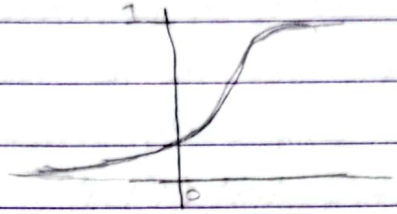
Date:

Sun Mon Tue Wed Thu Fri Sat

Activation functions.

1- Sigmoid: Squashes input in range of 0 to 1 used for binary classification

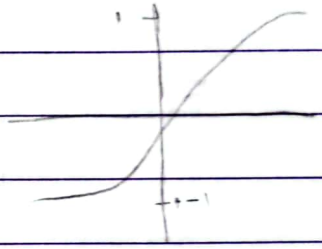
$$Sigmoid(x) = \frac{1}{1+e^{x}}$$



- vanishing gradient

2- Hyperbolic Tangent (Tanh): Squashes input into range between -1 & 1, often used in RNN

$$Tanh(x) = \frac{e^{x} - e^{-x}}{e^{x} - e^{-x}}$$

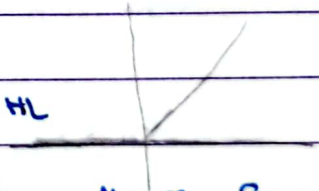


- vanishing gradient

3- Rectified Linear Unit (ReLU): Outputs the input if postive negitive input converted to 0, commonly used in deep learning models

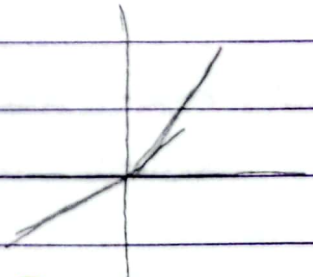
$$ReLU(x) = max(0, x)$$

Problem: neurons dying. Best for HL



4- LeakyRelu: Similar to LeakyReLU but allows small gradient when negitive input

$$LeakyReLU(x) = \begin{cases} x & \text{if } x > 0 \\ ax & \text{otherwise} \end{cases}$$



fixed problem of dying neurons of Relu.

less flexible Best for hidden layer

Date:

Sun Mon Tue Wed Thu Fri Sat

5- Parametric ReLU (PReLu): extends LeakyReLu by learning slope of negitive part.

when time < perform

leaky ReLu : a is fixed.

PReLu : a is learned to find optimal a

- Increased complexity longer training time

6- Exponential Leiner Unit (ELU): Similar to ReLu but smoother outputs. helps in training stability. Also faster convergence

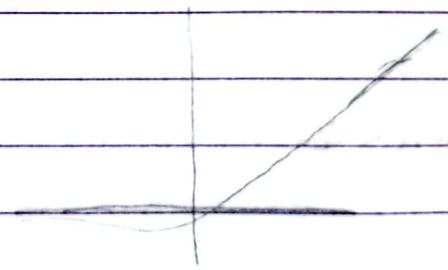
$$ELU = \begin{cases} x & \text{if } x > 0 \\ a(e^x - 1) & \text{otherwise.} \end{cases}$$

- Slightly more complex than ReLu or leaky ReLu

1- Swish: Proposed by google applies Sigmoid to the input and multiply itself

$$Swish(x) = x \cdot Sigmoid(x)$$

good alternative to ReLu



3- Softmax: used in multiclass classification, converts scores into probabilities summing to 1.

$$Softmax(x_i) = \frac{e^{x_i}}{\sum_j e^{x_j}}$$

- Output layer

- cannot be used for regression

Date:

Sun Mon Tue Wed Thu Fri Sat

9- Softplus: Smooth approximation of ReLU, helpful in optimization due to its differentiability

$$\text{Softplus}(x) = \log(1+e^x)$$

10- Gaussian Error Linear unit. (GELU): A smooth approximation of ReLU with gaussian noise claimed to improve performance.

$$\text{GELU}(x) = x \cdot \text{CDF}(x)$$

CDF(x) = the cumulative distribution function of the standard normal distribution.

