CMPE259

F22, HL

August 23 (Tue)

First Day of the Class

1. Organizational meeting
   "Greensheet"

Repo: github/Hualili/opencv/deep-learning-2022S/

Email: hua.li@sjsu.edu

(650) Hua-Lili 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)

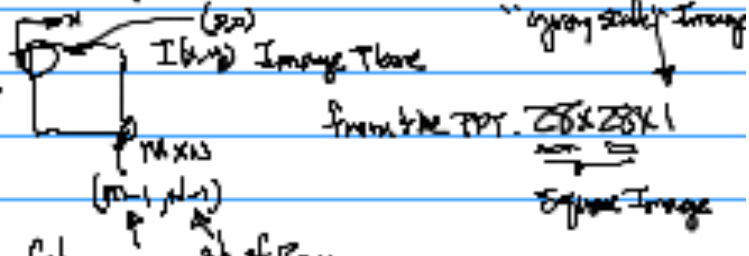
Note: 1- The lecture Note will be posted on to the github.

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 file.

Example: (github: 2022F-183_)

MNIST Architecture for Handwritten Digits Recognition.

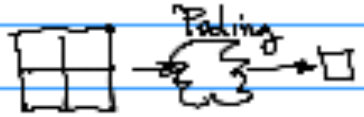


(gray scale image → 1 channel → 8 bit → [0, 255])

First Layer of the MNIST Architecture

n1 channel/plane of the 1st Convolution Layer C1

Next, pooling → Reduction of Resolution → 252 becomes



M1

From the Architecture diagram:

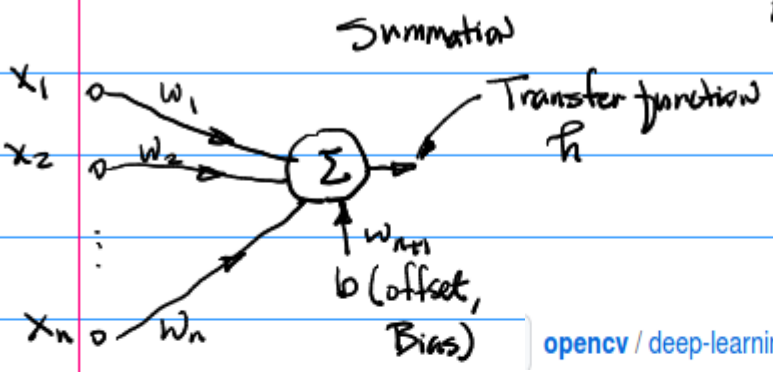
C1 M1 → C2 P2 → Flatten → (FF) FFNN

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: $X=(x_1, x_2, \cdots, x_n)$ ...1)

Weights, links each excitation to the Neuron

Ref2. Code
https://github.com/hualili/opencv/blob/master/deep-learning-2022s/2022S-110b-%232019S-31-6mnist-numerals-ch02.py

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

$x_1w_1 + x_2w_2 + \cdots + x_iw_i + \cdots + x_nw_n + bw_{n+1} = h$ ... (3)

$\sum_{i=1}^{n} x_iw_i + bw_{n+1} = h(x_iw_i)$ or simply $h(Xw)$

$h(Xw; b)$, $h(X, w)$

Example:

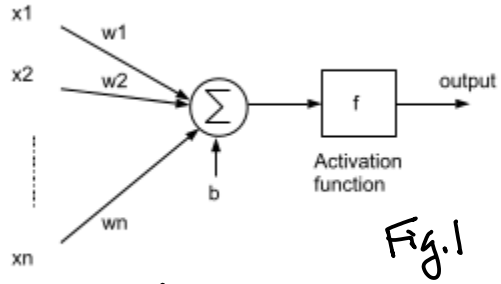
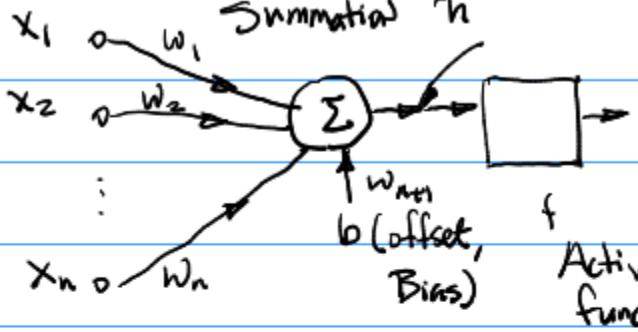


Fig.1

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



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

Transfer function $h(\cdot)$

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

$f(h(Xw)) = f\left(\sum_{i=1}^{n} x_iw_i + bw_{n+1}\right)$ ... (4)

$h(Xw) = \sum_{i=1}^{n} x_iw_i + bw_{n+1}$

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

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

Examples of Different Activation functions include RELU, Piecewise Linear.

