

CMPE258
Spring 2023

Jan. 26 (Thu)

Organizational Meeting
for Deep Learning Class.

Syllabus, "GreenSheet".

Note: 1° Syllabus is posted on the Class
github. Also SJSU CANVAS

<https://github.com/hualili/opencv/tree/master/deep-learning-2022s>

2023S-100-accessible-CMPE258-S23-v7-H...

San José State University
College of Engineering
Computer Engineering Department
CMPE258-Section 1 Deep Learning
S2023

Course and Contact Information

Instructor: Hua Harry Li, Ph.D.

Office Location: Engineering Building, Room 267A

Telephone: Mobile (650) 400-1116 Text message only

Email: hua.li@sjsu.edu

Office Hours: MW 4:30 - 5:30 PM;

On-line with Zoom

Join Zoom Meeting

<https://us04web.zoom.us/j/98416076832>

pwd=U1A3aEk1TnV4bjNLQk5CQkw0dDk4UT09 Meeting ID: 984

160 7683 Passcode: 121092

Class Days/Time: Tuesdays, Thursdays 4:30 - 5:45 PM

Classroom: Zoom (link to be shared in the SJSU email)

Note: 4° Office Hours. On Zoom.

The office hours are good for the
entire school semester, e.g.
from the 1st day of the class till the
last day of lecture.

Note: 3° Class on Zoom. Video
Cam Activation for the
class is required. Have
your Video Cam Ready By Next Session

3° Attendance Requirement: Attend Lecture
ON-Line is required.

CMPE258

Spring 2023

Note: 5. Class github. CANVAS is the only source for All Submissions, including Homeworks, Projects, Exam Papers etc. No

Faculty Web Page and MYSJSU Messaging (Optional)

Copies of the course materials such as the syllabus, major assignment handouts, etc. can be found on line at SJSU CANVAS, the same material is also provided at the following yahoo group, see URL below: e-mail

<https://github.com/hualili/opencv/tree/master/deep-learning-2022s>

Submission is Accepted.

Office hours zoom link: Join Zoom Meeting [https://us04web.zoom.us/j/9841607683?](https://us04web.zoom.us/j/9841607683?pwd=U1A3aEk1TnV4bjNLQk5CQkw0dDk4UT09)

pwd=U1A3aEk1TnV4bjNLQk5CQkw0dDk4UT09 Meeting ID: 984 160 7683 Passcode: 121092

Course Description

Note: Pre-requisite CMPE 255 OR CMPE 257 is required.

Deep neural networks and their applications to various problems, e.g., speech recognition, image segmentation, detection and recognition of temporal and spatial patterns, and natural language processing. Covers underlying theory, the range of applications to which it has been applied, and learning from very large data sets.

Prerequisite: CMPE 255 or CMPE 257 or instructor consent. Computer Engineering and Software Engineering majors only.

Course Learning Outcomes (CLO)

Note: Book Listed below is a good reference source.

Required Texts/Readings

Textbook

- Deep Learning with Python, 1st Edition, by François Chollet, ISBN-13: 978-1617294433, ISBN-10: 9781617294433, <https://github.com/hualili/opencv/blob/master/IP120-AI-DL/2018F/2018F-6-DeepLearningCh02.pdf>
- Robot Vision by B.K. P. Horn, the MIT press, ISBN 0-262-08159-8, or 0-07-030349-5 (McGraw Hill).
- Reference textbook Learning OpenCV, Computer Vision with the OpenCV Library by Bradski and Kaebler, O'Reilly Publisher, ISBN 978-0-596-51613-0, 2011.

Other Readings

1. OpenCV on line reference: <http://docs.opencv.org/index.html>
2. OpenGL on line reference (OpenGL programming guide): ftp://ftp.sgi.com/opengl/contrib/kschwarz/OPEN_GL/REFERENCE/OGL_PG/oglPG.pdf
3. My lecture notes <https://github.com/hualili/opencv/tree/master/IP120-AI-DL/2018F> and <https://github.com/hualili/opencv/tree/master/deep-learning-2020S>

2022F-101-cmpe258-note-part2-2022-12-6...

