CMPE259

F22, HL

1/

August 23 (Tue)

First Day of the Class

1. Organizational meeting

"Green Sheet"

Repo: github/hualili/opencv/deep-learning-2022s/

Email: hua.li@sjsu.edu

(650) 400-1116 Cellphone for

Text message Only.

Office Hours: M.W. On Zoom.

(See syllabus for the Zoom Link).

2 Software Tools:

Anaconda — Install it By the end of this week;

TensorFlow, TF 2.0

OpenCV.

3. Prerequisites: CMPE255 & CMPE257

Homework: To upload a Copy of your un-official Transcript to show the required Courses Satisfied.

On CANVAS.

4. Textbook. Deep Learning with Python.

Keras (API) for TF.

Robot Vision Book By Horn. (Heavy Theoretical Book, Good Reference for OpenCV Algorithms.

Good Theoretical Foundations)

5. Projects { Mandatory Assigned Project
Team Project (Mandatory)

4-Person Team. Presentation By the End of the Semester.

August 25 (Wed)

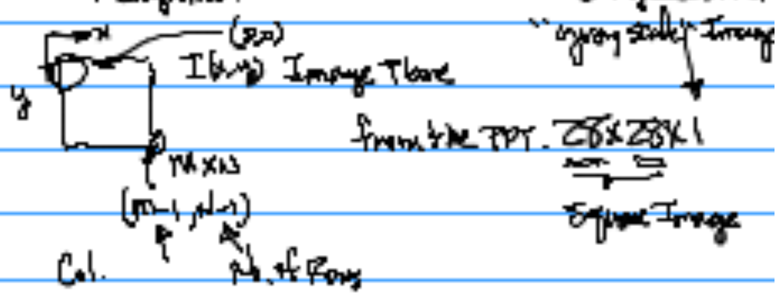
Note: 1. The lecture Note will be posted on to the github.

2. Zoom Recording will be posted on the the github.

Homework: By A week from today. 1. Anaconda Installation; 2. OpenCV Installation. Submission On Canvas. Jpg/Png Image → pdf → zip 2 pdf.

Example: (github: 2022F-103-)

MNIST Architecture for Handwritten Digits Recognition.



Gray scale image → 1 Channel → 8 bit → [0, 255]

First Layer of the MNIST Architecture

n1 Channel/plane of the 1st Convolution Layer

C1

Next, pooling → Reduction of Resolution → 2×2 becomes 1

M1

From the Architecture diagram:

C1 M1 → C2 M2 → Flatten → FFNN (FF)

To generalize the quick inspection of the the CNNs, we have to investigate the Behavior of Each Single Neuron as the Basic Building Block.

CmpE258

Aug. 25, 22

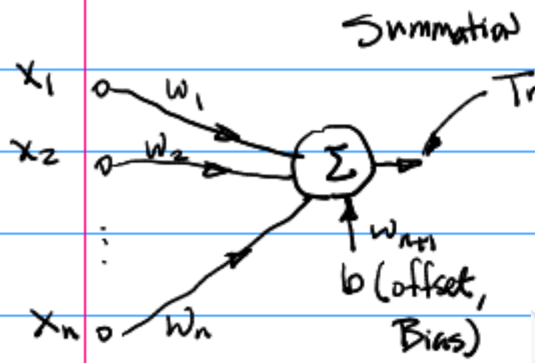
August 30. Tue

2/

Ref. 1

https://github.com/hualili/opencv/blob/master/deep-learning-2022s/2022F-103b-NN-Intro-Python-v5-2022-8-25.pdf

opencv / deep-learning-2022s / 2022S-103a-notation-neuro-loss-function-2022-2-8.pdf



Input / Excitation in Vector Form: $\underline{X} = (x_1, x_2, \cdots, x_n)$ ...(1)

