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=UlA3aEk1TnV4bjNLQk5CQkw0dDk4UT09 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: 2° 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

2/1

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

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

Office hours zoom link: Join Zoom Meeting https://us04web.zoom.us/j/9841607683?pwd=UlA3aEk1TnV4bjNLQk5CQkw0dDk4UT09 Meeting ID: 984 160 7683 Passcode: 121092

Course Description

Note6. Pre-requisite CmpE 255 OR CmpE257 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)

Note7: the Book Listed below is a good reference source.

Required Texts/Readings

Textbook

2nd

- 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). for Reference Only.
- *Reference textbook* Learning OpenCV, Computer Vision with the OpenCV Library by Bradski and Kaebler, O'Reilly Publisher, ISBN 978-0-596-51613-0, 2011.

Practical / Hand Book

Other Readings

1. OpenCV on line reference: http://docs.opencv.org/index.html for the future Reference
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

References from the lecture Note

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

Keyword "note"

Other equipment / material requirements

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

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.

Course Requirements and Assignments

SJSU classes are designed such that in order to be successful, it is expected that studen[ts] ... of forty-five hours for each unit of credit (normally three hours per unit per week), incl[uding] participating in course activities, completing assignments, and so on. More details abo[ut]

CmpE258
Spring 2023

3/1

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%

On-Line.

CMPE258 Deep Learning, S2022.

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).
1. Class Ref
2. Honesty Pledge Due this Thursday. Opt.
3. OpenCV & Anaconda Installation Due A week from Today. 1 pt.

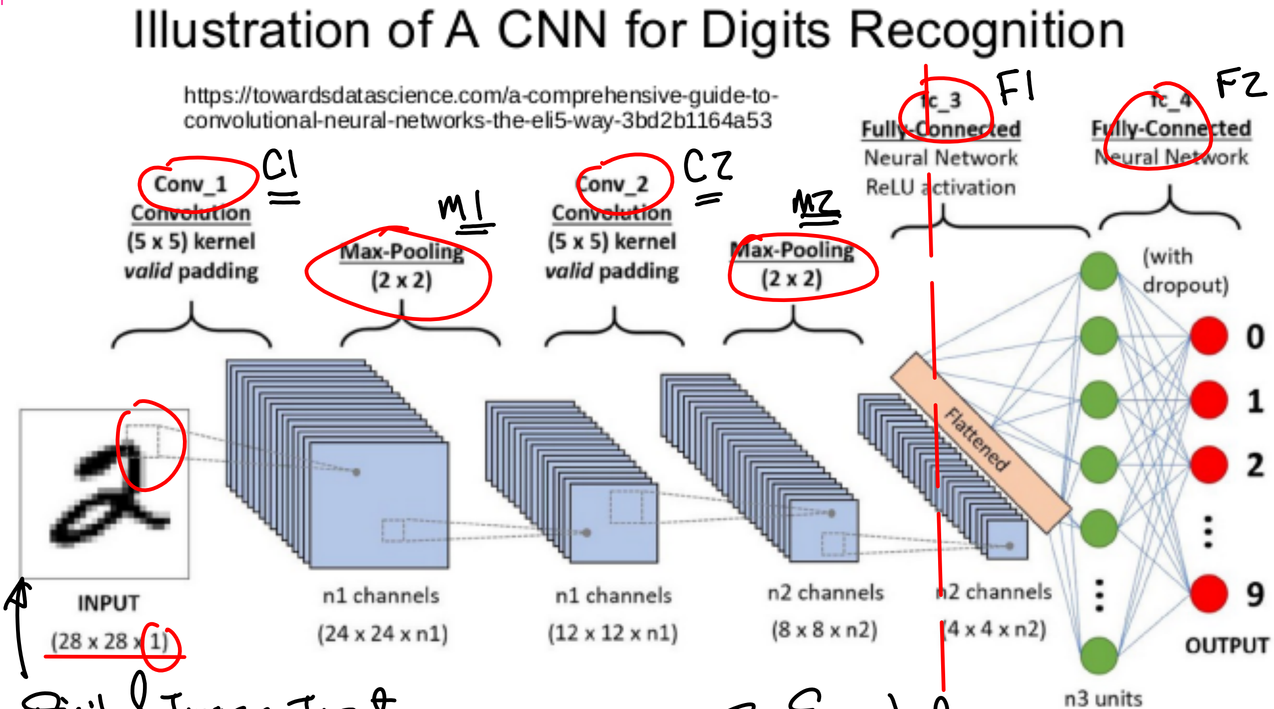
opencv / deep-learning-2022s / **2022F-103-NN-Intro-Python-v5-2022-8-25.pdf**

