CMPE258 Spring2022

1/

Feb1st. Organizational meeting.

1. Today's Topics "Greensheet"

2022S-100-accessible-CMPE258-S22-v5-HarryLi.pdf

2022S-104-Python-OpenCV-Anaconda-v2-HL-YY-2022-...

readme

Naming Convention Yr+Semester+ID +Name+Date

Contact Information: E-mail: hua.li@sjsu.edu

Text message to (650)400-1116.

Office Hours: M.W. 4:30-5:30pm.

Zoom (link to be shared in the email)
Join from PC, Mac, Linux, iOS or Android: https://sjsu.zoom.us/j/85616325978?pwd=MzlRbDJXVHBDQ2g1U0RPM2tYc045Zz09

Password: 451032

On-Line materials on github
https://github.com/hualili/opencv/tree/master/deep-learning-2020S
https://github.com/hualili/opencv/tree/master/deep-learning-2022s

Also, CANVAS → mostly for Assignments and projects.

All Assignment/Projects are posted on Both github & SJSU CANVAS.

Lecture Material consists of P.P.T. posted on github, and Lecture Notes (White-Board Written Notes)

Core Emphases of the Class: Deep Convolutional Neural Networks, And their Application in Image Analysis, Video Analysis.

1. Text Book:
https://github.com/hualili/opencv/blob/master
IP120-AI-DL/2018F/2018F-6-DeepLearningCh02.pdf

2. Computer Vision Book By Horn as a reference for Convolution & Image Segmentation, Contours Analysis (Binary Image).

3. OpenCV Reference Book (2nd Edition) together with On-Line Document (OpenCV).

Note: OpenGL (GL: Graphics Library) is just for Reference purpose, no need for this class. (But maybe helpful for the future research).

Unity is game Development platform, interactive 3D Graphics Design platform.

Programming Languages:
1. Python. 3.6 or 3.7
2. C/C++

Homework (Due A week from today) Feb.8th.
No Submission. Submit A Screen

Capture that shows OpenCV installed successfully, with Jpg or png file with Naming of the file as follows:

FirstName_LastName_SID_OpenCV.jpg

This Homework will be posted on CANVAS, Submission is on CANVAS.

Homework, Installation of Tensor Flow, Due 2 weeks Feb 15th.

Submission: Screen Capture that shows the installation is successful. Submission on CANVAS.

Submit jpg, png file with the Naming convention as follows:

FirstName_LastName_SID_TF.jpg

Note: Optional, for Edge AI Computing, Consider using NVDA Jetson NANO (4GB) version.

5% Bonus

Grading:

Homework, Projects : 30%
5% 25%

Project 1. Computer Vision for Preprocessing. plus Deep Convolutional Neural Nets To give Real Time Detection Result of Last 4 Digits of a Student ID.

10%.

Project 2. "Semester Long" project, with technical requirements (List) Team project. 4 person Team.

Each person has clear Definition of the tasks (Programming/Coding) And Balanced Contribution.

Final PPT, Demo Presentation

15%.

Midterm Exam: 30%.

Need to use your Laptop Computer, to Run/Execute code, modify the Code.

Final 40%

Introduction

Topics { Neural Networks formulation (Basic Building Blocks)
Digital Images/Videos.

Example: A Single Neuron Formulation (Some kind Brain Cell)

Step 1. Summation function.

Fig 1.

"Σ" Summation function.

Note: $\sum_{k=1}^{N} x_k = x_1 + x_2 + \cdots + x_N$

Step 2. Inputs

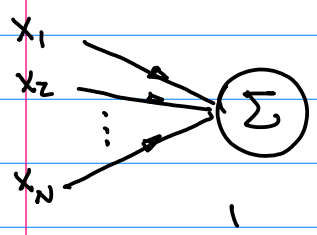


Fig 2.

