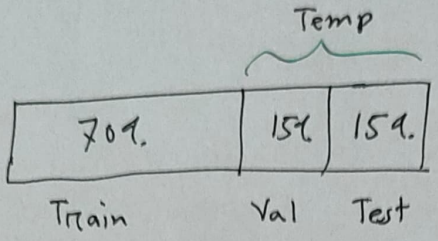
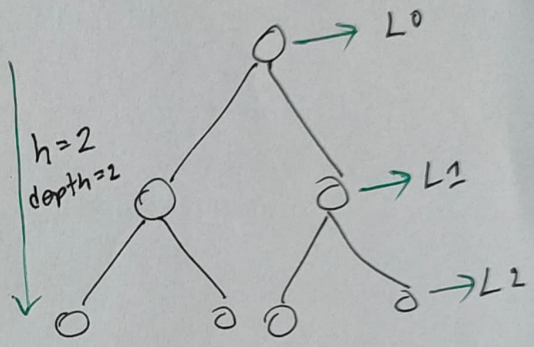
CSE 465 / L-14 / 04.03.2025 /

- train set used for train and create the model.
- validation set used for select the best model.



Decision tree classifier:

max_depth : important hyper parameter



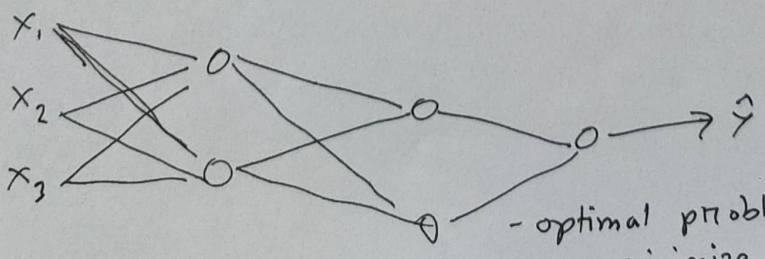
Why weight initialization is important?

deeplearning.AI ~~for vi~~

- visualise the importance of weight initialization.

L-15 / 09.03.2025 /

Challenges with optimizers in deep learning :



- optimal problem is to minimize the loss.

Gradient Descent:

$$W_{new} = W_{old} - \eta \frac{dL}{dW_{old}}$$

(η → learning rate)

- Here, optimal problem is to adapt the learning rate.
  - ↳ learning rate scheduling.

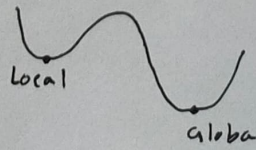
(i) Fixing the appropriate learning rate for NN.

(ii) we use a learning rate scheduler to change the learning rate.
  - ↳ pre-defined, when it will be change.
  - for this, it doesn't solve the problem. we don't know when it will converge.

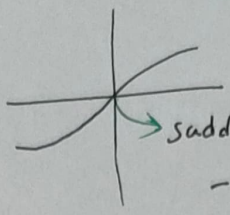
⇒ there are different weight.
- should we use same learning rate for all?
  ⇒ it can create another problem.

(iii) A fixed learning rate may not be appropriate for all weight and bias.

(iv) can get stuck into local minimum.
- stochastic most of the time avoid that, but not all the time.



Saddle point:



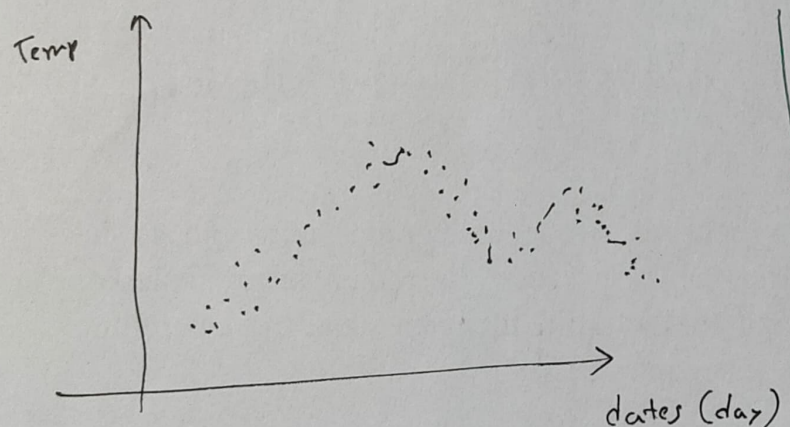
saddle point,

- not minimum, not maximum.
- don't stay just change to min-maxi or maxi to mini.
- derivative will be zero.