References from the lecture Note

Keyword "note"

Other equipment / material requirements

1. Python.
2. Or you may choose C++ as an option.
3. OpenCV.
4. Tensorflow Keras API.
5. Optional embedded board for assignment and projects: Nvidia Jetson NANO

Pycharm "Tool", Development/Debugging/Testing with Colab, Jupyter Note Book etc. All Line Tools are O.K. However, for the project Submission, Stand-Alone Python Code for Deployment is Required. Also this

Course Requirements and Assignments

SJSU classes are designed such that in order to be successful, it is expected that student of forty-five hours for each unit of credit (normally three hours per unit per week), including participating in course activities, completing assignments, and so on. More details about

In the exams, the Deployable,
Stand-Alone Code is Required.

2 projects, Team/Semester-Long

Project

Final Presentation
(Last week of the Semester).

Grading Policy

Quiz, Homework, Projects	30%
Midterm Examination	30%
Final Examination	40%

CMPE258 Deep Learning, S2022.

On-Line.

0-59	F
60-69	D
70-79	C
80-89	B
90-100	A

Classroom Protocol

Note: Homework Submission.
One week from Today.

Will post the Homework
on CANVAS.

From University Policy F15-7:

1.0 DEFINITIONS OF ACADEMIC DISHONESTY

1.1 CHEATING

San José State University defines cheating as the act of obtaining credit, attempting to obtain credit, or assisting others to obtain credit for academic work through the use of any dishonest, deceptive, or fraudulent means. Cheating includes:

- 1.1.1. Copying, in part or in whole, from another's test or other evaluation instrument, including homework assignments, worksheets, lab reports, essays, summaries, and quizzes;
- 1.1.2. Submitting work previously graded in another course without prior approval by the

Jan 31 (Thu).

[opencv / deep-learning-2022s / 2022F-103-NN-Intro-Python-v5-2022-8-25.pdf](#)

1. Class Rep

2. Honesty Pledge Due
this Thursday. Opt.

3. OpenCV & Anaconda
Installation Due A week
from Today. 1 Pt.

master - [opencv / deep-learning-2023s /](#)

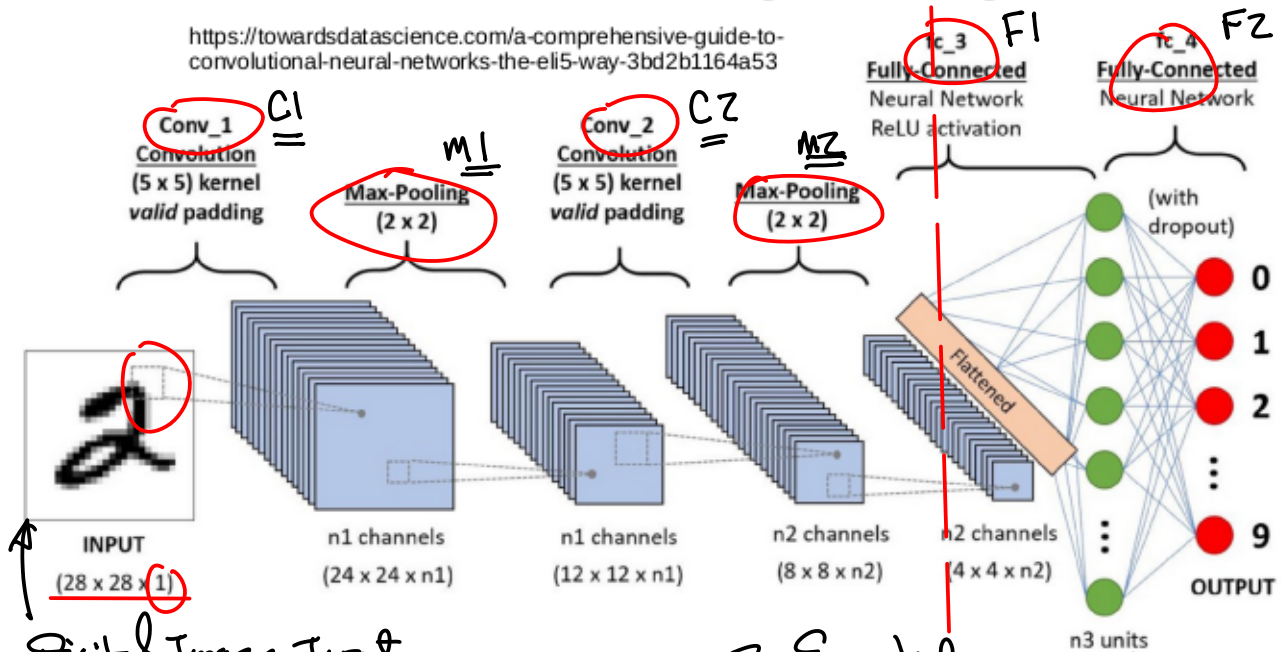
2023S-101-Note-cmpe258-2023-01-29.pdf	Add files via upload
2023S-200-canny.py	Add files via upload
accessible-CMPE258-S23-v7-HarryLi.pdf	Add files via upload