Note: Be Able to Build A Single Neuron per a technical Specification, Such as n=11, 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

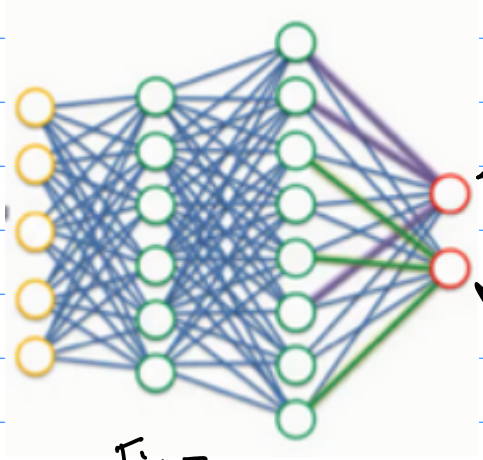


Fig.2

y_i ... (1)

Subscript: $i=1,2$

No. of Output at the Output Layer.

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

In practical Application,

y_i^j ... (2)

Superscript

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

Look at the Concept & Definition of Loss function. 3/

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

function f. function g

Comparision of the Similarity or difference between f and g.

$f-g$

f/g

Difference Between Two Functions.

Take this Approach to define Loss function,

$y - \hat{y}$ ... (3)

Ground Truth. Output (prediction) from the Neuron

Outputs

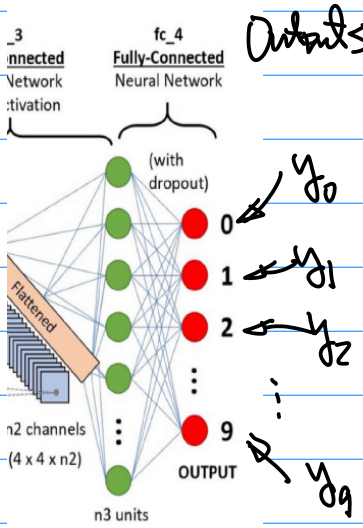


Fig3.

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

To measure All the output for Each Training/Experiment

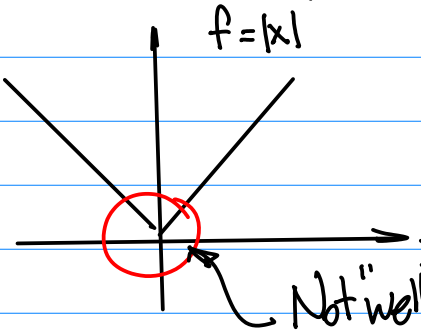
$$\sum_{i=0}^{q} (y_i - \hat{y}_i) \quad \text{...(4-b)}$$

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

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

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

Fix: Absolute Value ? → Squared Instead,



Not "well-behaved" !

∴ "Well-Behaved" System (Function) → derivative/Partial Derivative up to order "K".

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

→ J, or Φ

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

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

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

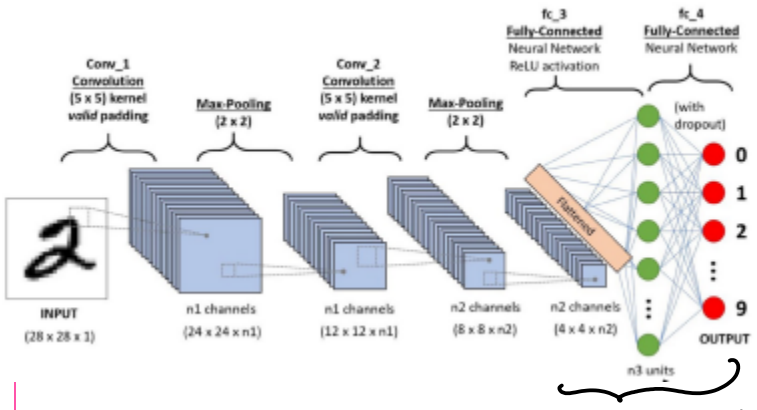
Sept 1st (Th).

Ref:

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

Example: Background on "Learning" of ANN.