- good in hidden layers - faster by perform better than ReLU

Date:

Sun Mon Tue Wed Thu Fri Sat

Loss functions

1. Mean Squared Error: average squared difference between actual and predicted values.
   - used in regression
   - Sensitive to outliers.
   - Not ideal for skewed distribution.
   - Best where outliers are minimal & normal dist

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

2. Mean Absolute Error: Average difference between actual & predicted values.
   - less Sentive to outliers.
   - Not differentiable, cannot be used in optimizing algorithms.
   - Not capture overall error as well as MSE
   - Best for where outlier are a concern.

$$MAE = \frac{1}{n} \sum_{i=1}^{n} |y_i - \hat{y}_i|$$

3. HuberLoss: Combines MSE & MAE.
   - uses MAE for small error & MSE for large
   - Remain differentiable for optimization
   - Hyperparameter delta as threshold for transition
   - Regression with outliers

a b c

a P(A) → [P(A) 0 0]

b → [0 P(B) 0]

c → [0 0 P(C)]

Date:

Sun Mon Tue Wed Thu Fri Sat

HL = $\begin{cases} \frac{1}{2}(y-\hat{y})^2 & \text{if } |y-\hat{y}| \leq \delta \\ \delta|y-\hat{y}| - \frac{1}{2}\delta & \text{otherwise.} \end{cases}$

4- Binary Cross entropy Loss: Measure average difference between predicted probability of correct class by 1

- Performance of binary class
- Not suitble for multiclass.

$$\text{Binary CE} = \frac{-1}{n}\sum_{i=1}^{n}\left[y_i \log(\hat{y}_i) + (1-y_i)\log(1-\hat{y}_i)\right]$$

5- Categorical Crossentropy: Generalization of binary CE for multiclass classification.

- Measure the average difference between the predicted probability distribution by the one hot encoded distribution.
- Sensitive to imbalanced class.

$$\text{C-CE} = \frac{-1}{n}\sum_{i=1}^{n}\sum_{j=1}^{m} y_{ij}\log(\hat{y}_{ij})$$

6- Sparse Categorical cross Entropy: Variant of CCE with Sparse targets. only considers loss for predicted class.

- More efficent than CCE
- Not good foor all multiclass classification tasks.
- Best for large number of class

$$= \frac{1}{n}\sum^{n}\log(\hat{y}_i)$$

Date:

Sun Mon Tue Wed Thu Fri Sat

7- Kull back- Leibler Divergence: Measure difference between 2 probability distribution
- used in IR, Style transfer where comparing distribution is crucial
- Not symetric $KL(A||B) \neq KL(B||A)$
- can be undefined in some cases.

$$KL(P||Q) = \sum_i P(i) \log\left(\frac{P(i)}{Q(i)}\right)$$

8- Hinge loss: used in SVM, Penalizes model for misclassification
- Maynot be as smooth as others loss function
- Slower convergence
- Best for SVM where maximizing

$$\text{Hinge} = \max(0, 1 - y \cdot \hat{y})$$

9- Contrastive loss: Used in Similarity learning
- Enforces smaller distance in similar datapoint (+ve pair)
- Enforces larger distance in disimilar datapoint (-ve pair)

- Careful selection of +ve & -ve pairs
- computationally expensive for large datasets
- Best for Siamese networks, facial recognition, IR

$$L(y, \hat{y}) = (1-y) \cdot \frac{1}{2} \hat{y}^2 + y \cdot \frac{1}{2} \max(0, m - \hat{y})^2$$

Date:

Sun Mon Tue Wed Thu Fri Sat

10- Triplet loss: used in triplet networks, minimizes the distance between Achor & positive example while maximizing distance between anchor & negitive example.

- computationally expensive
- Best for Image retreval & facial recognition
- Doesnt require class label. only need to know if 2 points are similar.

$$L(x_a, x_p, x_n) = \max(0, d(x_a, x_p) - d(x_a, x_n) + a)$$

11- Adversial loss: core concept in GANS.