Optimizers:

- Momentum
- NAG
- RMSProp
- Adam

Exponentially weighted moving average (EWMA)



⇒ time series data,

- data change according to time.

⇒ stock price, temp.

EWMA is a technique that can uncover trends in time seriese data.

$$V_t = \beta V_{t-1} + (1-\beta)\theta_t$$

V_t = moving average at time t

θ_t = data at time t

β = 0.9 [Previous data get most weight than current data]

* In time seriese data,

- in normal scenerio, data don't have any pattern.
- but the data of moving average may show the pattern of it.

* Pandas dataframe. EWM ~~fe~~ methods:

$\alpha = 1 - \beta$

$\alpha \uparrow \Rightarrow$ smoth capture (current data priority)

$\alpha \downarrow \Rightarrow$ not so smoth (Previous data priority)

* we can use this concept to change the learning rate based on current data.

Next Class

Midterm

- Calculation Related Question
- 30-40 marks

L-16 / 11.03.2025

Midterm Exam

L-17 / 16.03.2025 /

CNN:

- Convolutional Neural Network
- special kind of NN for processing data that has a known grid like topology like time series data (1D), or images (2D).

There are 4 types of layers in CNN:

- Convolution layer
- Pooling layer
- ~~Fatt~~ Flatten layer
- Fully connected layer (FC)

Limitation of ANN:

- For higher dimension image, learning parameter increased exponentially.
  - high chance of overfitting
  - takes a lot of time
- ANN can't identify if we change the position of the object.
  - image locality
    ⇒ pixels that are spatially close to each other in an image often share strong relationships, with pattern.

Properties of images as a data:

- Spatial locality (image locality)
- Translational invariance

⇒ if ~~an~~ an object in an image shifts to a different location, the neural network should still recognize it as the same object.

Comparison of ANN & CNN:

Slide Page - 11

Size of feature map:

image	filter	=	feature
$n \times n$	$k \times k$		$(n-k+1) \times (n-k+1)$

⇒ if we go deeper into the network, the feature maps keep shrinking, leading to information loss.

⇒ known as vanishing spatial information

Loss of edge information:

- edge pixel used fewer time compared to the central pixel.
  ⇒ one kind of biased feature extraction.

- for solution, we need to add extra pixel around the input image.
  ⇒ padding (usally 0 padding)

Benefits of Padding:

- maintain spatial dimensions
- prevents loss of edge information
- allows for control over feature map size
- improves performance in deep CNN.

Types of Padding in CNN:

(i) Valid Padding

- no padding
- kernel applies only to the original image.
- feature map shrink after each convolution.

$\Rightarrow$ output size $= (n-k+1) \times (n-k+1)$

(ii) Same Padding

- add zero padding
- output size is the same as input size

$\Rightarrow$ $p = \left(\frac{k-1}{2}\right)$; for odd sized kernels

(iii) Full Padding

- maximum size of padding

$\Rightarrow$ $p = k-1$

$\Rightarrow$ output size $= (n+2p-k+1) \times (n+2p-k+1)$

Types of padding technique:

(i) Zero padding

(ii) Replication Padding

(iii) Reflection Padding

(iv) Circular Padding

Stride:

- step size by which the convolutional filter (kernel) moves across the input image during convolution.

Stride:

- help us to control:
  - step size of receptive field
  - amount of shink in output size
  - efficiency of computation.

Stride, S

$S = 1$; filter moves one pixel at a time.
⇒ Dense feature extraction

$S = 2$; Downsampling occurs

$S > 2$; Aggressive Down sampling

$$\text{Output width} = \left\lfloor \frac{\text{input width} + 2P - k}{S} \right\rfloor + 1$$

$$\text{Output height} = \left\lfloor \frac{\text{input height} + 2P - k}{S} \right\rfloor + 1$$

Problem with CNN:

- memory usage
- translation variance

} Solution - Pooling

Pooling:

- down sampling operation
- reduce the spatial dimension of feature maps
- reduce computational cost
- improve translation invariance
- prevent over fitting
- operates on small region [2×2].