Step 3. Weights (Knowledge)

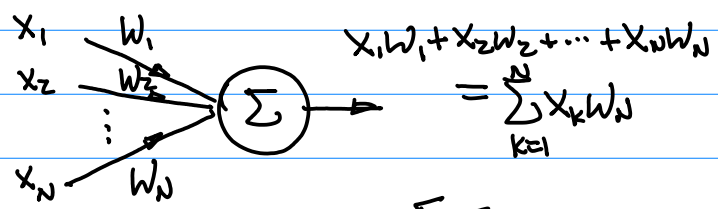


Fig 3.

$$\sum_{k=1}^{N} x_kW_N = x_1W_1 + x_2W_2 + \cdots + x_NW_N \quad \cdots (1)$$

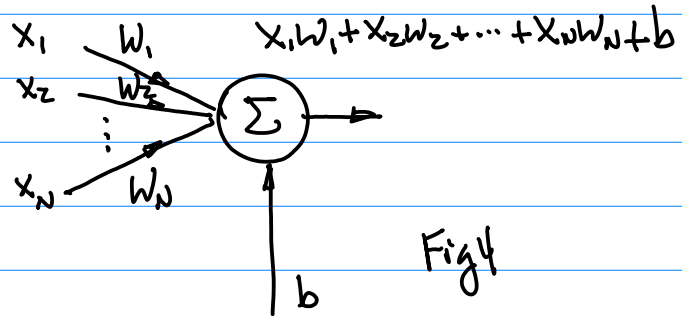


Fig 4

$$\sum_{k=1}^{N} x_kW_N = x_1W_1 + x_2W_2 + \cdots + x_NW_N + b \quad \cdots (1b)$$

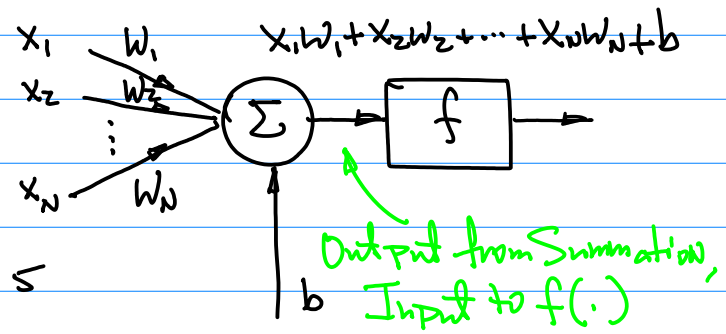


Fig 5

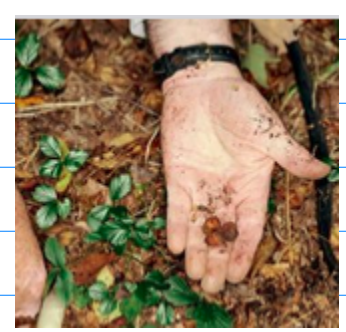
Note: Activation function f, denoted as f(·) (A function of Independent Variable "·", or A function of Input "·")

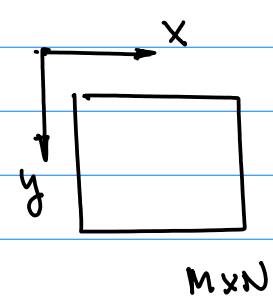
$$f(\cdot) = f(x_1W_1 + x_2W_2 + \cdots + x_NW_N + b)$$

$$= f\left(\sum_{k=1}^{N} W_k x_k + b\right) \quad \cdots (2)$$

Summary: The output of a Single is given by Eqn (2). where Activation function f(·) Can take different forms, it affects the Learning, Learning Speed.

Example: Digital Image, I(x,y)





$I(x,y)$

(x,y) Location of A picture element, "pixel"

Intensity, And/OR Color of An Image

In Case of a Single pixel, (x,y) is the Location of this pixel, I is Color/Intensity of the pixel

Note: For An Image $I(x,y)$ its features include
- Resolution $M \times N$ (No. of pixels per Row; No. of Rows Per frame)
- Pixel Depth: bpp (Bit per pixel)

$I(x,y)_{M \times N}$ OR $I(x,y)$

1024×768

pixels/Row — Rows

For A color Image, A pixel depth very often is equal to 24 (bpp)

r, g, b Primitive color of red (r), green (g), blue (b) has 8 bits quantization level, e.g.