master · opencv / **deep-learning-2023s** /

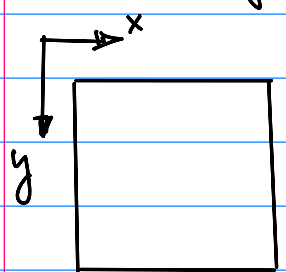
Sample Code: Reads ~~Image~~ An file, and displays it.

..	
2023S-101-Note-cmpe258-2023-01-26.pdf	Add files via upload
2023S-200-canny.py	Add files via upload
accessible-CMPE258-S23-v7-HarryLi.pdf	Add files via upload

Example:



1. Digital Image Input.



I(x,y)
A frame of Image.

M x N Resolution

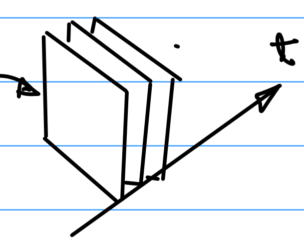
No. of Col. Per Row.
No. of Rows per frame

1920 x 1280
M N

Pixel: picture element

for a Video Clip, OR Video Stream.

I(x,y;t)



2. Example for 28x28x1,

8 bit Gray scale image

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

Darkest, Black   Brightest, White

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

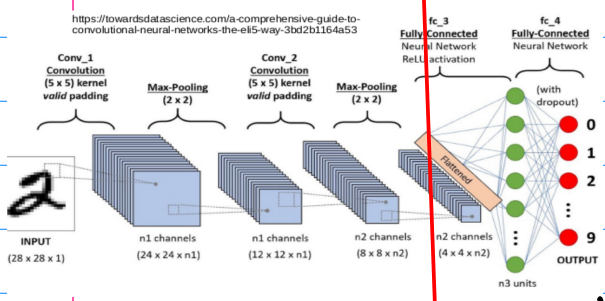
4. "Conv_1" → C → C1
Simplified Notation.   Adding An Index to Label the Convolution Block

"Max-Pooling" → M → M1

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

C1 M1 C2 M2

Cmpe258
Spring23



Feature Extraction Decision Making.

Fully Connected (Feed Forward) Dense NN → F
F1

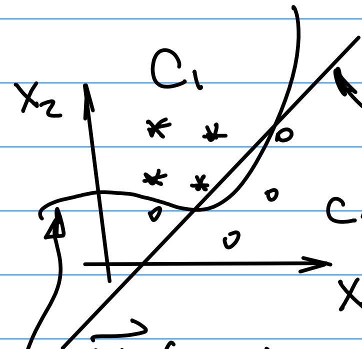
And then the 2nd Block: F2.

Conclussion: The Architecture is defined as C1M1C2M2F1F2

Feb 2nd (Thu).
Note: 1° Attendance, please use Chat message to text me privately
Your First-Last Name, and 4-Digits SID.
2° Introduction & Team Formation.
Ref: from the class github.
1° PPT
2022F-103b-NN-Intro-Python-v5-2022-8-25.pdf
2° Lecture Notes (Last Semester)
2022F-101-cmpe258-not e-2022-11-1.pdf
3° Whitepaper 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:
C1M1_C2M2_C3M3.



Decision-making Function1.

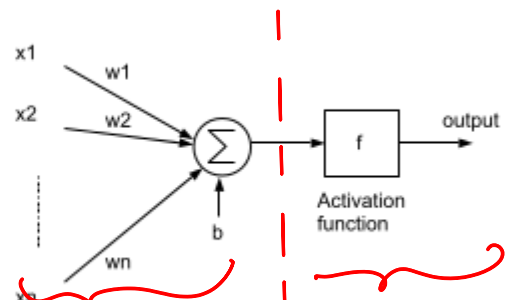
$\vec{X} = (x_1, x_2)$

Decision making Function 2

1st Dimension Area/Size of A traffic Sign

2nd Dimension. Color of the traffic Sign.