1- Discriminator loss: This measure discriminator's ability to distinguish real & fake data.
often: binary-crossentropy.

2. Generator loss: This is based on how well the generator was fooled the discriminator normally negetive of discriminators loss

- Training instability: finding right balance between the generator & disariminator. Sometimes one overpower other.
- Some cases generator stuck in loop

- Best for GAN: Image, text, generation

Date:

Sun Mon Tue Wed Thu Fri Sat

Optimizers

1- Stochastic Gradient descent: (SGD)

- Updates parameters by taking step in direction of negetive gradient of loss function.

- Sloco convergence, noisy updates.
- Sensitive to learning rate.
- ~~Slow~~
- Simple to implement
- computationally inexpensive
- Best for baseline for comparision with others

$$\theta_{t+1} = \theta_t - \eta \nabla J(\theta_t)$$

2- Mini Batch Gradient Descent:

- Similar to SGD but updates parameters in mini batches instead invidiviually.

- More efficient, less noisy
- Problem in crossing optimal mini batch size.

$$\theta_{t+1} = \theta_t - \eta \nabla J(\theta_t)$$

where $J(\theta_t)$ = loss of mini batch

Date:

Sun Mon Tue Wed Thu Fri Sat

3- Adam:

- Adapts learning rate for each parameter based on past gradients
- faster convergence, better performance
- can be agressive leads to large parameter updates
- good general purpose optimizer

$$m_t = B_1 \cdot m_{t-1} (1 - B_1) \cdot \nabla J(\theta_t)$$

$$v_t = B_2 \cdot v_{t-1} (1 - B_2) \cdot \nabla J(\theta_t)^2$$

$$\theta_{t+1} = \theta_t - \frac{\eta}{\sqrt{v_{t+1}} + \varepsilon} \cdot m_{t+1}$$

- B_1 & B_2 ⇒ Hyper parameter ⇒ for momentum & forgetting
- m_t & v_t : mean & variance of past gradients

4- RMSprop:

- Improves on SGD by adapting Learning Rate of each parameter by using history of squared gradients
- solves the vanishing gradient problem.
- Can be slow to converge
- Might not be as good as adam

Date:

Sun Mon Tue Wed Thu Fri Sat

$$v_t = \beta \cdot v_{t-1} + (1-\beta) \cdot (\nabla J(\theta_t))^2$$

$$\theta_{t+1} = \theta_t - \frac{\eta}{\sqrt{v_t + \varepsilon}} \cdot \nabla J(\theta_t)$$

5- Adagrad:

- Adapts LR for each parameter
- Based on square root of the sum of past squared gradients
- lead to very low LR with each update
- Not suitible for sparse model.
- Rarely used

$$g_t = g_{t-1} + (\nabla J(\theta_t))^2$$

$$\theta_{t+1} = \theta_t - \frac{\eta}{\sqrt{g_t + \varepsilon}} \cdot \nabla J(\theta_t)$$

6- Adadelta:

- Extension of Adagrad
- Reduces the problem of vanishing LR
- only look at for a window of past gradients
- Requires tuning

Date:

Sun Mon Tue Wed Thu Fri Sat

$$\Delta\theta_t = \frac{\sqrt{E[\Delta\theta^2]_{t-1} + \varepsilon}}{\sqrt{E[g^2]_t + \varepsilon}} \cdot \nabla J(\theta_t)$$

7- Adamax:

- Similar to Adam but uses 'infinity norm' of past gradients instead of the mean
- In certain cases large gradients can effect Adam's performance
- Need to find optimal settings.

$$m_t = \beta_1 \cdot m_{t-1} + (1-\beta_1) \cdot \nabla J(\theta_t)$$
$$u_t = \max(\beta_2 \cdot u_{t-1}, \|\nabla J(\theta_t)\|_\infty)$$

$$\theta_{t+1} = \theta_t - \frac{\eta}{u_t + \varepsilon} \cdot m_t$$

8- NAdam:

- Combines Nestrov momentum technique, with Adam,
- Might not provide significant benefit over adam
- Can be used for faster convergence

$$\theta_{t+1} = \theta_t - \hat{g}_t + \beta_1 \cdot \frac{(1-\beta_1) \cdot \Delta\nabla J(\theta_t)}{(1-\beta_1^t) \cdot (\nabla J(\theta_t))^2}$$

Date:

Sun Mon Tue Wed Thu Fri Sat

9- RAdam: Solve the issue in Adam where the gradient of mean become 0
- Rectifies the variance of estimation
- leads to better generalization

10- Ranger: (RAdam + lookahead)

Date:

Sun Mon Tue Wed Thu Fri Sat

AutoEncoders

- Special type of NNs where input is same as output
- Auto encoder consists of
  1- Encoder: compresses the input & produces the code.
  2- Code: Efficient representation of input data Latent vector.
  3- Decoder: Reconstructs the input only using this code.

- Unsupervised learning, unlabelled data.

Why use Autoencoder:

- Useful for dimension reduction.
- powerful feature detector (feature selection)
- can be used for unsupervised pre-training of DNNs
- Capable of generating new data similar to training data, can be used as generative model.

PCA vs Autoencoder:

- Non linear activation & multiple layer,
- Can learn from convolution layer.
- More efficient
- gives multiple representation as output of each layer

Date:

Sun Mon Tue Wed Thu Fri Sat

- There are 2 configuration of Autoencoders:

1- Under complete:
- Dimension of latent space is less than the Dimension of input space
- Model forced to produced a compressed represent of input
- Used for dimension reduction, feature extraction
- less prone to overfitting
- May struggle to capture all variation of input

2- Over complete:
- Dimension of latent space is more than dimension of input space.
- More parameter than undercomplete.
- produces a latent representation of more complex feature than input
- Prone to overfit.
- Regularization, Dropout & sparsity used for genralu

Types of Autoencoders:

1- Vanilla AE: - Basic AE,
- fully connected dense layers

2- Sparse AE: - Sparse constraints
- Good for feature extraction

Date:

Sun Mon Tue Wed Thu Fri Sat

3- Denoising AE: - Trained to reconstruct clean data from noisy input

4- Variational AE: - Probabilistic AE
- learns prob distribution over latent space
- Widely used for generative modeling tasks

5- Convolutional AE:
- Convo layers instead of fully connected layers
- used for image related tasks
- Capture spatial hierarchies of features

6- Recurrent AE:
- Incoprates RNN layer in either Encoder or decoder or both.
- Good for sequential data, time series, text, audio

7- Adversial AE:
- Concept of AE with adversial training
- Encoder, decorder by A discriminator.
- GAN

8- Spiked Slab: - Sparse coding
- feature are binary

Date:

Sun Mon Tue Wed Thu Fri Sat

GAN

- GANs consists of 2 NNs trained simultaneously in competetive manner

1- Generator: Takes random noise as input and tries to generate data sample similar to real data

2- discriminator: Recieves real sample & fake sample from generator. and distinguishes between real & generated

- During training gen & dis are trained iteratively
- trained on minimax game framework. generator minizes the likelyhood of discriminator correctly classifying samples while discriminator maximizes its accuracy

- Application: Image generation, style transfer, translation

Problems: - Training instability: one network over power other

- Mode collapse: generator stuck in loop, generates similar data instead diverse realistic samples

- Hard to find balance in both Network's loss

Date:

Sun Mon Tue Wed Thu Fri Sat

Conditional GAN:

- Both Network recieve conditional information.
- Conditions: guide for generation process.

- Generator produce realistic sample while following conditions.
- Discrimanator learns to distinguish between real & generated data while considering conditions.