$r: [0,255]$, $g: [0,255]$, $b: [0,255]$.

Feb 8th (Tue)

Today's Topics: 1° Introduction, Basic Building Blocks, Math Formulation.

2° Simple Python Code for OpenCV.

Example: Notations & Formulation

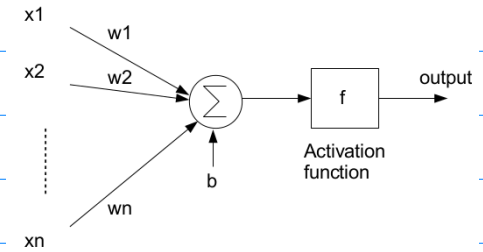


Fig. 1

1. Notation for Input $x_i, i=1,2,\dots,N$

$\{x_i \mid i=1,2,\dots,N\}$ ... (1)

Vector form,

$(x_1, x_2, \dots, x_N)$ ... (1b) $\to X$

Introduce Superscript j for Experiment j

Input x_i^j, $i=1,2,\dots,N$; $j=1,2,\dots,P$

Hence Eqn (1) Becomes

$\{x_i^j \mid i=1,2,\dots,N; j=1,2,\dots,P\}$

$(x_1^j, x_2^j, \dots, x_N^j)$ for Experiment j

2. Notation for Weight

w_i, for $i=1,2,\dots,N$

hence,

$(w_1, w_2, \dots, w_N)$ ... (2) $\to W$

3. Inputs & Weights

x_i w_i

$w_i x_i$ for $i=1,2,\cdots,N$

$(x_1, x_2, \cdots, x_N)\cdot(w_1, w_2, \cdots, w_N)$

$= w_1x_1 + w_2x_2 + \cdots + w_Nx_N = \sum_{i=1}^{N} w_i x_i$

$(w_1, w_2, \cdots, w_N)\cdot(x_1, x_2, \cdots, x_N)$

$= w_1x_1 + w_2x_2 + \cdots + w_Nx_N = \sum_{i=1}^{N} w_i x_i$

4. Transfer Function, Denoted as

h

OR $h(\cdot)$

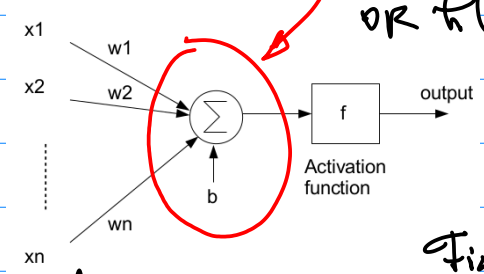


Fig. 2

$$h = \sum_{i=1}^{N} w_i x_i + b = W\cdot X + b \quad \cdots (3)$$

Offset ("Bias")

h, or $h(\cdot)$, or Eq (3), or

$h(w_i; b)$, or $h(w_i)$

5. Activation Function. f

Acts Like a Switch, ON/OFF &

Attenuate the Output

f, $f\left(\sum_{i=1}^{N} w_i x_i + b\right)$, OR

$f(h(w_i, b))$, OR $f(h(\cdot))$

The output of A Neuron is denoted as

y, and $y = f\left(\sum_{i=1}^{N} w_i x_i + b\right)$

$= f(h(w_i; b)) \quad \cdots (4)$

5. Outputs for A Neural Network

y_k, for $k=1,2,\cdots,Q$

if $Q=1$, y_1, also better to just y_1 for a single Neuron as y

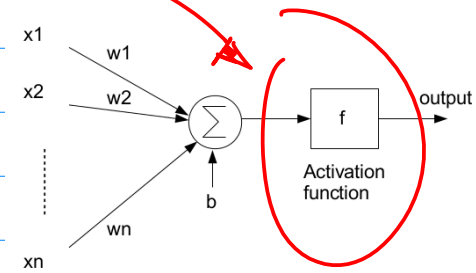
Question: What output? whose output?