Consider A Single Neuron Below:



Input Output

Note: 1° Input / Excitation
$X = (x_1, x_2, \cdots, x_n)$ ... (1)
2° Weights, Links to Allow X to Connect to A Neuron.
$W = (w_1, w_2, \cdots, w_n)$ ... (2)
$w_i \in [0, 1]$

3° Combining All the Inputs

$$x_1w_1 + x_2w_2 + \cdots + x_nw_n \quad \ldots(3)$$

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

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

and its vector form is

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

or simply denoted as X.

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

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

III. Notation for Weights

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

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

and its vector form is

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

or simply denoted as W.

And

$$x_1w_1 + x_2w_2 + \cdots + x_nw_n = \sum_{i=1}^{N} x_iw_i \quad \ldots(3\text{-}1)$$

And

$$\sum_{i=1}^{N} w_ix_i = w_1x_1 + w_2x_2 + \ldots w_Nx_N \quad (9)$$

Or simply in a short hand vector form notation:

$$\sum_{i=1}^{N} w_ix_i = W \cdot X. \quad (10)$$

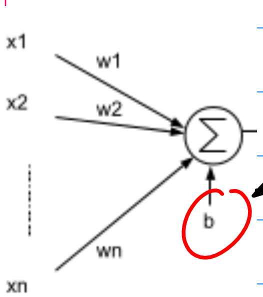
Vector Dot-Product

$$(x_1, x_2)\cdot(w_1, w_2) = x_1w_1 + x_2w_2$$

V. A Transfer Function

A transfer function is defined as

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



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

$$\sum_{i=1}^{N} x_iw_i \text{ v.s. } \sum_{i=1}^{N} x_iw_i + b$$

Offset

4. Define A Transfer function

$$h = \sum_{i=1}^{N} x_iw_i + b \text{ OR } \quad \ldots(4)$$

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

$$\text{And } W_{N+1} = 1 \quad \ldots(4\text{-}1)$$

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

5. Activation Function.

Denoted as "f"

$$y = f \quad \ldots(5)$$

Output ↓

$$f(x_i, w_i) \quad \ldots(5\text{-}1)$$

↓

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

$$f(h(x_i, w_i; b)) \quad \ldots(5\text{-}3)$$

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

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

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

Or simply written as

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

6. Output from a Neuron y

Output from a Neuron, if from an Experiment.

$\hat{y}$

$\downarrow$

Suppose it is from the experiment j

$j=1,2,\dots,M$

$\hat{y}^j$ ← Superscript.

Feb 7 (Tue).

Note: 1° github Ref.

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

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

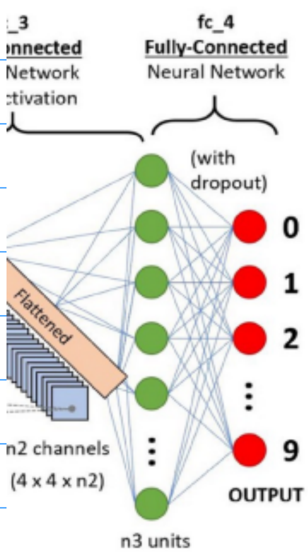
2022F-101-cmpe258-note-2022-11-1.pdf Notes

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 $j=1,2,\dots,N$

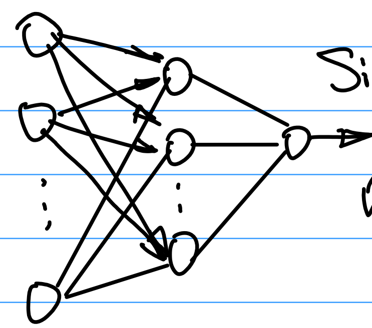
$$\hat{y}_i^j \big|_{j=1} = \hat{y}_i$$

Now, consider the performance of a Convolution Neural Network

Comparison of the Network Output $\hat{y}$ to the "Ground Truth" y

$\downarrow$

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



Single Output

$\hat{y}-y$ ... (1)

Fig. 2

For more than One Nodes, in Fig. 1.