Weights, links each excitation to the Neuron

$$W = (w_1, w_2, \cdots, w_n) \quad \cdots (2)$$

Ref 2. Code

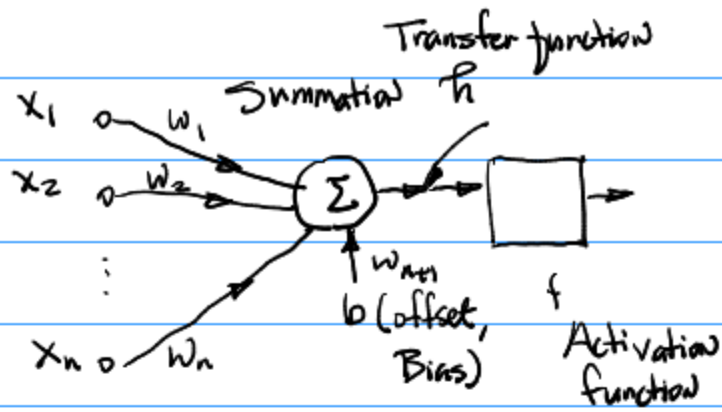
https://github.com/hualili/opencv/blob/master/deep-learning-2022s/2022S-110b-%232019S-31-6mnist-numerals-ch02.py

$$x_1 w_1 + x_2 w_2 + \ldots + x_i w_i + \cdots + x_n w_n + b w_{n+1} = h$$

Example:

$$\sum_{i=1}^{n} x_i w_i + b w_{n+1} = h(x_i w_i) \text{ or Simply } h(\underline{X} W) \quad \cdots (3)$$

$$h(\underline{X} W; b),\ h(\underline{X}, W)$$



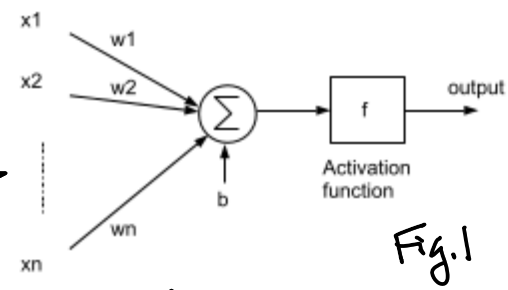


Fig. 1

$(x_1, x_2, \cdots, x_n)$ Feature Vector with Dimension N.

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

Transfer function $h(\cdot)$.

$$w_{n+1} b = b'$$

$$f(h(\underline{X} W)) = f\left(\sum_{i=1}^{n} x_i w_i + b w_{n+1}\right) \quad \cdots (4)$$

$$h(\underline{X} W) = \sum_{i=1}^{n} x_i w_i + b w_{n+1}$$

Examples of Different Activation functions include RELU. A piecewise Linear.

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

Activation function. Its output is the Response of the Neuron.

Note: Be Able to Build A Single Neuron per a technical specification, such as well, Activation $f(\cdot)$, Draw a Block

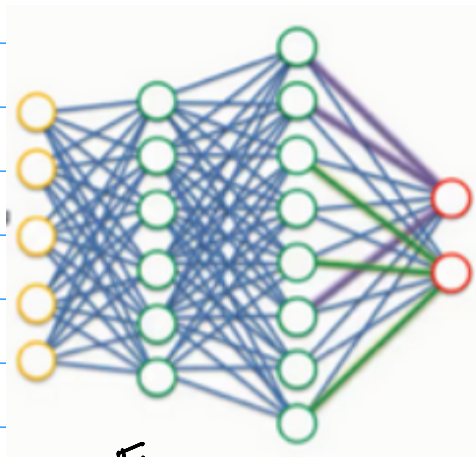
CMPE258

Aug. 30.

Consider the output of the Neuron y from Eqn(17).

Output of A Single Neuron.

For multiple Neuron Output, see Fig.2



y_1

y_2

Fig.2

y_i $\quad\ldots(1)$

Subscript: $i=1,2$

No. of Output at the Output Layer.

y_i, $i=1,2,\ldots,M$.