Sample Code: Reads Image
file, and displays
it.

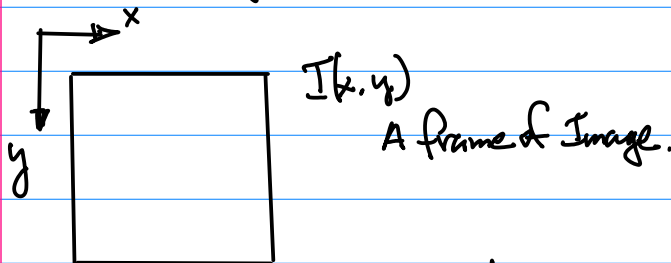
Example:

Illustration of A CNN for Digits Recognition

<https://towardsdatascience.com/a-comprehensive-guide-to-convolutional-neural-networks-the-eli5-way-3bd2b1164a53>



1. Digital Image Input.

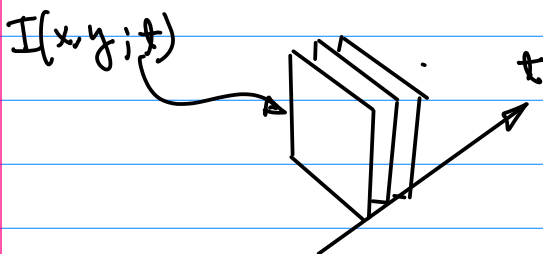


$M \times N$ Resolution
No. of Col. Per Row.
No. of Rows Per frame

1920x1280

M N

Pixel: picture element
for a Video clip, OR Video Stream.



2. Example for $28 \times 28 \times 1$,

8 bit Gray scale image

$I(x,y) \in [0,255]$

Brightest, White
Darkest, Black

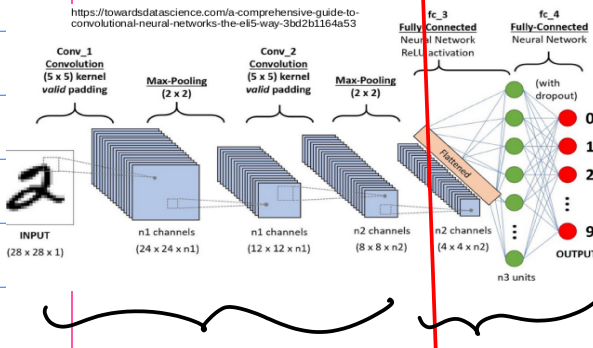
3. the "square" indicated on the input Image plane shows an Arbitrary Location of a Convolution Kernel.

4. "Conv-1" $\rightarrow C \rightarrow C1$

Simplified Notation. Adding An Index to Label the Convolution Block

"Max-Pooling" $\rightarrow M \rightarrow M1$

Hence, the Architecture for the Above CNN (Convolutional Neural Network) is Can be described as $C1M1C2M2$



Feature Extraction Decision Making.

Fully Connected (Feed Forward) Dense NN $\rightarrow F$

$F1 \rightarrow$

And then the 2nd Block: $F2$.

Conclusion: The Architecture is defined as $C1M1C2M2F1F2$

Feb 2nd (Thu).

