Lec-1 Date No.

*) Programming :- We give data and program as input to the Computer to get a desired output

Machine learning :- We give data & desired output (or) expected output to Computer to get program as an output

*) Supervised :- Training data with desired output values that are provided before training

Unsupervised :- Training data without desired outputs

Semi supervised :- Training data with small desired outputs and iteratively train the ~~data~~ model using the most confident predictions

→ Train initial ~~data~~ model on a small set of labeled data.
→ Use this model to generate predictions on a larger set of unlabeled data.
→ Select the most confident predictions and add them to labeled data set
→ Retrain the model on expanded data set
→ Repeat step 2 and 4 until desired output is achieved

*) Types of learning :- i) Active learning ii) Federated learning iii) Reinforcement learning

*) Active learning :- Train data with and without labels; Interactive labeling of the training data by asking a human expert to label the least confident data samples.
Ex :- Pool-based active learning ; Diversity sampling ; Stream-based active learning

*) Federated learning :- Training data is localized ; learn a central model using distributed local data without sharing data

MATRIKAS

Date No.

*) Reinforcement learning:- an agent learns to interprets its environments by interacting with it through actions are awarded (or) penalized.

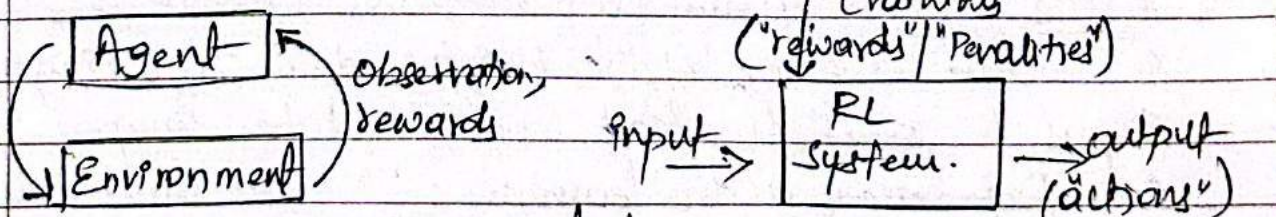
Examples of Supervised learning:-

*) Predict whether a patient, hospitalized due to heart attack, will have a second heart attack.
Identify strongest predictors of heart attack
Predictions based on many data modalities:- demographics, diet, clinical measurements, clinical history.

*) Predict the price of stocks in 6 months from now, based on i) Performance of the company (ii) other economic metrics (iii) other exogeneous variables.

*) Identify handwritten numbers in a digitalized image & identify animals in image

*) Examples of Reinforcement learning:- Define the measures to be taken by the public health authorities under a pandemic situation.

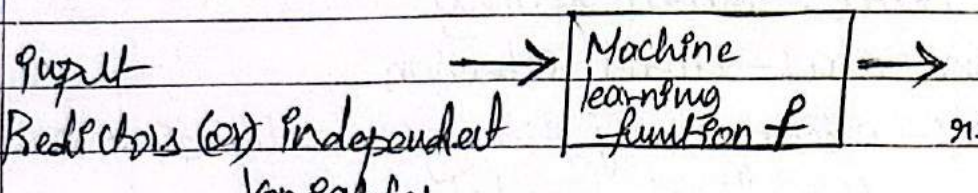


*) Examples of Federated learning:-

Given data from distributed hospitals learns a central model that can propose the optimal patient treatment without sharing any data.

MATRIKAS

Classical learning setup:- Attributes Date No.
Categorical, discrete, ordinal, numerical



Classification:- It is task of learning a target function f that maps an input (attribute set) x to one of the predefined class labels y

Regression:- It is task of learning a target function f that maps an input (attribute set) x to a value in range of real values y.

*) The Bias variance decomposition:- The metric to optimize is Mean Squared Error (MSE)

*) How good is f:- If training error drops to zero as model complexity increases lead overfitting (not good)
i) High bias → underfits (ii) High variance → overfits
(iii) low bias & low variance → good; This can be achieved by selecting appropriate model complexity, using regularization techniques & selecting its right features: In order to achieve a goodness of Machine learning function, can be evaluated based on its ability to model relation b/w the input features & targeted variables & bias, variance properties

*) Model selection & assessment:-
Model selection:- estimating the performance of different models inorder to choose the best one.
Model assessment:- having chosen a final model, estimating its prediction error on new data

MATRIKAS

*) Data divided into three parts :- Test, validation, train
i) Training data :- Model fitting
ii) Validation data :- Model selection
iii) Test-Data :- assessment of the generalization error of the final model

*) Hold out and subsampling :- Two common techniques to create traing and test sets-
Holdout :- Reserves 50% for training 25% each for test & validation.
Random subsampling :- Repeated holdout
Stratified subsampling :- Preserves class balances in the samples.

*) K-fold cross validation :-

1) Divide data into K equal parts (or) folds.
2) Iterate K times :
i) Take the k^{th} fold as validation set, remaing as training set
ii) Train the model using training data
iii) Test the model using validation set
3) Average the accuracy of all K iterations & use it as the performance metric.

*) Comparing learning curves :-

High Bias :- As training examples increases, training error increases too ; model is not fitting the data with an appropriate curve. Bias leads to comparable errors in cross validation & training. Adding more data does not help.

MATRIKAS

Increase no. of features add more complex features. Date No.