In practical Application,

y_i^j $\quad\ldots(2)$

Superscript

$j=1,2,\ldots,P$ No. of Experiments performed, Training Performed.

Look at the Concept & Definition of Loss function.

Mathematically To Compare a Neural Network Output (Single Neuron Output)

function f. function g

Comparision of the Similarity or diffence between f and g.

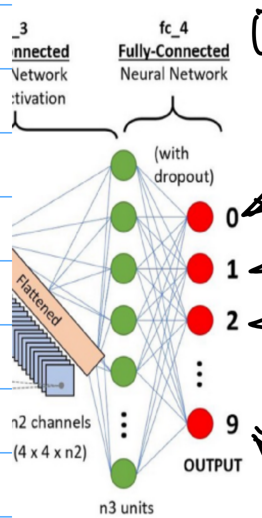
$$\begin{cases} f-g \\ f/g \end{cases}$$

Difference Between Two Functions.

Take this Approach to define Loss function,

$$y - \hat{y} \quad\ldots(3)$$

Ground Truth. Output(prediction) from the Neuron Outputs



y_0, y_1, y_2, $\vdots$, y_9 $\quad \hat{y}_i$

$i=0,1,2,\ldots,9$

Fig3.

$y_i - \hat{y}_i$ ... (4-a)

To measure All the output for Each Training/Experiment

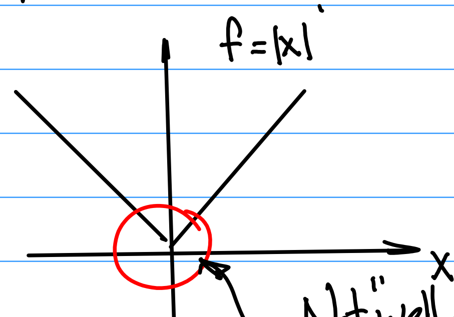
$$\sum_{i=0}^{q} (y_i - \hat{y}_i) \quad \ldots (4\text{-}b)$$

Expand this to Experiment/Training up to "P" Times

$$\sum_{j=1}^{P} \left[\sum_{i=0}^{q} (y_i^j - \hat{y}_i^j) \right] \quad \ldots (4\text{-}c)$$

Note: Eqn (4-c) may Lead to positive & Negative Terms Cancellation.

Fix: AbsoluteValue ? $\rightarrow$ Squared Instead.



Not "well-behaved"!

"Well-Behaved" System (Function) $\rightarrow$ derivative/Partial Derivative up to order "K".

$$L = \sum_{j=1}^{P} \left[\sum_{i=0}^{q} (y_i^j - \hat{y}_i^j) \right]^2 \quad \ldots (4\text{-}d)$$

$\rightarrow$ J, or Φ

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

Ground Truth

For a Single Neuron @ the Output Layer

Training Based "Steepest Gradient Descent" SGD.

Example: Given A function $f(x) = x^2$, Find its Derivative

$$\frac{df}{dx} = 2 \cdot x$$

To get rid of Coefficient from the derivative,

Let's define $f(x) \triangleq \frac{1}{2} x^2$

then, $\frac{d}{dx} f(x) = \frac{1}{2} \cdot 2 \cdot x = x$

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

Sept 1st (Th).

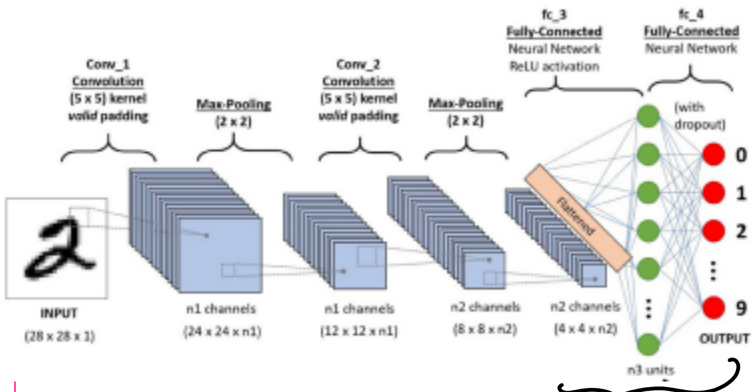
Ref:

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

Example: Background on "Learning" of A NN.