Note: 1^o Attendance, please use

Chat message to text me privately

Your First-Last Name, and 4-Digits STD.

2^o Introduction & Team Formation.

Ref: from the class github.

1^o [2022F-103b-NN-Intro-Python-v5-2022-8-25.pdf](#) \leftarrow PPT

2^o Lecture Notes (Last Semester)

[2022F-101-cmpe258-note-2022-11-1.pdf](#)

3^o White paper on Single Neuron Formulation

[2022S-103a-notation-neuro-loss-function-2022-2-8.pdf](#)

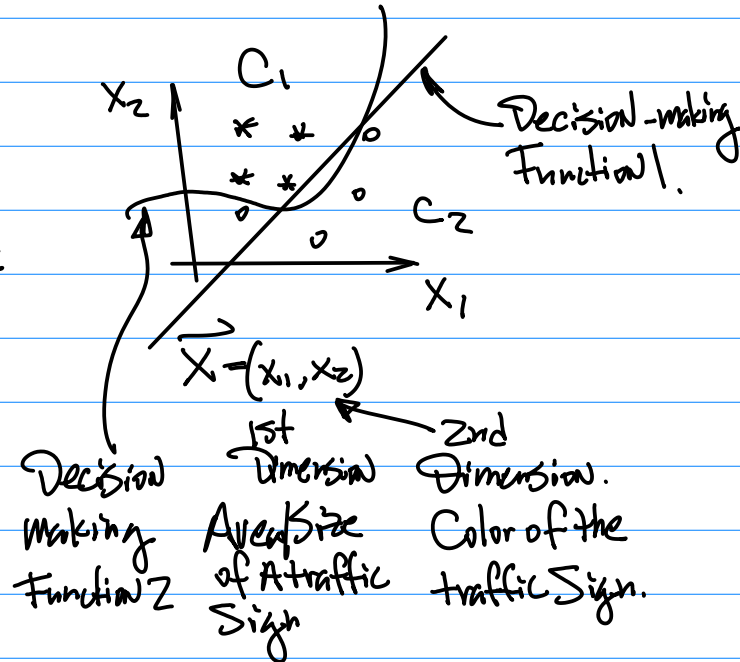
Example: Continuation of the Architecture

Question: How to Extract more features in general?

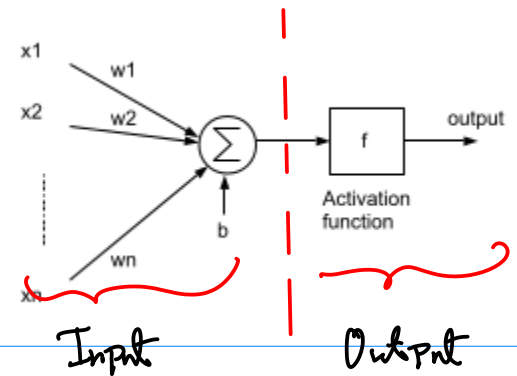
$C1M1$ Architecture v.s. $C1M1C2M2$?

In general we can increase the Number of Convolutional Blocks. for Example:

$C1M1C2M2C3M3$.



Consider A Single Neuron Below :



Note: 1^o Input/Excitation

$$X = (x_1, x_2, \dots, x_n) \dots (1)$$

2^o Weights, Links to Allow X to Connect to A Neuron.

$$W = (w_1, w_2, \dots, w_n) \dots (2)$$

$$w_i \in [0, 1]$$

3° Combining All the Inputs

$$x_1 w_1 + x_2 w_2 + \dots + x_n w_n \dots (3)$$

Notation for a neuron input x_i , $i = 1, 2, \dots, N$ is written as

$$\{x_i | i = 1, 2, \dots, N\} \quad (1)$$

and its vector form is

$$(x_1, x_2, \dots, x_N) \quad (2)$$

or simply denoted as X .

Now, introduce a superscript j for experiment j . The input is x_i^j , and $i = 1, 2, \dots, N$ and $j = 1, 2, \dots, P$.

$$\{x_i^j | i = 1, 2, \dots, N; j = 1, 2, \dots, P\} \quad (3)$$

III. NOTATION FOR WEIGHTS

Notation for a weight w_i , $i = 1, 2, \dots, N$ is written as

$$\{w_i | i = 1, 2, \dots, N\} \quad (4)$$

and its vector form is

$$(w_1, w_2, \dots, w_N) \quad (5)$$

or simply denoted as W .