Output from the Neuron (OR Network)

$\tilde{y}$ (Tilde)

Ground Truth is denoted as y

6. Loss function, Objective Function, OR difference



"Supervised Learning"

For Experiment j (multiple experiments)

$$\tilde{y}^j - y^j \quad , \quad j=1,2,\dots,p \quad \dots (5)$$

Output from the NN (Neuron or Network)

Ground Truth for the experiment

for $j=1$, $\tilde{y}^1 - y^1$

$j=2$, $\tilde{y}^2 - y^2$

$\vdots$

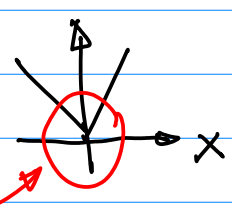
$j=p$, $\tilde{y}^p - y^p$

Therefore, put all these differences (Loss) together

$$(\tilde{y}^1 - y^1) + (\tilde{y}^2 - y^2) + \dots + (\tilde{y}^p - y^p)$$

$$= \sum_{k=1}^{p} (\tilde{y}^k - y^k) \quad \dots (6)$$

Note:



$y = |x|$

To deal with the issue of Absolute value of a function, Let's square it.

Hence Eqn (6) becomes

$$\sum_{k=1}^{p} (\tilde{y}^k - y^k)^2 \quad \dots (6b)$$

7. Objective function

$$L \triangleq \sum_{k=1}^{p} (\tilde{y}^k - y^k)^2 \quad \dots (7)$$

OR,

$$L(w_i) \triangleq \sum_{k=1}^{p} (\tilde{y}^k - y^k) \quad \dots (7b)$$

$$= \sum_{k=1}^{p} (\tilde{y}^k(f(w_i)) - y^k) \quad \dots (7c)$$

8. To generalize the result in Eqn (7c) for multiple output, we have the following formulation.

From Eqn (7)

$$\sum_{k=1}^{p} (\tilde{y}^k_{k_2} - y^k_{k_2})^2$$

k_2: Output k_2 for multiple output

y_j, $j=1,2,\dots,k_2,k_2+1,\dots,M$

We can count All outputs loss.

$$\sum_{k=1}^{p} (\tilde{y}^k_1 - y^k_1), \quad \sum_{k=1}^{p} (\tilde{y}^k_2 - y^k_2), \quad \dots$$

$$\dots \sum_{k=1}^{p} (\tilde{y}^k_M - y^k_M)$$

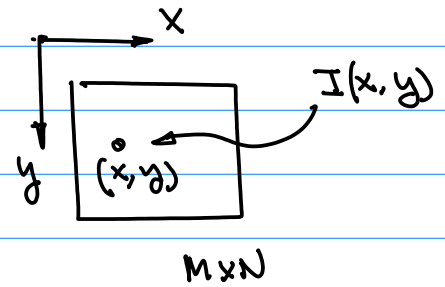
Hence

$$\sum_{k_2=1}^{M} \sum_{k_1=1}^{p} (\tilde{y}^k_{k_2} - y^k_{k_2})^2 \quad \dots (8)$$

CMPE258 Feb8,22

$$L(w_i) \triangleq \frac{1}{2} \sum_{k_2=1}^{m} \sum_{k_1=1}^{p} \left(\tilde{y}_{k_2}^{k_1} - y_{k_2}^{k_1} \right)^2 \quad \ldots (8b)$$

Example: Digital Image $I(x,y)$



Installation of OpenCV → Python OR (C/C++, But Python is Better)

Different Packages for ML/DL, may Need different Version of Python, And different Packages

Anaconda is a well developed, Adopted tool for Package management.

1. Check github class Reference

2022S-104a-Python-OpenCV-Anaconda-v2-HL-YY-2022..

Note: For installation of Anaconda, Check github document, 2022S-104-~

Once Anaconda is installed, then Let's take a look the configuration for

Tensorflow & OpenCV Environment

Step 1.