$\hat{y}_i - y_i$ ... (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 the 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

$$\sum_{i=1}^{N} \left(\hat{y}_i^j - y_i^j\right) \quad \ldots (2)$$

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

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

From Eqn (2),

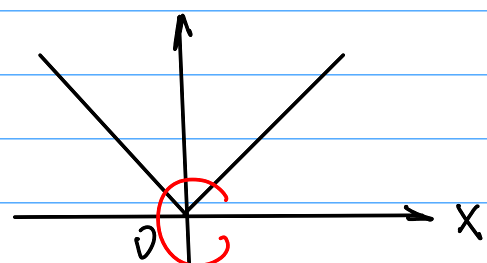
$$\sum_{i=1}^{N} \left(\hat{y}_i^j - y_i^j\right) = \left(\hat{y}_1^j - y_1^j\right) + \left(\hat{y}_2^j - y_2^j\right) + \left(\hat{y}_3^j - y_3^j\right) + \cdots + \left(\hat{y}_N^j - y_N^j\right)$$

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

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

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

$$f = |x| \quad \ldots (4)$$



Now, to Evaluate the performance for All Experiments,

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

Eqn (4) define A Loss function.

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

Now, from the white paper, we have

$$L_{total} = \frac{1}{2} \sum_{j=1}^{P} (\tilde{y}^j - y^j)^2. \quad (23)$$

for a single Output Node.

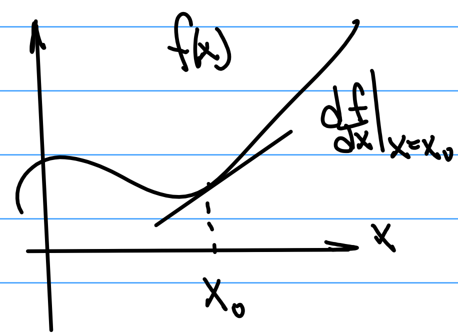
Coefficient "1/2" is constant to allow a better/Simpler manipulations in Gradian Descent Analysis.

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

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

Background:

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



$\left.\frac{df}{dx}\right|_{x=x_0} > 0$

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

$f(x_1, x_2, \cdots, x_N)$ N-Dimensional

then, define partial derivative as follows

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

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

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

f_{x_i}

from Eqn (1),

$$\frac{\partial f}{\partial x_i} \triangleq \lim_{\partial x_i \to 0} \frac{\Delta f_{x_i}}{\Delta x_i} = \lim_{\delta x_i \to 0} \frac{f(x_1, x_2, \cdots \delta x_i, \cdots x_n) - f(x_1, x_2, \cdots, x_n)}{\delta x_i}$$

Now, define a gradient

$$\nabla f = \begin{pmatrix} \frac{\partial f}{\partial x_1} \\ \frac{\partial f}{\partial x_2} \\ \cdots \\ \frac{\partial f}{\partial x_i} \\ \cdots \\ \frac{\partial f}{\partial x_n} \end{pmatrix} \cdots (2)$$

$\frac{\partial f}{\partial x_i}$

N Dimensional Vector.

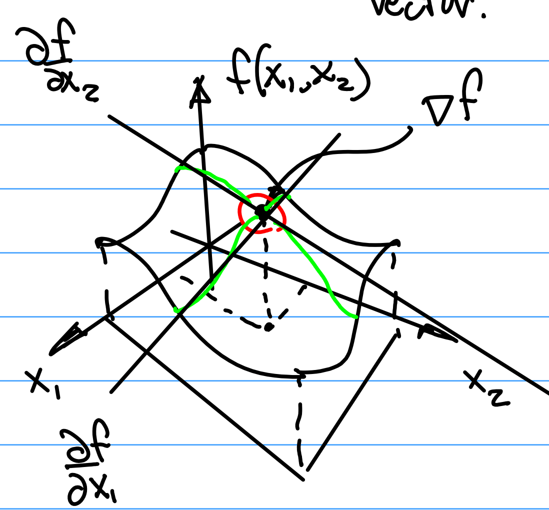


Fig. 1

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

CMPE258

Spring 2023

In Eqn(6), 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 ($\{x_i \mid i=1,\cdots,N\}$) 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.

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

higher Order terms.