And

$$x_1 w_1 + x_2 w_2 + \dots + x_n w_n = \sum_{i=1}^N x_i w_i \dots (3-1)$$

And

$$\sum_{i=1}^N w_i x_i = w_1 x_1 + w_2 x_2 + \dots + w_N x_N \quad (9)$$

Or simply in a short hand vector form notation:

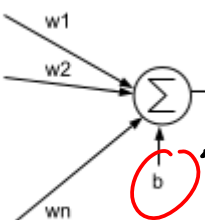
$$\sum_{i=1}^N w_i x_i = W \cdot X.$$

V. A TRANSFER FUNCTION

A transfer function is defined as

$$h = \sum_{i=1}^N w_i x_i = W \cdot X + b \quad (11)$$

x_1
 x_2
 \vdots
 x_n



Adding A "bias" (Threshold) to Regulate/Control the Input, then the Response of the Neuron.

$$\sum_{i=1}^N x_i w_i \text{ v.s. } \sum_{i=1}^N x_i w_i + b \quad \text{offset}$$

4. Define A Transfer function

$$h = \sum_{i=1}^N x_i w_i + b \text{ OR } \dots (4)$$

$$= \sum_{i=1}^{N+1} x_i w_i, \text{ where } x_{N+1} = b.$$

$$\text{And } w_{N+1} = 1 \dots (4-1)$$

$$h \rightarrow h(x_i, w_i) \rightarrow h(X, W; b)$$

5. Activation Function.

Denoted as "f"

$$y = f \dots (5)$$

Output f

$$f(x_i, w_i) \dots (5-1)$$

$$f(h(x_i, w_i)) \text{ OR } \dots (5-2)$$

$$f(h(x_i, w_i; b)) \dots (5-3)$$

Vector Dot-Product
(x_1, x_2) · (w_1, w_2)
(10)

$$= x_1 w_1 + x_2 w_2$$

$$y = f\left(\sum_{i=1}^N w_i x_i = W \cdot X + b\right). \quad (17)$$

where y is the output of the neuron, and the activation function can be rewritten as

$$y = f\left(\sum_{i=1}^N w_i x_i = W \cdot X + b\right) = f(h(w_i, b)). \quad (18)$$

Or simply written as

$$y = f(h(\cdot)) = f(h(w_i, b)). \quad (19)$$

6. Output from a Neuron y

Output from a Neural Network from an Experiment.

Suppose it is from the experiment j
 $j=1, 2, \dots, M$
 $\hat{y}_j \leftarrow$ Superscript.

Feb 7 (Tue).

Note: 1st github Ref.

2022S-103a-notation-neuro-loss-function-2022-2-8.pdf

2022F-103b-NN-Intro-Python-v5-2022-8-25.pdf

2022F-101-cmpe258-note-2022-11-1.pdf

White paper

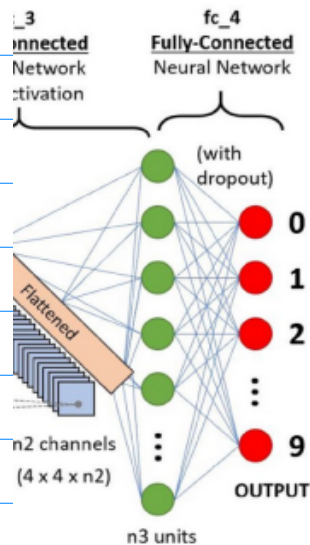
PPT.