Running O

Step 1. Create configuration file .yml for CPU and GPU version

conda-cpu.yml

```
name: yolov4-cpu

dependencies:
  - python==3.7
  - pip
  - matplotlib
  - opencv
  - pip:
    - opencv-python==4.1.1.26
    - lxml
    - tqdm
    - tensorflow==2.3.0rc0
    - absl-py
    - easydict
    - pillow
```

Name of the environment

17 lines (16 sloc) | 269 Bytes

```
name: yolov4-gpu

dependencies:
  - python==3.7
  - pip
  - matplotlib
  - opencv
  - cudnn
  - cudatoolkit==10.1.243
  - pip:
    - tensorflow-gpu==2.3.0rc0
    - opencv-python==4.1.1.26
    - lxml
    - tqdm
    - absl-py
    - easydict
    - pillow
```

Step 2. Create the environment

Step 2. Configure/create the environment in the folder you will run your OpenCV program by

```
$conda env create -f conda-cpu.yml      #for CPU
$conda env create -f conda-gpu.yml      #for GPU
```

file Name to Be used

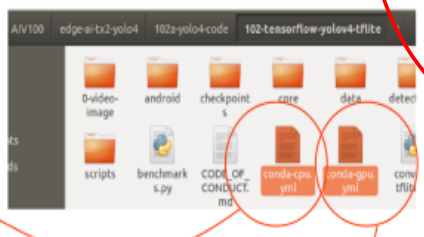
CMPE258 Feb8

Step3 Activate the enviroment

Step4. Run Your Python Code (OpenCV or T.F.)

Running OpenCV with Conda Environment

Step 1. Create configuration file .yml for CPU and GPU version



conda-cpu.yml

```yolo
name: yolov4-cpu

dependencies:
  - python==3.7
  - pip
  - matplotlib
  - opencv
  - pip:
    - opencv-python==4.1.1.26
    - lxml
    - tqdm
    - tensorflow==2.3.0rc0
    - absl-py
    - easydict
    - pillow
```

conda-gpu.yml

```yolo
name: yolov4-gpu

dependencies:
  - python==3.7
  - pip
  - matplotlib
  - opencv
  - cudnn
  - cudatoolkit==10.1.243
  - pip:
    - tensorflow-gpu==2.3.0rc0
    - opencv-python==4.1.1.26
    - lxml
    - tqdm
    - absl-py
    - easydict
    - pillow
```

Step 2. Configure/create the environment in the folder you will run your openCV program by

```
$conda env create -f conda-cpu.yml      #for CPU
$conda env create -f conda-gpu.yml      #for GPU
```

Step 3. Activate the environment with the name that you have created by

```
$ conda activate yolov4-cpu  #for CPU
$ conda activate yolov4-gpu  #for GPU
```

Name of the environment in the .yml file

```
V100/edge-ai-tx2-yolo4/102a-yolo4-code/102-tensorflow-yolov4-tflite$ conda activate yolov4-gpu
(yolov4-gpu) harry@workstation:/media/harry/easystore/backup-2020-2-15/CTI/3proejcts/3-8-smart-
V100/edge-ai-tx2-yolo4/102a-yolo4-code/102-tensorflow-yolov4-tflite$
```

conda-cpu.yml

```yolo
name: yolov4-cpu

dependencies:
  - python==3.7
  - pip
  - matplotlib
  - opencv
  - pip:
    - opencv-python==4.1.1.26
    - lxml
    - tqdm
    - tensorflow==2.3.0rc0
    - absl-py
    - easydict
    - pillow
```

Name of the environment to be created

Step 4. Now run your Python OpenCV program

```
$python myOpenCV.py image.png
```

Note: Install Anaconda → Create the environment for T.F. & OpenCV

Execute OpenCV ← Activate the environment

Sample Code (From the class github)