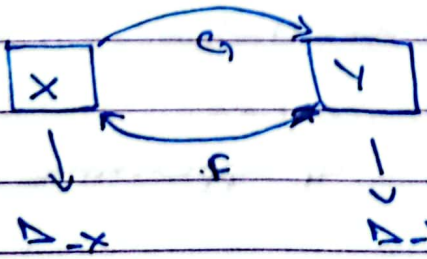
- minimizes 2 losses 1- adversial loss
  2- Conditional loss

- Used when specific characterstics desired in in generated sample

Cycle GAN:

- learns to translate image from one domain to another.

- 2 generators, 2 discriminators.
- ~~generator~~



Date:

Sun Mon Tue Wed Thu Fri Sat

- Cycle consistency loss: Ensures translation process is consistent in both dierections.
  - helps to preserve semantic content during translation.

- used in style transfer, object transfiguration.

Types of GANs:

1- Vanilla GAN: Basic GAN.

2- CGAN: - Conditional information
  - class labels, text description other info
  - Allows for more controlled generation of samples

3- WGAN: - Wasserstein loss as metric
  - Addresses some instebility by mode collapse

4- Cycle GAN: - unpaired image translation
  - cycle consistency loss
  - 4 Networks.

4- ACGAN: - Deep convolution NN as disc by gene
  - More stable by effective

5- Style GAN: - control style by appearance of generated image
  - fine grain control like facial expression

Date:

Sun Mon Tue Wed Thu Fri Sat

7- Star GAN: - Multi domain image translation

8- Big GAN: - Scaled up version of GAN
- Used to improve image quality / resolution.

RNNs

- Short-term dependency: immediate or near immediate relationships between adjacent elements.

- long-term dependency: relationship between elements separated by several steps or time intervals.

- long term dep are particularly because traditional RNN often struggle to capture them due to vanishing gradients

RNNs: Class of ANN designed to handle sequential data by
- Maintains hidden state that capture info about previous inputs.
- Connection that form a cycle allowing them to exhibit temporal dynamic behaviour
- Used for sequences, time series, NLP,

Date:

Sun Mon Tue Wed Thu Fri Sat

Working:

Input: each element of the sequence is represented as a vector

Recurrent connections:

- At step t input vector x_t and previous hidden state h_{t-1} are combined to produce new hidden state h_t

- $h_t = f(W_{hh} h_{t-1} + W_{xh} x_t + b_h)$

W_{hh} & W_{xh} are weight matrices.
b_h is bias f = activation function

Output: hidden stat h_t can be used to generate output y_t which can be used to predict

BPTT: - Back propagates through time
- unfolds the network over time

Date:

Sun Mon Tue Wed Thu Fri Sat

Strengths:

- Capable of modeling sequential dependencies
- Can handle variable length of inputs & can produce variable length of outputs.
- Maintain a hidden state that retains info about previous inputs.

Weakness:

- Vanishing / Exploding gradients
- difficulty in retaining info over long sequences.
- Training is challenging due to nature of sequential data.

LSTM: - Type of RNN &
- Solve vanishing gradient problem
- Capture long term dep in data
- Gated mechanism

1- forget Gate: Takes input for current input state x_t and previous hidden state h_{t-1}
- produced a forget gate vector & determines what percentage of long term memory will be remembered