Notes

Example:

Note: 1. Output y

$$y_i = f_i(h(\cdot)) = f_i(h(w_i, b))$$



y_i

$i=0, 1, 2, \dots$

\hat{y}_i output @
Node i

Fig. 1

Extend the Notation \hat{y}_i to
allow us to describe an experiment
 j , for $j=1, 2, \dots, N$

$$\hat{y}_i^j \mid \begin{matrix} j=1 \\ j=2 \end{matrix} \hat{y}_i^1$$

Now, consider the performance of
a Convolution Neural Network
Comparison of the Network Output
 \hat{y} to the "Ground Truth"

y

\hat{y}/y OR $\hat{y}-y$ for A
= Single
Output

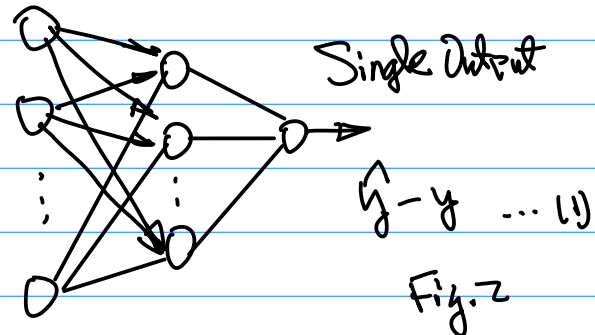


Fig. 2

For more than one Nodes, in Fig. 1.

$$\hat{y}_i - y_i \dots (1-b)$$

If for multiple experiments, then

$$\hat{y}_i^j - y_i^j$$

at Node i , Experiment/Training j , we
are Comparing the Network Output to
-h | e ground Truth.

Note: This Technique, e.g., Comparison
of $\hat{y}_i^j - y_i^j$ is a Supervised Learning.

Now, To measure the performance for all the Nodes, we have

$$\sum_{i=1}^N (\hat{y}_i^j - y_i^j) \dots (2)$$

Note $\hat{y}_i^j - y_i^j \geq 0$ OR $\hat{y}_i^j - y_i^j < 0$

From Eqn (2),

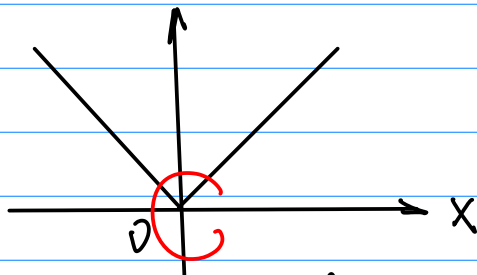
$$\sum_{i=1}^N (\hat{y}_i^j - y_i^j) = (\hat{y}_1^j - y_1^j) + (\hat{y}_2^j - y_2^j) + (\hat{y}_3^j - y_3^j) + \dots + (\hat{y}_N^j - y_N^j)$$

To Address this Problem, we can use Squared Value, such as

$$\sum_{i=1}^N (\hat{y}_i^j - y_i^j)^2 \dots (3)$$

OR, Absolute Value. Consider the "behavior" of Absolute function,

$$f = |x| \dots (4)$$



Now, to Evaluate the performance for All Experiments,

$$\sum_{j=1}^M \sum_{i=1}^N (\hat{y}_i^j - y_i^j)^2 \dots (4)$$

Eqn (4) define A Loss function.

$$L = \sum_{j=1}^M \sum_{i=1}^N (\hat{y}_i^j - y_i^j)^2$$

Now, from the white paper, we have

$$L_{total} = \frac{1}{2} \sum_{j=1}^P (\bar{y}^j - y^j)^2 \dots (23)$$

for a single Output Node.

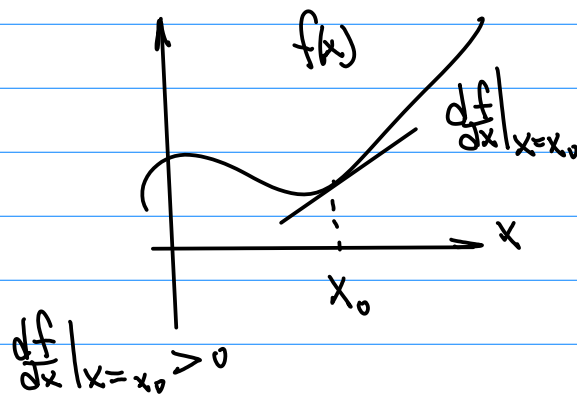
Coefficient " $\frac{1}{2}$ " is constant to allow a better/Simpler manipulations in Gradient Descent Analysis.

