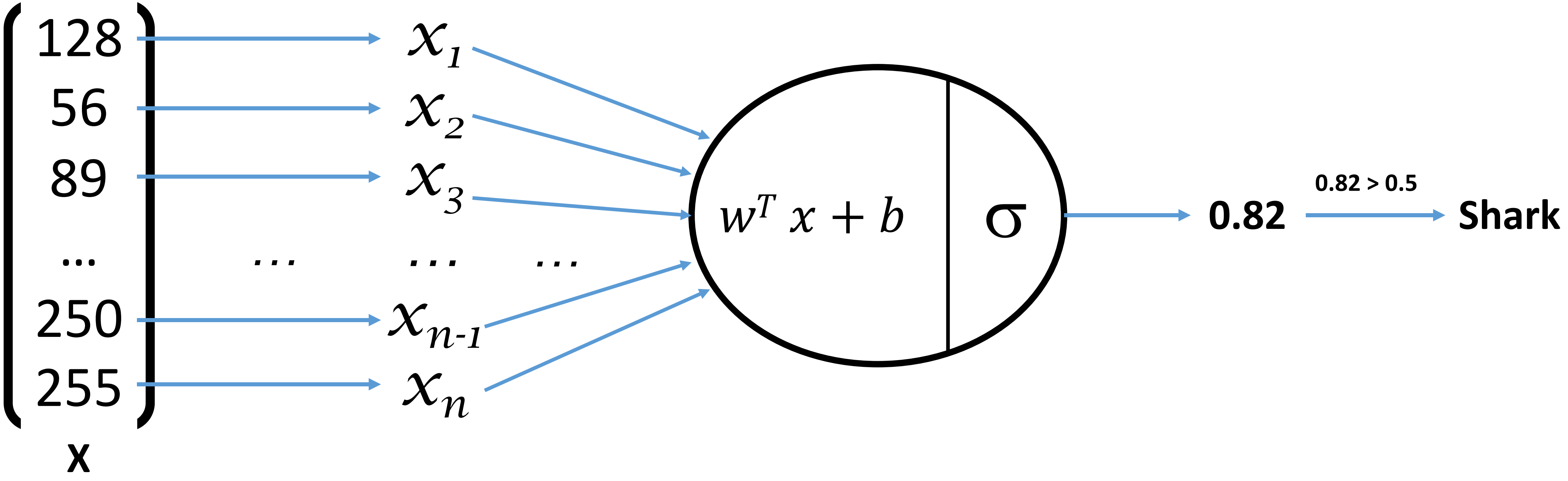


Image dimension:
64X128 = **8192 Pixels**

image.reshape(image.shape[0]*image.shape[1]*image.shape[2],1)



Source and Reference: http://cs230.stanford.edu/files/C1M2.pdf

Logistic Regression as Neural Network

Problem of Binary Classification → Logistic Regression (Shark ? → 1 | Not Shark? → 0)

$\boxed{w^T x} + \boxed{b}$ → Linear function of input *x*

$$\boxed{w_1 x_1 + w_2 x_2 + \ \ldots + wnxn \quad + \ b}$$

Weighted sum of inputs

Where,

W → Weights

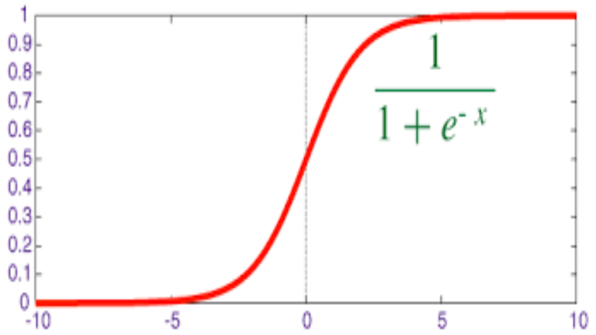
X → Inputs

b → Bias term

σ → Activation function

σ =

(Sigmoid function)



Rule of thumb:
In case of binary classification, Sigmoid function is the obvious choice for output layer

Source and Reference: http://cs230.stanford.edu/files/C1M2.pdf

Logistic Regression as Neural Network

Problem of Binary Classification → Logistic Regression (Shark ? → 1 | Not Shark? → 0)

Parameters:

1. **w (weight)**
2. **b (bias)**
3. **Output a =** $\sigma(\boldsymbol{w}^T \boldsymbol{x} + \boldsymbol{b})$

Loss function for Logistic Regression:

$$\mathcal{L}(a, y) = -(y \log a + (1 - y)\log(1 - a))$$

Logistic Regression pipeline with the math looks like:

X, **W**, **B** → $\boxed{\boldsymbol{w}^T \boldsymbol{x} + \boldsymbol{b}}$ → $\boxed{a = \sigma(\boldsymbol{w}^T \boldsymbol{x} + \boldsymbol{b})}$ → $\boxed{\mathcal{L}(a, y)}$

Logistic Regression as Neural Network

Problem of Binary Classification → Logistic Regression (Shark ? → 1 | Not Shark? → 0)

Gradient Descent for learning parameters:
It is an iterative approach for error correction in a machine learning model.