Basic Building Blocks

$f(x_0)$: Constant, K

$\left.\frac{df}{dx}\right|_{x=x_0}(x-x_0)$: $A(x-x_0) \rightarrow y = ax+b$

Linear Function.

$\left.\frac{d^2f}{dx^2}\right|_{x=x_0}(x-x_0)^2$: $A(x-x_0)^2 \rightarrow y = ax^2+bx+c$

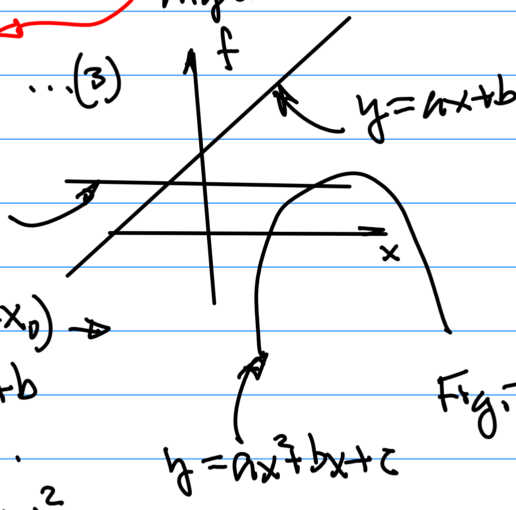


Fig.2

Remark 1: If function f(x) has derivatives up to 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

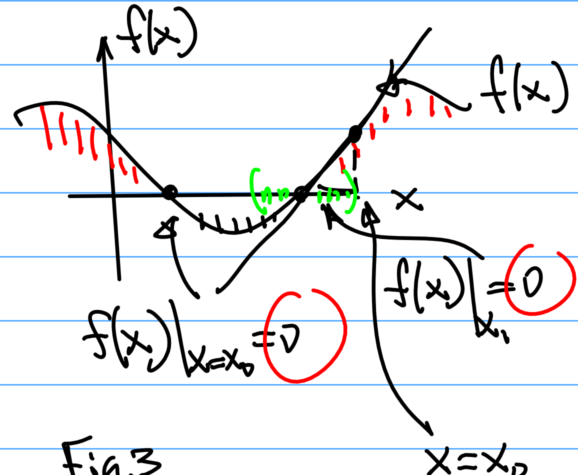


Fig.3

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

↓ Removal of $R_n(x_0)$

Approximate

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

↓

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

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

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

Step $k+1$ (x), Step k (x_0).

$$f(x,y)=f(x_0,y_0)+\frac{1}{1!}\frac{\partial f}{\partial x_1}(x-x_1)+\frac{1}{1!}\frac{\partial f}{\partial x_2}(x-x_2)+\dots \quad \dots(1)$$

just the Linear term,

$$f(x,y)\cong f(x_0,y_0)+\frac{1}{1!}\frac{\partial f}{\partial x_1}(x-x_0)+\frac{1}{1!}\frac{\partial f}{\partial x_2}(y-y_0)$$

$$=f(x_0,y_0)+\frac{\partial f}{\partial x_1}(x-x_0)+\frac{\partial f}{\partial x_2}(y-y_0)$$

Let $x_0=a$, $y_0=b$. Hence,

$$f(x,y)\simeq f(a,b)+f_{x_1}(x-a)+f_{x_2}(y-b) \quad \dots(2)$$

from the definition of a gradient, we have

$$\nabla f(x,y)=\begin{pmatrix}\frac{\partial f}{\partial x}\\ \frac{\partial f}{\partial y}\end{pmatrix}=\begin{pmatrix}f_x\\ f_y\end{pmatrix} \quad \dots(3)$$

Feb 16 (Thu).

Example: Continuation. Now Consider N-Dimensional Case for Taylor Expansion.

$f(x_1,x_2,\dots,x_n)$ take $n=2$.

without Lose of the generality.