Consider the Improvement of A Neural Network Performance. See Eqn (24) from the white Paper.

$$\frac{\partial L}{\partial w_{i,k}} = \frac{\partial}{\partial w_{i,k}} \frac{1}{2} \sum_{j=1}^P \sum_{i=1}^M (\bar{y}_i^j - y_i^j)^2 \dots (24)$$

Background:

Derivative of a function $y = f(x)$.





Feb 14 (Tue).

Homework: Installation of T.F.

Version 2.0 or higher. For the future projects & Homework, Source Code Submission is required with a Stand-Alone Python Code.

Submission of the Installation on CANVAS, in A week, Feb. 21 (Tue).

1° Screen Capture of the Installation.

2° Run the NN Python Code. with minor modification of Stopping Condition Based on the Loss function.

→ 2022S-103c-#nn_sample_2022.py

Base Line Reference code on the github.

Today's Ref. Gradient Descent

2022S-105c-#20-2021S-4gradient-descent-v2-final-2021-2-8.pdf

Example: From the white paper, we have a function denoted as

$f(x_1, x_2, \dots, x_n)$ N-Dimensional

then, define partial derivative as follows

$$\frac{\partial f}{\partial x_i} = \lim_{\delta x_i \rightarrow 0} \frac{f(x_1, \dots, x_i + \delta x_i, \dots, x_n) - f(x_1, \dots, x_i, \dots, x_n)}{\delta x_i} \quad (1)$$

$\frac{\partial f}{\partial x_i}$ Derivative w.r.t x_i (with respect to)

where $i=1, 2, \dots, N$.

f_{x_i}
from Eqn (1),

$$\frac{\partial f}{\partial x_i} \triangleq \lim_{\delta x_i \rightarrow 0} \frac{\Delta f_{x_i}}{\delta x_i} = \lim_{\delta x_i \rightarrow 0}$$

$$\frac{f(x_1, x_2, \dots, x_i + \delta x_i, \dots, x_n) - f(x_1, x_2, \dots, x_n)}{\delta x_i}$$

Now, define a gradient

$$\nabla f = \begin{pmatrix} \frac{\partial f}{\partial x_1} \\ \frac{\partial f}{\partial x_2} \\ \vdots \\ \frac{\partial f}{\partial x_i} \\ \vdots \\ \frac{\partial f}{\partial x_n} \end{pmatrix} \quad \begin{matrix} \frac{\partial f}{\partial x_i} \\ \dots (2) \\ \text{N Dimensional Vector.} \end{matrix}$$

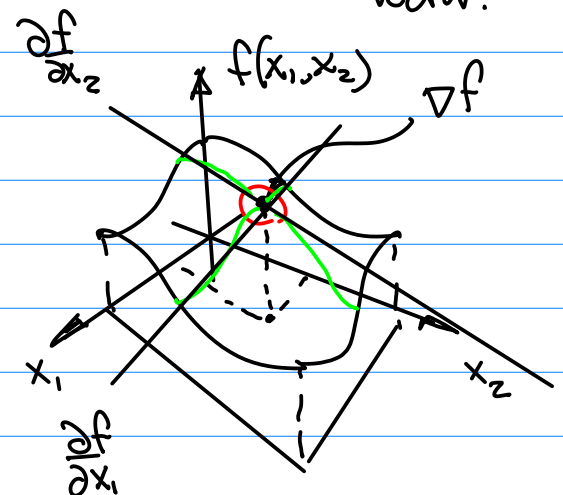


Fig.1

Note, Gradient at the illustrated point provides a better view of the function Behavior.

In Eqn(b), we have

$$f(x_1, x_2) \simeq f(a, b) + \frac{\partial f}{\partial x_1}(x_1 - a) + \frac{\partial f}{\partial x_2}(x_2 - b) \quad (6)$$

Motivation: Use the gradient function to measure the behavior of the Loss function, And to update the Weights ($\{x_i | i=1, \dots, N\}$) in such a way to reduce/minimize the loss function along its fastest descent direction. (Steepest)

The Technique to achieve this is Based on Taylor Expansion Series.