Types of Pooling:

(i) Max Pooling

- takes large value in the pooling window
- preserve strongest features

(ii) Average Pooling

- compute the mean value in the pooling region.
- Retains global structure.

(iii) Global Pooling

- reduce the entire feature map to a single value per channel.
- replace fully connected layers, reducing parameter
- prevent overfitting in deep networks.

Comparison of Pooling and Convolution:

slide - 54

When not to use pooling:

- if spatial relationships are important
- if information loss is harmful
- if using strided convolution as an alternative

Difference in CNN architecture:

slide - 56

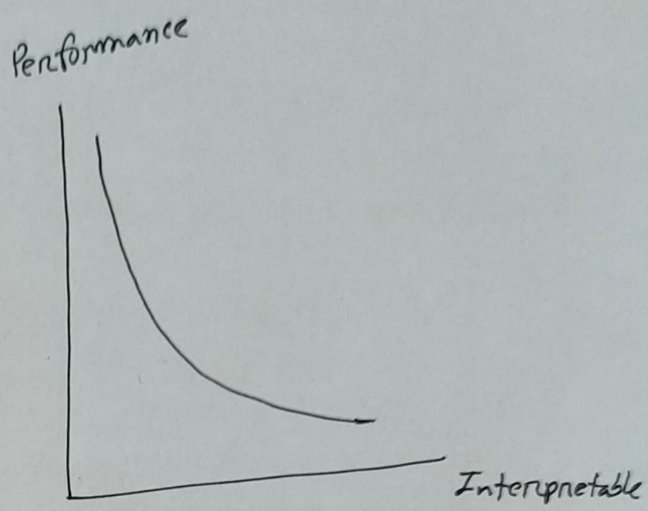
L-18 / 18.03.2025

Project Idea Submission

* Explainable AI (XAI)

⇒ LIME ⇒ XAI model

L = Local
I = Interpretable
M = Model-agnostic
E = Explanations

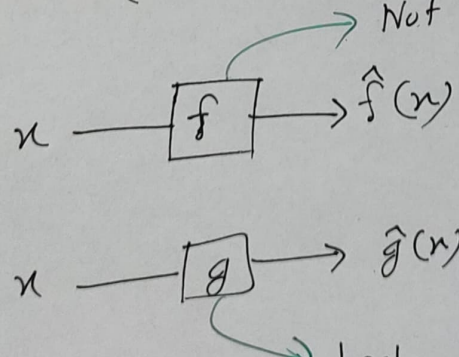


* "Why should I trust you?"
Explaining the Predictions of any Classifier

- Marco, Sameer, Carlos ⇒ 2016

* Loss function of LIME:

$L(f, g, \pi_x)$

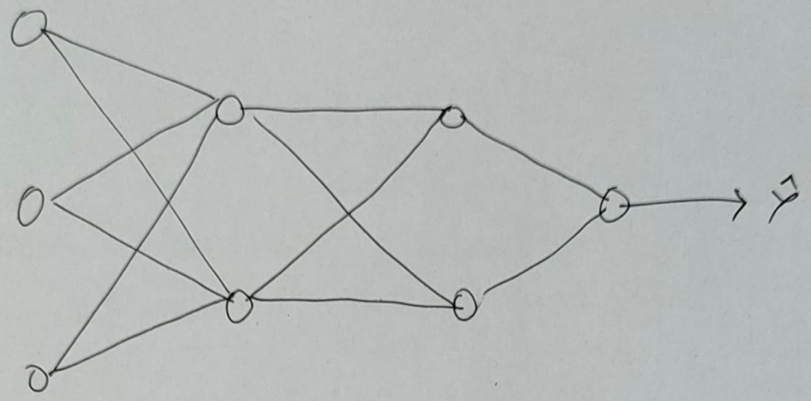


G = set of interpretable model.
$g \in G$.

best model, that predict the same as f to find out the explanation.