$f(x_1,x_2)$ Then,

$$f(x,y)-f(a,b)\simeq(x-a,y-b)\cdot\begin{pmatrix}f_x\\ f_y\end{pmatrix} \quad \dots(3\text{-}b)$$

From the white paper, let's make

$$\Delta x_1=x_1-a=-f_{x_1},\ \Delta x_2=x_2-b=-f_{x_2} \qquad (11)$$

Therefore, substitute $\Delta x,\Delta y$ $(\Delta x_1,\Delta x_2)$ into Eqn (3-b)

$$f(x,y)-f(a,b)\simeq(-f_x,-f_y)\cdot\begin{pmatrix}f_x\\ f_y\end{pmatrix}$$

$f(x,y) - f(a,b) \simeq -(f_x^2 + f_y^2)$ Discussion on Activation function $f(\cdot)$

Since

$$f_x^2 + f_y^2 \geq 0$$

we have

$$-(f_x^2 + f_y^2) \leq 0 \quad \cdots (4)$$

which leads to

$$f(x,y) - f(a,b) \leq 0$$

or

$$f(x,y) \leq f(a,b)$$

Note: $\cdots (5)$

$$f(x,y) \simeq f(a,b) + f_{x_1}(x-a) + f_{x_2}(y-b)$$

$f(x,y)$ to Be minimized

(x,y) Next Step

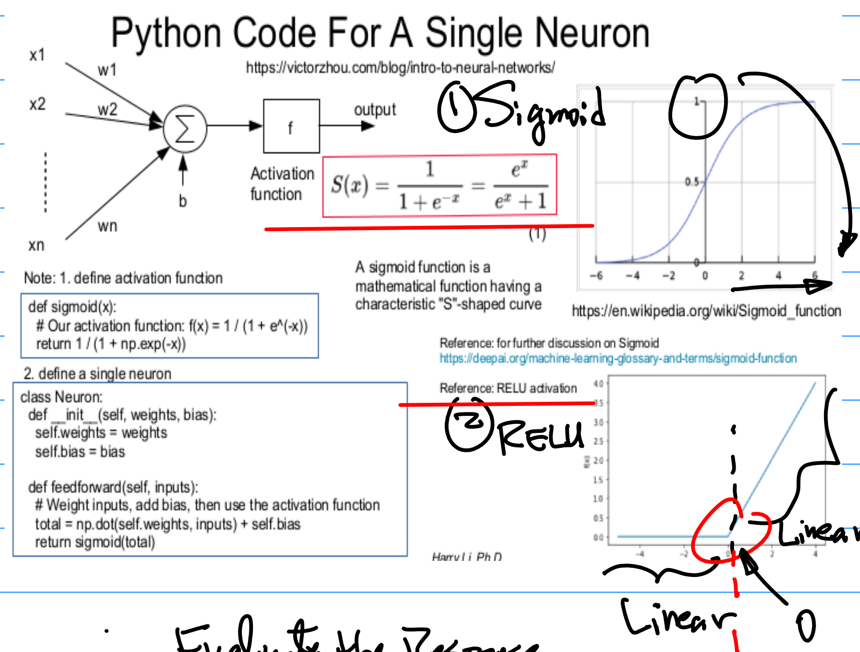
Current Step

$x-a$, $y-b$

Δx Δy

$f(a,b)$ Current Loss

To minimize $f(x,y)$, we will have Choose a better way to update the next step (x,y), hence, the Negative gradient.



Evaluate the Response of A Activation function, Suppose we have $h(w_i, x_i; b) = 0.5$. Find Response of the Sigmoid, and the RELU

Sol Since the Sigmoid $S(x) = \dfrac{1}{1+e^{-x}}$

we have $x = h(w_i, x_i; b) = 0.5 \quad \cdots (6)$

so

$$S(0.5) = \frac{1}{1+e^{-x}}\bigg|_{x=0.5} = \frac{1}{1+e^{-0.5}}$$

And for the RELU,

$$f(x) = \begin{cases} ax & \text{for } x \geq 0 \\ 0 & \text{o/w} \end{cases} \quad \cdots (7)$$

$$f(x_1, x_2) - f(a,b) = (\Delta x_1, \Delta x_2)\nabla f = -(f_{x_1}^2 + f_{x_1}^2) \quad (12)$$

Leads to the conclusion of minimization of the Loss function By using Negative gradient.