Consider $f(x)$, we can write it in a Standardized way.

$$f(x) = f(x_0) + \frac{df}{dx}(x-x_0) + \frac{1}{2!} \frac{d^2f}{dx^2}(x-x_0)^2 + \frac{1}{3!} \frac{d^3f}{dx^3}(x-x_0)^3 + \dots + R_n(x_0) \quad \dots (3)$$

Higher Order terms.

Basic Building Blocks

$f(x_0)$: Constant, K

$\frac{df}{dx} \Big|_{x=x_0} (x-x_0)$: $A(x-x_0) \rightarrow$

$y = ax + b$

Linear Function.

$\frac{d^2f}{dx^2} \Big|_{x=x_0} (x-x_0)^2$: $A(x-x_0)^2 \rightarrow$

$y = ax^2 + bx + c$

Remark 1: If function $f(x)$ has derivatives upto Order K , then, it can always

be written as a sum of the Basic Building Block, e.g. a Constant term, a Linear term, a quadratic term, etc. Based on the Taylor Series.

Now, Find A Root of the Function.

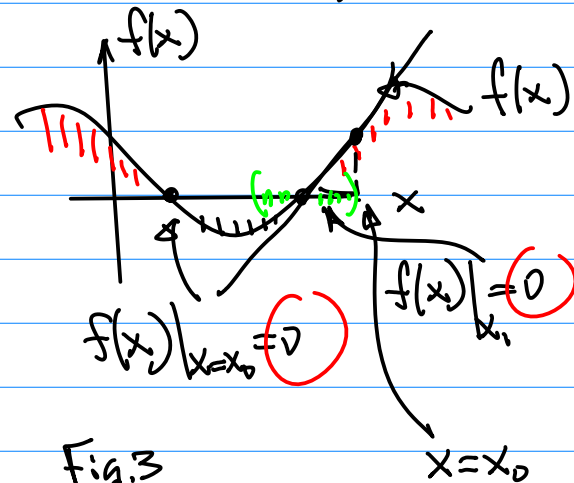


Fig.3

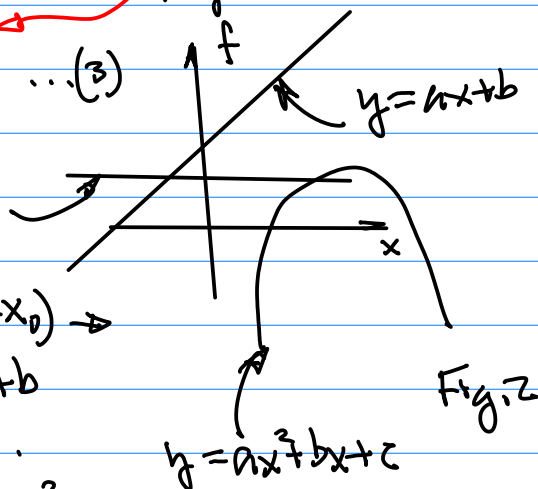


Fig.2

$$f(x) = f(x_0) + \frac{df}{dx}(x-x_0) + \left(\frac{1}{2!} \frac{d^2f}{dx^2}(x-x_0)^2 + \right.$$

$$\left. \frac{1}{3!} \frac{d^3f}{dx^3}(x-x_0)^3 + \dots + R_n(x_0) \right)$$

↓ Removal of $R_n(x_0)$

Approximate

$$f(x) \approx f(x_0) + \frac{df}{dx}(x-x_0) + \left(\frac{1}{2!} \frac{d^2f}{dx^2}(x-x_0)^2 + \right.$$

$$\left. \frac{1}{3!} \frac{d^3f}{dx^3}(x-x_0)^3 + \dots - \frac{1}{(n-1)!} \frac{d^{n-1}f}{dx^{n-1}}(x-x_0)^{n-1} \right)$$

↓

$$f(x) \approx f(x_0) + \frac{df}{dx}(x-x_0)$$

$$f(x) - f(x_0) \approx \frac{df}{dx}(x-x_0)$$

$$\frac{f(x) - f(x_0)}{\frac{df}{dx}} = x - x_0$$

Step
k+1

Step k.