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

CMPE258
Sept 1st.



Fully Connected
FeedForward NN

To Train ANN, we take eqn (23), e.g.
Loss function (error function),

$L(\cdot) \rightarrow L(W_{\sim})$

Independent Variables

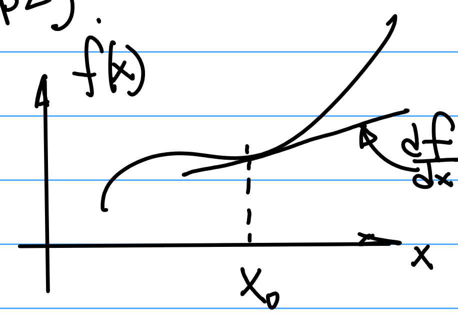
Minimize the Loss $L(\cdot)$, which is the process of Training, which leads to Learning for the NN.

↓

Since the Loss function in Eqn (23) is formulated with a ground Truth y, this define a Supervised Learning.

Math. Formula: Prediction of Function's Behavior, we like to know given the current function value (Loss function) how this function is going to change at next moment, increase? stay the same? Or Decrease?

Derivative of A given function is a good indicator to give us the description of the function behavior Next Step Ahead (Very Small tiny Step Δ).



$$\frac{df}{dx} = \lim_{\Delta x \to 0} \frac{f(x+\Delta x) - f(x)}{\Delta x} \quad \ldots (1)$$

↓ Intuition

$f(x+\Delta x) - f(x) \rightarrow > 0$ (derivative)

$\Delta x \simeq 1$ unit

if the derivative is greater than 0

then $f(x+\Delta x) > f(x)$, the next Step function $f(x)$ is increased;

if the derivative $\frac{df}{dx} < 0$, then

$f(x+\Delta x) < f(x)$

if the derivative $\frac{df}{dx} = 0$, then

$f(x+\Delta x) = f(x)$

Consider Two Dimensional Case as an Example for n-dimensional Case.

CMPE258

Sept. 1st.

$$f(x_1, x_2) \begin{cases} \dfrac{\partial f(x_1, x_2)}{\partial x_1} \rightarrow \text{Single Neuron} \\[2ex] \dfrac{\partial f(x_1, x_2)}{\partial x_2} \end{cases}$$

In Case of Training They are "Weights" w_1, w_2.

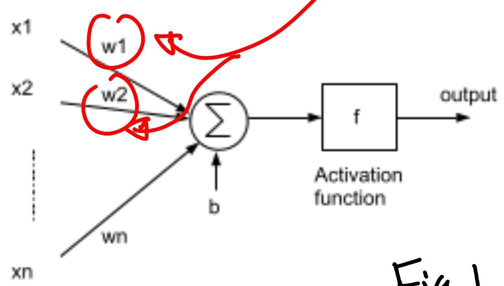


Fig. 1

$\frac{\partial f}{\partial x_1} \rightarrow \frac{\partial f}{\partial w_1}$ In the Context of Training.

Conclusion: Use the Partial Derivate With reset to the Weight W_i as an indicator to measure if the Loss function is got reduced or not.

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

Example: Consider A Technique which allows the trainning to be more effect, e.g., to minize training and prediction error (Loss) function

Steps for Development of this technique:

Multi-dimensional Derivative

↓

gradient

↓

the Steepest gradient

↓

the Steepest Descent gradient

↓

The Core technique to train NN. SGD

Ref: from the github

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

Homework (Opt). Due A week from Today. Sept. 8th, Thursday. ON CANVAS.

1° Installation of TensorFlow. Version 2.0 or higher.