III. GRADIENT EXAMPLE

Example:

Example 1. Given

$$y = f(x_1, x_2) = x_1^2 + x_1 x_2$$

Find its gradiant as follows [1]:

$$\nabla f = \begin{pmatrix} \frac{\partial f}{\partial x_1} \\ \frac{\partial f}{\partial x_2} \end{pmatrix} = \begin{pmatrix} 2x_1 + x_2 \\ x_1 \end{pmatrix}$$

$$\frac{\partial f}{\partial x_1} = \frac{\partial}{\partial x_1}(x_1^2 + x_1 x_2) = 2x_1 + x_2$$

$$\frac{\partial f}{\partial x_2} = \frac{\partial}{\partial x_2}(x_1^2 + x_1 x_2) = 0 + x_1$$

$f(x) = f(x)\Big|_{x=h(w_i, x_i;\ b)=0.5} = f(0.5)$

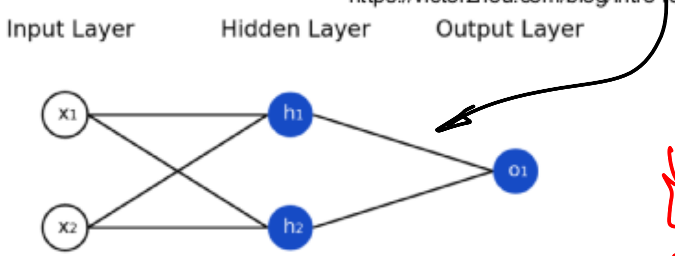
$= a \times 0.5 = 0.5a$

Example: Python Code for Simple Multi-Layer Feed forward NN.

Note: 1° Derivate, gradient Computation are Carried out by Finit Difference Approach, But in this Sample code, we are using a close form solution. see below

```python
def deriv_sigmoid(x):
  # Derivative of sigmoid: f'(x) = f(x) * (1 - f(x))
  fx = sigmoid(x)
  return fx * (1 - fx)
```

2° Write Python Code for NN like the following




```python
def __init__(self):
  weights = np.array([0, 1])
  bias = 0

  # The Neuron class here is from the previous section
  self.h1 = Neuron(weights, bias)
  self.h2 = Neuron(weights, bias)
  self.o1 = Neuron(weights, bias)

def feedforward(self, x):
  out_h1 = self.h1.feedforward(x)
  out_h2 = self.h2.feedforward(x)

  # The inputs for o1 are the outputs from h1 and h2
  out_o1 = self.o1.feedforward(np.array([out_h1, out_h2]))

  return out_o1

network = OurNeuralNetwork()
x = np.array([2, 3])
print(network.feedforward(x))
```

3° The code for MSE (Mean Square Error) Loss function.

$$\text{MSE} = \frac{1}{n}\sum_{i=1}^{n}(y_{true} - y_{pred})^2$$

The Mean Square Error Function as a very often it is also defined as objectiv minimize the loss function becomes ob as shown below step by step.

$$y_{true} - y_{pred}$$

$$\downarrow$$

$$(y_{true} - y_{pred})^2$$

$$\downarrow$$

$$\sum_{i=1}^{n}(y_{true} - y_{pred})^2 \longrightarrow \text{Exp}\left[\sum_{i=1}^{n}(y_{true} - y_{pred})^2\right]$$

```python
def mse_loss(y_true, y_pred):
  # y_true and y_pred are numpy arrays of the same length.
  return ((y_true - y_pred) ** 2).mean()
```

CMPE258
Spring 2023

Homework: Due A week from Today.
Feb. 23 (Thu).
CANVAS.

1° DownLoad the Sample code from the github. Modify and Run the code

2022S-103c-#nn_sample_2022.py

Check the convergence under different Learning Rate.

2° 2022S-102b-homework2-building-blocks-gradient-2022-3-1.pdf

CMPE258
A Single Neuron Basic Building Blocks and Gradient Descent Function
Homework
HL

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

Figure 1.

1. Given the equation in Figure 1, design by drawing a single neural, for N=3, and w1=0.3, w2=0.9, w3=0.83, suppose the bias b = 0.1.
2. Based on the equation in Figure 1, explain what is the function h(.), based on the parameters in Question 1 (above), for x1=0.1, x2=14, x3=-7.5, find h=?
3. Suppose we choose the following function for activation function f, find the output of the neuron based on the equation in Figure 1, with the parameters in Question 1 and 2.

3° Update your OpenCV Code to Display Video from a Live WebCAM and from a file.