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



Fully Connected Feedforward NN

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

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

↑ Independant Variables

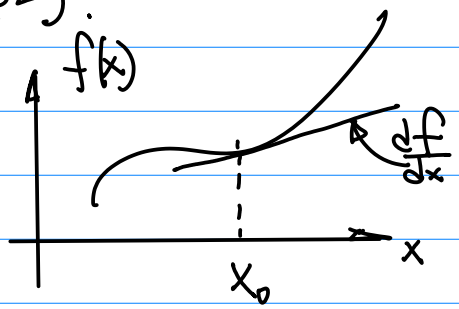
Minimize the Loss $L(\cdot)$, which is the process of Training, which leads to Learning for the NN.

↓

Since the Loss function in Eqn(23) is formulated with a ground Truth y, this defines 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

$f(x+\Delta x) - f(x) \rightarrow > 0$ (derivate if the derivative is greater than 0)

$\Delta x \simeq 1$ unit

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.

$f(x_1, x_2)$ → $\frac{\partial f(x_1,x_2)}{\partial x_1}$, $\frac{\partial f(x_1,x_2)}{\partial x_2}$ → Single Neuron

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

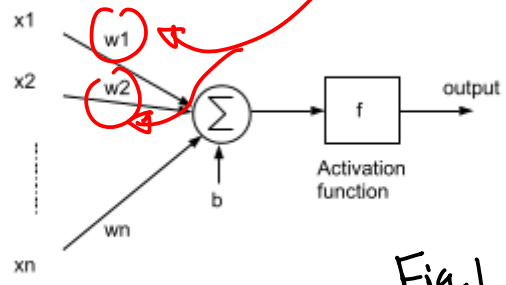


Fig.1

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

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

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

Example: Consider A Technique which allows training to be more effect, e.g., the 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/Enviroments 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

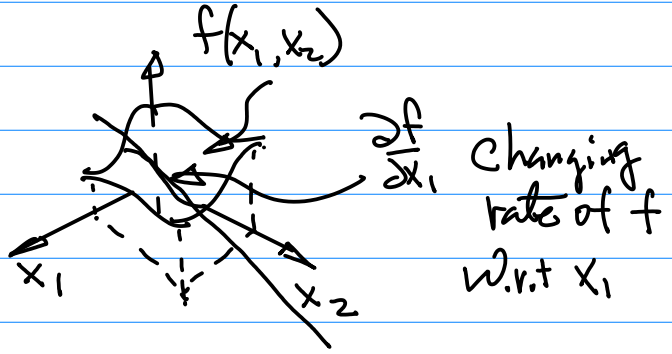
$$f(x_1, x_2, \cdots, x_n) \quad \cdots (2)$$

Weights, $w_1, w_2, \cdots, w_N$

Partial Derivatives:

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

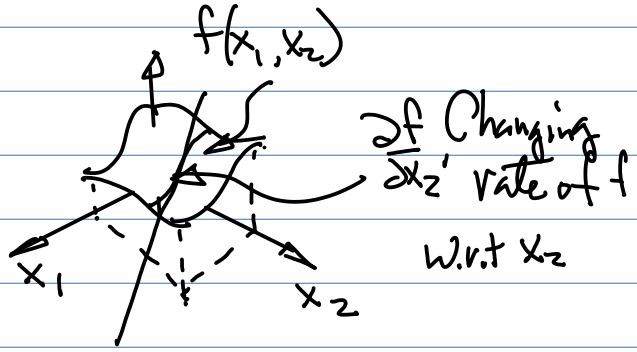
$\frac{\partial f}{\partial x_n}$ w.r.t x_n.



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

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

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

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.

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

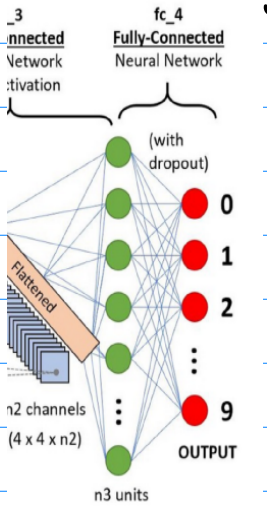
for n=2,

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

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) Dimension n=2, (x_1, x_2)
Time Index "k", Superscript
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

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

IV. Gredient Steepest Descent for Minimization

Conclusion:

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

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)?

$f(x)$ = Constant Term + A Linear Term + A Quadratic Term + A Cubic Term + ...(4)

CmpE258
Sept 6, 22

Taylor Expansion:

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

Note: The screen Capture for your homework refereme.