2° Screen Capture to show the installation is successful

Note: All Different Development Tools | Environments including google co-lab, jupyter Note Book etc. Are OK., However for the Deployment purpose, Projects homework Submission must be in Python Stand-Alone form.

Sept. 6 (Th)

Homework: Due 1 week from Today Sept 13.

Cmpe258
Sept. 6.

1. OpenCV Installation, Python.
2. Use Smart phone to Capture 5~10 Seconds Video Clips. .avi, mp4 (mpeg 4).
3. Sample Code, github. See CANVAS for the Detailed links & Requirements.
4. Submission to CANVAS.
   ① Python Code;
   ② Original & Processed image Side by Side with your Name + SID.
   ③ Create One pdf file to Cover the Source Code, And Screen Captured Images.
5. Naming Convention

   HW_CV_First_LastName_Cmpe258_SID.zip

https://github.com/hualili/opencv/blob/master/deep-learning-2022s/2022S-104d-%232-pdisplay-2019-1-30.py

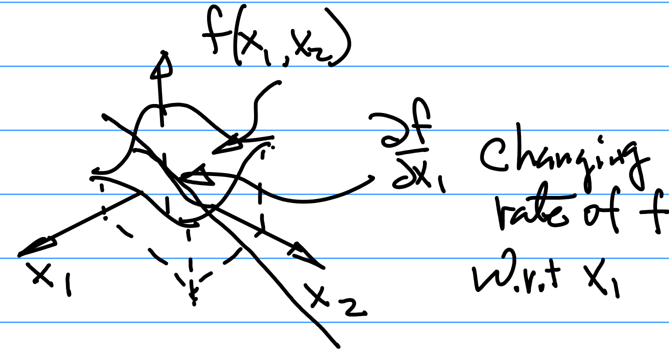
Higher Dimensional Function

$$f(\underbrace{x_1, x_2, \cdots, x_n}_{\text{Weights}, w_1, w_2, \cdots, w_N}) \quad \ldots (2)$$

Partial Derivatives:

$\frac{\partial f}{\partial x_1}$ w.r.t x_1, $\frac{\partial f}{\partial x_2}$ for x_2, ...

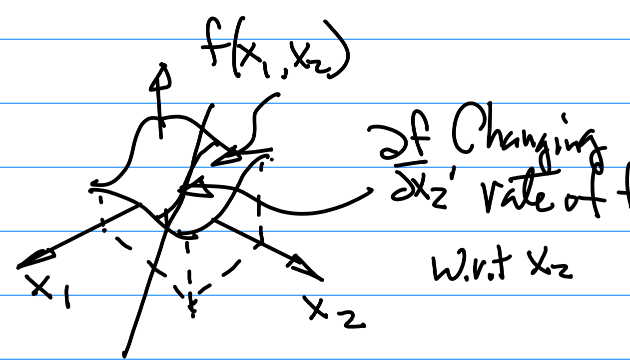
$\frac{\partial f}{\partial x_n}$ w.r.t x_n.



Example: Gradient Definition

Ref:

https://github.com/hualili/opencv/blob/master/deep-learning-2022s/2022S-105c-%2320-2021S-4gradient-descent-v2-final-2021-2-8.pdf



Loss function

Derivative, e.g. given $f(x)$, then

$$\frac{df}{dx} = \lim_{\Delta x \to 0} \frac{f(x+\Delta x) - f(x)}{\Delta x} \quad \ldots (1)$$

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

Consider the minization of function f (Loss Function) w.r.t. All possible weights.

Therefore, Put all the partial Derivatives together to form A vector, e.g., gradient.