L-12 / 23.03.2025

* Feedforward model:



$$w_n = w_0 - \eta \frac{dL}{dw_0}$$

$$b_n - b_0 - \eta \frac{dL}{db_0}$$

* Gradient descent is the foundation of most optimizers

* Gradient descent:

  (i) Batch GD: entire dataset; not suitable for large dataset

  (ii) Stochastic GD: uncertain

  (iii) Mini-Batch GD: subset.

Example - Slide - 5

* Loss function for regression problem:

MSE:

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

Loss function for Batch GD:

$$w_n = w_0 - \eta \frac{dL}{dw_0}$$

$$\approx w_0 - \eta \nabla_{w_0} L(w)$$

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

$$\therefore \nabla_w L(w) = \frac{2}{N} \sum_{i=1}^{N} (y_i - \hat{y}_i)(-x_i)$$

$$y_i - \hat{y}_i = y_i - wx_i - c$$

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

Stochastic GD:

$$w_n = w_0 - \eta \frac{dL}{dw_0}$$

$$L = (y_i - \hat{y}_i)^2$$

$$\therefore \frac{dL}{dw_0} = -2x_i (y_i - \hat{y}_i)$$

Mini Batch-GD:

Previously, $L = \frac{1}{N} \sum_{i=1}^{N} (y_i - \hat{y}_i)^2$

Now, $L = \frac{1}{m} \sum_{i=1}^{m} (y_i - \hat{y}_i)^2$ [$m < N$

$$\therefore w_n = w_0 - \eta \frac{dL_{batch}}{dw_0}$$

$$\frac{dL_{batch}}{dw_0} = \frac{1}{m} \sum_{i=1}^{m} (y_i - \hat{y}_i)^2$$

Companion slide - 10

Advanced optimizers :

(i) Momentum

(ii) Adagrad

(iii) RMS prop

(iv) Adam

Momentum optimizer :

- SGD with momentum

⇒ Accelerate GD by keeping a moving average of past GD

$$V_t = \beta V_{t-1} + (1 - \beta) \nabla J(\theta)$$

$$\theta = \theta - \alpha V_t$$

↳ moving average

Pseudocode of the momentum optimizer

Slide - 13

Properties : Slide - 14

L-20 / 25.03.2025 /

Adagrad Optimizer :

- Adaptive gradient algorithm
- adapt the learning rate.

Previously,

$$\theta = \theta - \eta \nabla_\theta J(\theta)$$

in adagrad,

$$\theta = \theta - \underbrace{\frac{\alpha}{\sqrt{G_t + \epsilon}}}_{\text{adaptive learning rate}} \cdot \nabla_\theta J(\theta)$$

Here, square of gradient

$$G_t = G_{t-1} + \left(\nabla J(\theta)\right)^2$$

$G_3 = G_2 + \cdots$

$G_2 = G_1 + \cdots$

$G_1 = G_0 + \cdots$

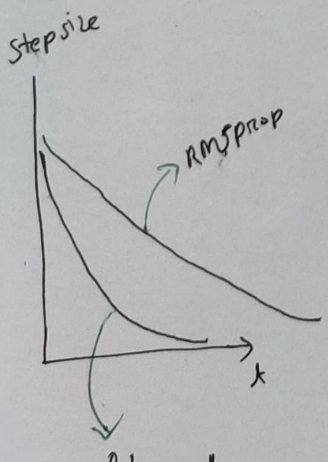
⇒ this is the sum of squared gradients

⇒ gradient can be negative, but square will be always positive, so this G_t will increase all the time. as it is in the denominator, the learning rate will decrease over time.

RMSprop:

- Root Mean Square Propagation.
- adaptive learning rate.

Here now, $G(t) > E[g^2]_t$



$$\theta = \theta - \frac{\alpha}{\sqrt{E[g^2]_t + \epsilon}} \cdot \nabla J(\theta)$$

$$E[g^2]_t = \beta E[g^2]_{t-1} + (1-\beta)\left(\nabla J(\theta)\right)^2$$

↳ momentum factor, just like moving average.

Adam Optimizers

- Adaptive moment estimation
- combined of momentum and RMS prop

formula: slide-19

- Bias connection ⇒ Example: slide-21,22

Comparison table: slide-23

Next class
Quiz-2
06.04.2025
CNN

Project Submission
18.04.2025