2022S-104d-#2-pdisplay-2019-1-30.py

a. Program Header ← Name, Program Name, Version, Status (Debug, Release), Date. Note

b. cv2 OpenCV

Caption of the window

```
"""
pdisplay.py
Demo read and display image
"""
import sys
import cv2
import numpy as np

#main(sys.argv[1:])
window_name = 'Display Image'

imageName = sys.argv[1] #get file name from command line

src = cv2.imread(imageName, cv2.IMREAD_COLOR)

if src is None:
    print ('Error opening image!')
    print ('Usage: pdisplay.py image_name\n')
```

CMPE258 Feb8,22

2nd Ref on the github

2022S-103a-notation-neuro-loss-function-2022-2-8-1.pdf

```
while True:
    cv2.imshow(window_name, src)

    c = cv2.waitKey(500)
    if c == 27:  #ESC
        break

    ind += 1
```

③

Keyboard input to exit.

Note: These Python functions are required. (memorize them!)

① import cv2 ② cv2.imread( )

③ cv2.imshow( ).

Homework (Due A week from today) Feb 15

1° Installation of OpenCV.

2° Installation of Anaconda.

3° Use your Smartphone to take photo, and Save it for OpenCV Program

4° Write A Python Program (Ref. Code from the class github is ok) to Display:

a. Your Name + SID (4 Digits)

b. Your Smartphone picture.

5° Submission: One pdf file And Zipped. On CANVAS.

Feb 15

Ref: 1° 2022S-103a-Notation

2° 2022S-103C

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

VIII. Minimize the Loss Function

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

Now, consider to Optimize Neural Network Performance By minimizing the Loss function.

Mathematical Background

a. Derivative(s) → b. Partial Derivatives

↓

c. Gradient

↓

A special kind of Gradient: Desent d. To Reduce Loss

↓

"Steepest" ~

↓

Training A Neural Network

↓

Generalize the concept: "Learning" (Supervised learning)

Derivatives.

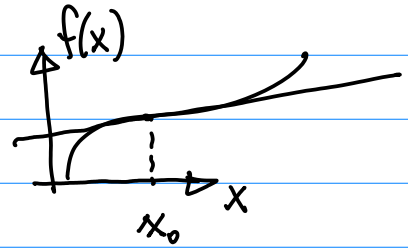
Definition $\frac{d}{dx} f(x) \triangleq \lim_{\Delta x \to 0} \frac{\Delta f}{\Delta x}$

$$= \left. \frac{f(x+\Delta x) - f(x)}{\Delta x} \right|_{\Delta x \to 0} \quad \cdots (1)$$

Note Eqn(1) is Based on "Forward Difference".

$f(x+\Delta x)$

forward of x



if $f'(x) = \frac{d}{dx} f(x) > 0$, then $f(x)$ at $x = x_0$ will increase

if $f'(x) < 0$, then $f(x)$ will decrease

if $f'(x) = 0$, then $f(x)$ is unchanged.

For a function $f(x_1, x_2)$, the partial Derivative of $f(x_1, x_2)$

$$\frac{\partial f}{\partial x_1}, \frac{\partial f}{\partial x_2} \quad \cdots (2)$$

$$\frac{\partial f(x_1,x_2)}{\partial x_1} \triangleq \lim_{\Delta x_1 \to 0} \frac{f(x_1+\Delta x_1, x_2) - f(x_1,x_2)}{\Delta x_1} \quad \cdots (2a)$$

$$\frac{\partial f(x_1,x_2)}{\partial x_2} \triangleq \lim_{\Delta x_2 \to 0} \frac{f(x_1, x_2+\Delta x_2) - f(x_1,x_2)}{\Delta x_2} \quad \cdots (2b)$$

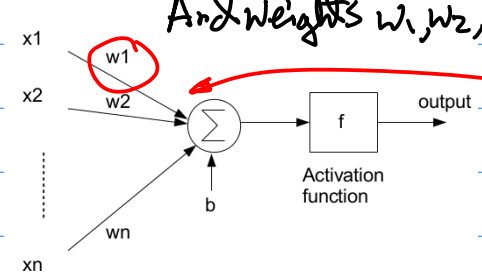
$\frac{\partial f}{\partial x_1} > 0$, $f \uparrow$ as $x_1 \uparrow$

$\frac{\partial f}{\partial x_2} > 0$, $f \uparrow$, as $x_2 \uparrow$

Define gradient of a function $f(x_1, x_2)$

(Note: Connection to Neural Network, And a Single Neuron. inputs $x_1, x_2, \cdots x_N$ And weights $w_1, w_2, \cdots, w_N$)

Fig1.