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

for $n=2$,

$$\nabla f(x_1, x_2) = \begin{bmatrix} \frac{\partial f}{\partial x_1} \\ \frac{\partial f}{\partial x_2} \end{bmatrix} \quad \cdots (2b)$$

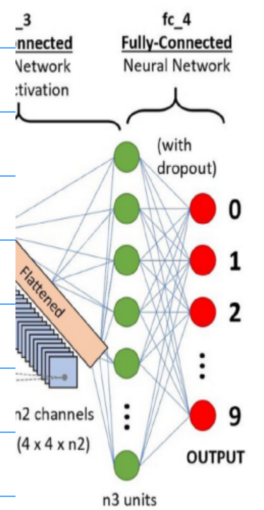
for $n=3$

$$\nabla f(x_1, x_2, x_3) = \begin{bmatrix} \frac{\partial f}{\partial x_1} \\ \frac{\partial f}{\partial x_2} \\ \frac{\partial f}{\partial x_3} \end{bmatrix} \quad \cdots (2c)$$

d. On the Right hand side of Eqn(5):

(x_1^k, x_2^k) $\begin{cases} \text{Dimension } n=2, (x_1, x_2) \\ \text{Time Index "k", Superscript} \end{cases}$

Output of the NN with its weights at Step K (Time) is x_1^k, x_2^k



e. On the Left (x_1^{k+1}, x_2^{k+2}), at the step $k+1$, to Reduce the Loss function, so update the new step By following

$$-\nabla f = -\eta \begin{pmatrix} \frac{\partial f}{\partial x_1} \\ \frac{\partial f}{\partial x_2} \end{pmatrix} \quad \cdots (3)$$

IV. Gredient Steepest Descent for Minimization

Conclusion:

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

a. Loss function f

b. $n=2$

e.g. $f(x_1, x_2, \cdots, x_n) \rightarrow f(x_1, x_2)$

c. Gradient

$\nabla f(x_1, x_2)$, or ∇f

Background:

Given a function $f(x)$, How do you Approximate this function By using Basic Building Blocks (B^3)?

$$f(x) = \underbrace{\text{Constant}}_{\text{Term}} + \underbrace{\text{A Linear}}_{\text{Term}} + \underbrace{\text{A Quadratic}}_{\text{Term}} + \text{A Cubic Term} + \cdots \quad \cdots (4)$$

CmpE258

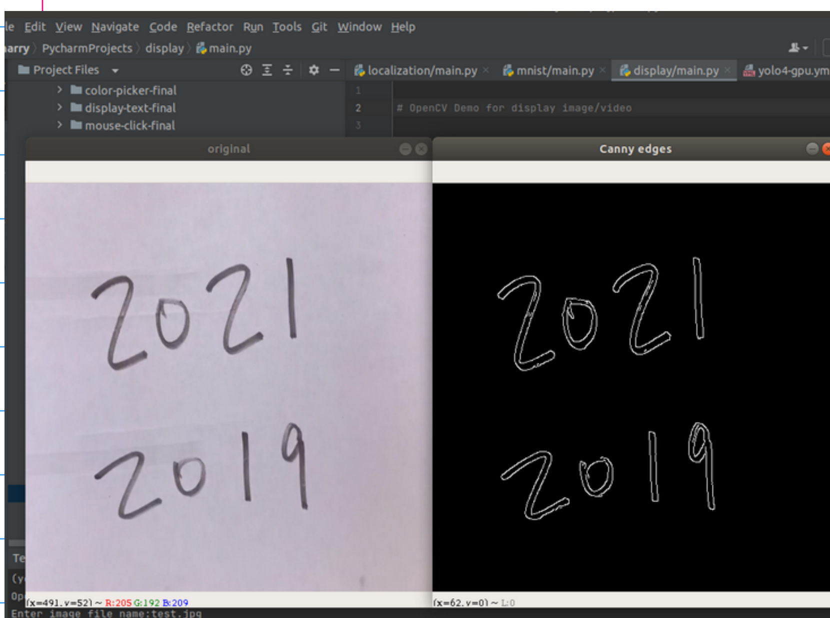
Sept 6, 22

Taylor Expansion:

$$f(x) = f(x_0) + \frac{df}{dx}\cdot(x-x_0) + \frac{d^2f}{dx^2}(x-x_0)^2 + \dots + R_n(x) \quad \dots (4)$$

$$f(x_1,x_2) = \underbrace{f(x_1,x_2)\Big|_{\substack{x_1=x_{10}\\ x_2=x_{20}}}}_{\text{Constant}} + \underbrace{\frac{\partial f}{\partial x_1}(x_1-x_{10}) + \frac{\partial f}{\partial x_2}(x_2-x_{20})}_{\text{Linear Term}} + \underbrace{\frac{\partial^2 f}{\partial x_1^2}(x_1-x_{10})^2 + \frac{\partial^2 f}{\partial x_2^2}(x_2-x_{20})^2}_{\text{2nd Order}} + \dots + \underbrace{R_n(x_1,x_2)}_{\text{Higher Order Terms}} \quad \dots (5)$$

Note: The screen capture for your homework reference.



The goal is to verify the formula for updating the weights of A given NN. (see Handout 1, Eqn(5)).

Step 1. Taylor Expansion → Step 2 Simplify the Taylor Expansion By just using up to the Linear terms → Step 3. Re-arrange the Taylor Expansion in the form of Training formula (In Eqn(5), in Handout 1) → Step 4. Analyze the re-arranged formula, to Reach the Observation which lead to the Conclusion, e.g., using gradient descent, we can Redme the Loss

Sept. 8 (Thu)

Note: 1° Check the CANVAS for Both Homeworks.

Example: From 1D Case in Eqn (4), we can expand the Taylor Expansion to higher Dimension n. to Capture multiple excitations, multiple weights w_i, $i=1,2,\dots,n$.

Consider $n=2$

CmpE258
Sept. 8, 22

function through each step of the training.

From the Handout, we have as Step 1 & 2:

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

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

Comparison of A "loss" function, write

$\Delta x_1 = x_1 - a$
$\Delta x_2 = x_2 - b$

And $\nabla f = \begin{pmatrix} f_{x_1} \\ f_{x_2} \end{pmatrix}$

Hence, we have

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

Let $\Delta x_1 = -f_{x_1}$, $\Delta x_2 = -f_{x_2}$

Therefore,

$$f(x_1,x_2)-f(a,b) = (-f_{x_1}, -f_{x_2})\begin{pmatrix} f_{x_1} \\ f_{x_2} \end{pmatrix}$$

$$= -(f_{x_1}^2 + f_{x_2}^2) \leq 0$$

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

OR, $f(x_1,x_2) \leq f(a,b)$

Loss function Updated at the next step by Eqn(5) in Handout 1.

Loss function at the Current Step.

Note: The Requirement for this discussion on Notations, And Formulation, especially, the Eqn(5) in Handout 1 is required To Be Able to use these tools to Analyze the problem, And to Perform Verification (e.g. design).

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

To Be Continued.

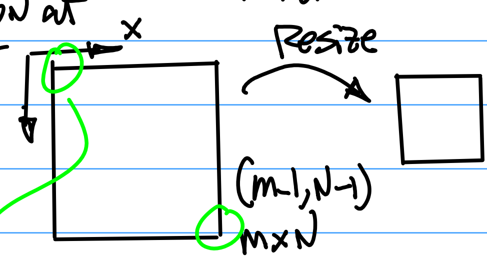
OpenCV Homework. Sample code

opencv/2022S-104d-#2-pdisplay-2

a. b. c.

```
#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')

ind = 0

while True:
    cv2.imshow(window_name, src)

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

Note: OpenCV Resize Function.



CmpE258
Sept.8,22


```python
import numpy as np
#import argparse
import cv2

img = input('Enter image file name:')

image = cv2.imread(img, cv2.IMREAD_COLOR)

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

image = cv2.resize(image, (512, 512))

gray = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)
edges = cv2.Canny(gray,100,200)
```

a.

b.

a. Conversion to Gray-scale image;
b. Canny Edge Detection.