For 1 Sample the loss function is:
$\mathcal{L}\,(\mathrm{a}, \mathrm{y}) = -\,(y \log a + (1 - y)\log(1 - a))$

For m Sample the loss function is:
$\mathcal{GD}(w,\ b) = x = \frac{1}{m}\sum_{\mathrm{i}=1}^{\mathrm{m}} \mathcal{L}\,(a, \mathrm{y})$

Question: Find w and b that will minimize $\mathcal{GD}(w,\ b)$

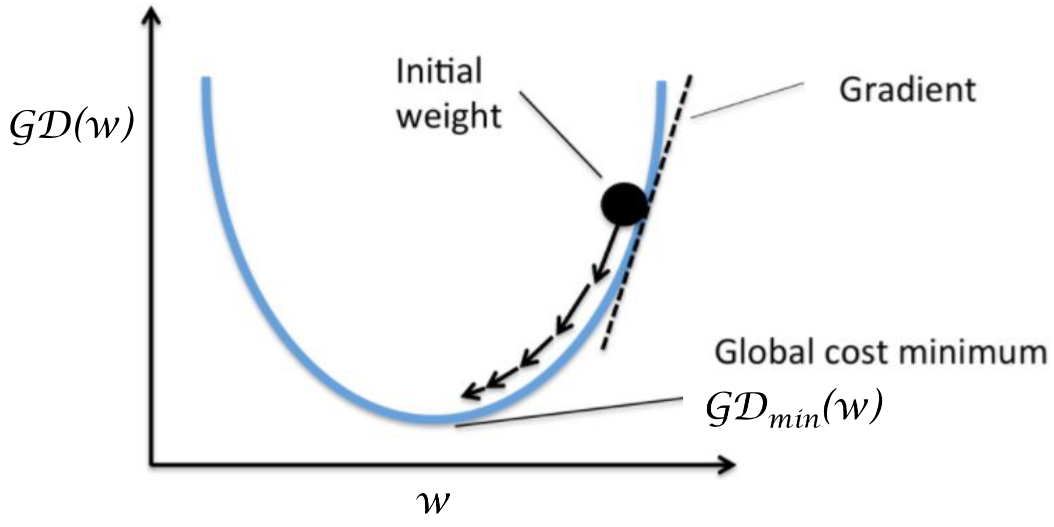


Image Source: https://subscription.packtpub.com/book/big_data_and_business_intelligence/9781788397872/1/ch01lvl1sec22/gradient-descent
Source and Reference: http://cs230.stanford.edu/files/C1M2.pdf

Logistic Regression as Neural Network

Problem of Binary Classification → Logistic Regression (Shark ? → 1 | Not Shark? → 0)

Gradient Descent for learning parameters:
It is an iterative approach for error correction in a machine learning model.

Updating the ***w*** and ***b*** iteratively, :
$\boldsymbol{w} = \boldsymbol{w} - \alpha \boldsymbol{dw}$

Updating the ***b***:
$\boldsymbol{b} = \boldsymbol{b} - \alpha \boldsymbol{db}$

Where,

$\boldsymbol{dw} = \dfrac{\partial GD(w,b)}{\partial w}$

$\boldsymbol{db} = \dfrac{\partial GD(w,b)}{\partial b}$

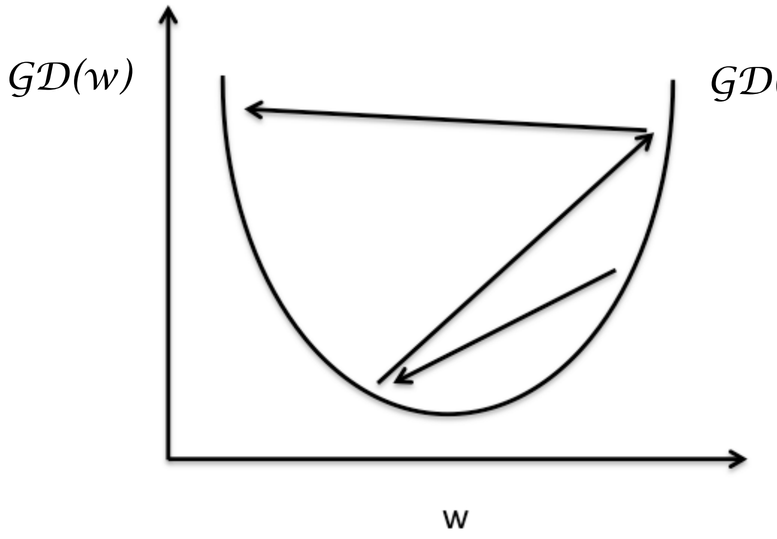
α → Learning rate

Logistic Regression as Neural Network

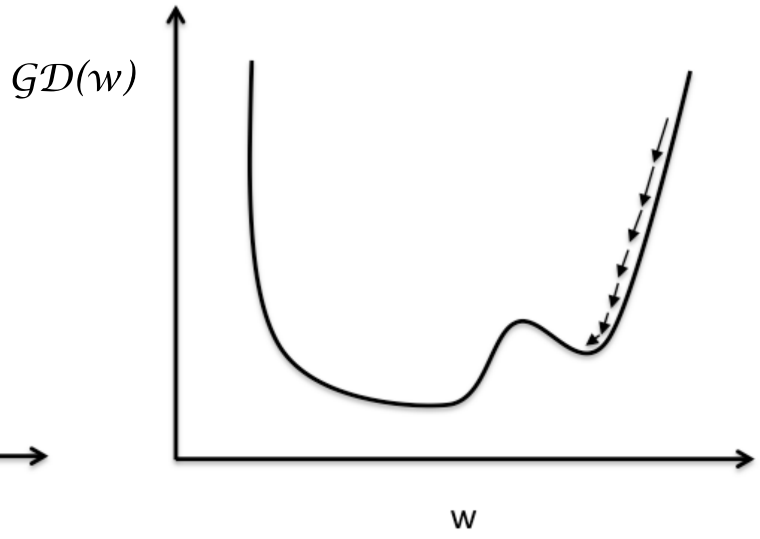
Problem of Binary Classification → Logistic Regression (Shark ? → 1 | Not Shark? → 0)

Gradient Descent for learning parameters:

Learning rate(α) issues:



Large learning rate: Overshooting.



Small learning rate: Many iterations until convergence and trapping in local minima.