Defined a vector

$$\begin{pmatrix} \frac{\partial f(x_1,x_2)}{\partial x_1} \\ \frac{\partial f(x_1,x_2)}{\partial x_2} \end{pmatrix} \text{ or } \begin{pmatrix} \frac{\partial f}{\partial x_1} \\ \frac{\partial f}{\partial x_2} \end{pmatrix}$$

We can use github Lecture Notes to expand Eqn(3) to N-Dimensional Case $\cdots (3)$

$$\nabla f = \begin{bmatrix} \frac{\partial f}{\partial x_1} \\ \frac{\partial f}{\partial x_2} \\ \vdots \\ \frac{\partial f}{\partial x_n} \end{bmatrix} \quad \cdots (3b)$$

For A Single Neuron, we have gradient function as follows.

$$\nabla f = \begin{bmatrix} \frac{\partial f}{\partial w_1} \\ \frac{\partial f}{\partial w_2} \\ \vdots \\ \frac{\partial f}{\partial w_N} \end{bmatrix} \quad \cdots (3c)$$

Guideline for the Development of "Training" Technique:

Negative Gradient gives the Steepest Descent of a loss function f.

Note: Negative gradient gives the Best way to adjust w_i in a Neural Net

to make the Neural Net to Reach to optimal solution.
e.g. minimized loss function.

The Approach to verify the Above Technique:

Generalized loss function, as f, or $f(\cdot)$, or $f(x_1, x_2, \cdots, x_N)$, or $f(\underline{x})$, or $f(w_1, w_2, \cdots, w_N)$, or $f(w)$

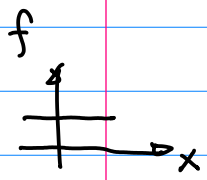
↓

Simple/Generalized Description to connect independent Variable $x_1, x_2, \cdots, x_N$ (or $w_1, w_2, \cdots w_N$) to the function f (loss function)

Taylor Expansion for $f(x)$

$$f(x) = f(x_0) + \frac{f'(x)}{1!}(x-x_0) + \frac{f''(x)}{2!}(x-x_0)^2 + \cdots + \frac{f^{(k)}(x)}{k!}(x-x_0)^k + \cdots \quad \cdots (4)$$

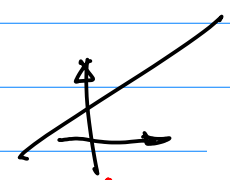
$\Rightarrow$ Constant + 2nd (Linear function) +



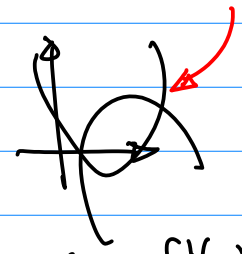
$$\frac{f'(x_0)}{1!}(x-x_0)$$

Coefficient, Slop. $a(x-x_0)$

$y = ax + b$ Linear function



$$\frac{f''(x_0)}{2!}(x-x_0)^2 \quad \text{Quadratic Term.}$$



$$f(x) = f(x_0) + \frac{f'(x_0)}{1!}(x-x_0) + R_n(x)$$

higher Order terms

$$f(x) \simeq \underbrace{f(x_0) + \frac{f'(x_0)}{1!}(x-x_0)}_{\text{Linear function}}$$

$$f(x) = f(x_0) + \frac{f'(x_0)}{1!}\Delta x \quad \cdots (5)$$

We expand function f (Loss) into multi-dimensional Case, e.g. $x_1, x_2, \cdots$ (or $w_1, w_2, \cdots$).