Cell State= LTM

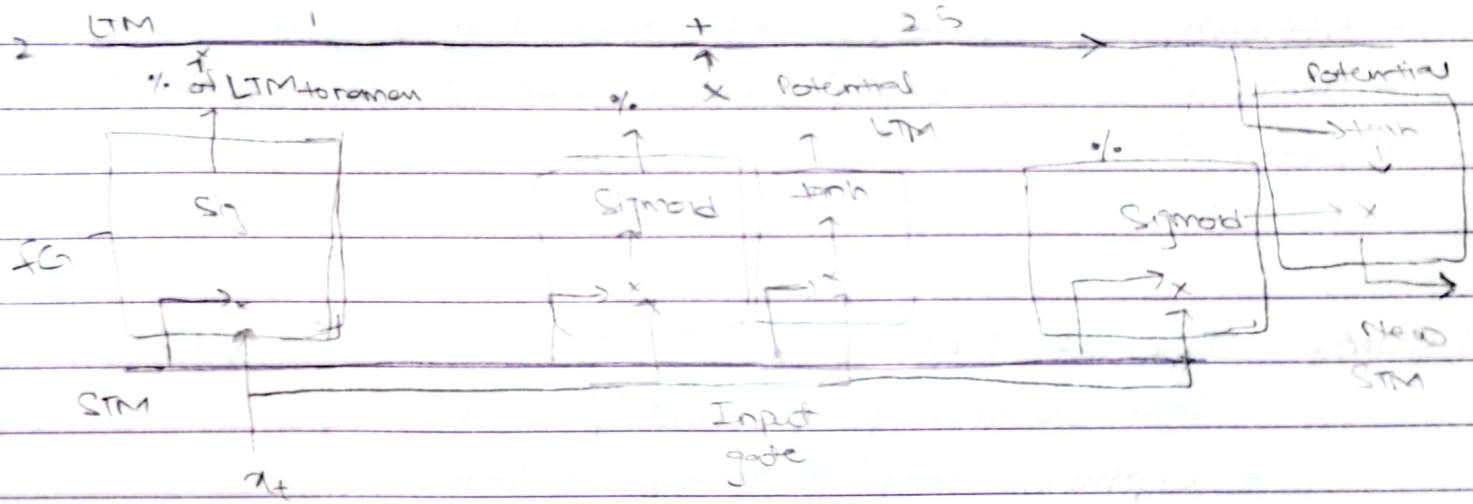
Hidden State= STM

Date:

Sun Mon Tue Wed Thu Fri Sat

2- Input gate: Combines the short term memory with the input to create a potential long term memory And how much percentage of that potential LTM should be add to LTM

3- Output gate: updates the short term memory
- uses new LTM to create a potential STM
- uses input by STM to find how much percentage of potential STM should be passed on.



Strength: - effective in capturing long range dep
- Mitigating vanishing gradient
- Maintain memory cell to store long sequences

Weakness: - more complex than RNNs
- Require more computational power
- Sensitive to noisy input

Date:

Sun Mon Tue Wed Thu Fri Sat

GRU: Type of RNN, similar to LSTM
- Solve vanishing gradient by capture long term dep.
- Simplify architecture of LSTM by combining the forget gate & input gate into single update gate
- And merging cell & hidden state.

update gate: uses input x_t and h_{t-1} with sigmoid to determine how much of previous hidden state to retain and how much to update.
- Determine tradeoff between retaining & updating hidden state.

Reset gate (optional): Some variation include this gate decides how much of past info to forget

Date:

Sun Mon Tue Wed Thu Fri Sat

Strengths:

- Simpler architecture to LSTM . easier to train
- fewer parameters. more efficient
- Effective longterm dep

Weakness:

- May not capture long term dep as good as LSTM
- less control over memory

Bi-Directional RNN: Type of RNN that process in both forward & backward direction

- Allows them to capture information from both past & future context.

- Input sequence is processed by 2 seprate RNNs.
  1. forward 2. backward.
- each independently update their hidden state

- The hidden state of each RNN at each time step captures info from both past & future
- Bi- info is then concatenated to produce final representation.

Date:

Sun Mon Tue Wed Thu Fri Sat

Strength:

- Better understand context
- Improved performance
- Robust to noise

weakness:

- More complex, high comp cost
- access to future may lead to overfitting.

GNN

- Graph neural Network one class of ANN used to inference on data in Graph

- Used to predict node level, edge level, graph level task
- GNN do what CNN failed at

Why CNN failed in graphs:

- Arbitrary size
- complex topology
- unfixed node ordering

GNN operate on graph structure. node → entity
relation' edge → relationship

- Message passing: at each layer nodes exchange information
  - acheived through propagation.
  - Node aggregate its neighbour info to update its own representation

- Node embedding: Representation of graph in low dimension

- Aggregate function: used to combine info from neighbouring nodes