Choose work on 2D Case. e.g.

$$f(x_1, x_2) = f(x_{10}, x_{20}) + \frac{\partial f}{\partial x_1}(x-x_{10}) + \frac{\partial f}{\partial x_2}(x-x_{20}) + \cdots \quad \cdots (6)$$

$$f(x_1, x_2) \simeq f(x_{10}, x_{20}) + \frac{\partial f}{\partial x_1}(x-x_{10}) + \frac{\partial f}{\partial x_2}(x-x_{20}) \quad \cdots (7)$$

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

Question, if we change independant variable x_1 and x_2 by the negative gradient of f, then the function f is decreased or increased?

Based on the Theory, we should have loss function f decreased in an optimized way, e.g. "Steepest Descent."

$$(x_1^{k+1}, x_2^{k+1}) = (x_1^k, x_2^k) + [-\eta(\nabla f)^t] \qquad (5)$$

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

$$(\nabla f)^t = \left(\frac{\partial f}{\partial x_1}, \frac{\partial f}{\partial x_2}\right)$$

x_1^{k+1}, x_1^k

step @ k+1 at step k

$$\begin{cases} x_1^{k+1} = x_1^k + (-\eta)\frac{\partial f}{\partial x_1} & \cdots(8a) \\ x_2^{k+1} = x_2^k + (-\eta)\frac{\partial f}{\partial x_2} & \cdots(8b) \end{cases}$$

If we use Eqn(8a), (8b) to update the independant variables x_1, x_2 (or weights w_1, w_2), then you are sure to be able to find loss function f at this step k+1, is smaller than itself, e.g. f function at step k.

$$f(x_1, x_2) - f(a, b) = (x_1 - a, x_2 - b)\nabla f \qquad (10)$$

The Above equation, eqn(10), is Taylor Expansion

Note: from the Analysis below, we have:

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

$$f_{x_1} = \frac{\partial f}{\partial x_1}$$

$$f_{x_1}^2 = \left(\frac{\partial f}{\partial x_1}\right)^2 \geq 0$$

Similarly $f_{x_2} = \frac{\partial f}{\partial x_2}$, $f_{x_2}^2 = \left(\frac{\partial f}{\partial x_2}\right)^2 \geq 0$

Therefore $-\left[\left(\frac{\partial f}{\partial x_1}\right)^2 + \left(\frac{\partial f}{\partial x_2}\right)^2\right] \leq 0$

Hence $f(x_1, x_2) - f(a, b) \leq 0$

$$f(x_1, x_2) \leq f(a, b) \quad //$$

$$x_1 - a = -\eta\frac{\partial f}{\partial x_1}$$

$$x_1^{k+1} - x_1^k = -\eta\frac{\partial f}{\partial x_1}$$

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

$x_1^k = a$, $x_2^k = b$ $\quad f(a, b)$

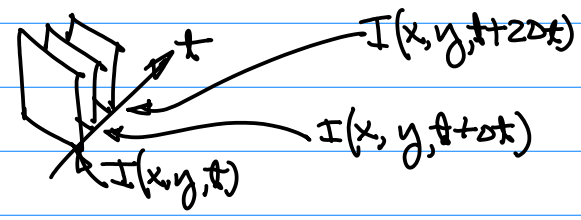
$$f(x_1, x_2) \simeq f(x_{10}, x_{20}) + \frac{\partial f}{\partial x_1}(x - x_{10}) + \frac{\partial f}{\partial x_2}(x - x_{20}) \quad \text{Eqn(7) PP.11}$$

OpenCV

Note: One of the tools for Python Programming for this class is Pycharm.

Create A New Project → Configuration of the environment & packages Needed

Example: OpenCV for Video.



Exercise Use your Smartphone to Capture a video clip which has the following feature:

1. White Background, Such as Printer paper or white Board, then, write your last 4 Digits on this white Background Paper/Board.
2. Be sure to use a marker to give adequate contrast and width for each mark. (720P or 1080P Resolution)
3. Save the video (may have 20~60 seconds video clip) for the future use By Next Lecture.

Note: we will discuss Computer Vision preprocessing techniques in the next Lecture.