High variance: Test error grows slowly but well within the desired range as no. of training examples increases. High variance leads to overfitting & hence high test error the model is forced to learn more generalized properties. get more data and reduce no. of features.

*) K-fold cross validation pitfall: Data leakage: If information from test set is inadvertently include in the training set during K-fold cross val. This can be happened if the data preprocessing (or) feature selection is done on entire dataset, rather than just the training set

*) Nested Cross validation: It involves an outer loop of K-fold cross val for evaluating the generalization performance of the model & inner loop for tuning the hyperparameters of the model. The inner loop is performed on each outer loop and is used to select the best hyperparameters for the model.

We deploy the feature set obtained by applying the winning procedure to the whole dataset.

*) Bootstrapping: Create B replications of same size of training set of size N by selecting examples uniformly with replacement. ~~To do on each replacement B~~

- Train on each replication b
- Test on whole training set

Leave One Out Bootstrapping: Train on each replication b Test each sample not in b

Problems: #) The avg no. of distinct observations in each bootstrap sample is about $0.632N$; it's bias roughly behaves like as two-fold cross val; if the learning curve has considerable slope at sample $N/2$, leave one out bootstrap will be overestimate of the true error.

MVIRTKAS

*) .632 Bootstrap:- extension of leave one out Bootstrap. pulls the leave one out Bootstrap estimate down towards the training error rate, and hence reduces its upward bias.

lec-2

Examples of supervised learning:- i) quantitative ii) categorical
i) Quantitative :- "stock price ; categorical :- heart attack "
↳ outcome measurement ↳ Predict based on a set of features.

*) Regression when Predict quantitative outputs.
classifications :- " : categorical outputs (qualitative).

*) Regression :- Statistical measure that attempts to determine the relation b/w dependent variables Y & independent variables X; used to predict next values of dependent variable Y from the values of & Independent variable X
→ Dependent variable Y :- is the variable whose value changes as the consequence of change in other values in system.
→ Independent variable x :- $x_1, x_2, \ldots, x_n$ are input to the system & may take diff values freely. Also called as predictors

*) Simple Linear Regression :- Mathematical relation b/w the variables X & Y is a linear relationship.
X :- cause ; Y :- effect in cause-and-effect relation
Model relation b/w two variables by fitting a linear equation to the observed data.

*) How do we ~~find~~ fit linear model ?
fit.

MATRIKAS

We attempt "best fit line" by minimizing the actual and predicted values. But positive differences could offset negative ones. So we take the squared differences.

*) Ordinary least squares (OLS) minimizes the Residual Sum of Squares (RSS) of observed values to the straight line.

*) When there is single input variable it is referred to simple linear regression, when there are multiple input variable then it is Multiple linear Regression.

*) Cognitive functions:- The abilities of AI model to perform tasks that are typically associated with human cognition, such as perception, reasoning, learning, decision making, language processing.

Gauss-Markov theorem:- When all assumptions hold, OLS will produce better coefficient estimates compared to all other linear model estimation methods.

*) Gradient Descent:- It's an optimization algorithm that finds the linear regression coefficients iteratively. This is used by model to reduce the loss function & achieving the best fit line. To check if it's working is to check the error rate decreases as for each iteration.

Working process of Gradient decent:-

i) It assigns an initial Random value to the coefficients

ii) It iteratively updates the coefficient values proportional to the negative of the gradient of the function.

iii) Finally it finds a locally (e.g. global) minimum for the loss function.

How good the fit is?:- The coefficient determination can be used to determine how well the models fits the training data. Denoted by R^2; $R^2 = \frac{\text{Variance explained by model}}{\text{Total variance}}$

MATRIKAS

Problems with fitting the data:- Overfitting, Underfitting

Overfitting:- the model models the training data too well; If we have large no. of features & the test score is worst than the training score

Underfitting:- Model that can model the training data nor generalize to new data. If we have fewer features and the test score & training score is poor for both.

**) Regularisation:- Not satisfied with Ordinary least Squares (OLS):- (i) Low performance (ii) Interpretation. Regularising the coefficient estimates ⇒ i) Ridge Regression (ii) Lasso Regression

Ridge Regression:- Uses a penalty term, called a Regularisation parameter to limit the complexity of the model, which prevents overfitting. It works by adding the sum of the squares of the coefficient to the loss function which is then minimized during training. The Penalty term adds small bias to the model.

Lasso Regression:- It adds absolute values of the coefficients. This has the effect of shrinking some of the coefficients to zero, effectively performing variable selection and reducing no. of features in the model.

Conclusion:- Ridge adds a penalty term the limits the size of the coefficients while lasso adds a penalty term that eliminates some of the coefficients all together.

MATRIKAS

Example of Ridge Regression:- Average credit card debt Balance is measured for each individual. We have information on several predictors: age, no. of cards, income, credit rating

*) Evaluation Metrics:- 1) Mean Squared Error (2) Root Mean Squared Error (3) Mean Absolute Error (4) Mean absolute percentage error

1) MSE:- Is calculated as the mean (or) average of the squared differences between predicted & expected target values in a dataset. The higher the MSE the larger the error in the model.

2) RMSE:- The square root of the MSE, which means that the units of the error are the same as the units of the target value that is being predicted. The higher the RMSE the larger the error in the model.

3) MAE:- Average of the absolute error values and like RMSE the units of the error score match the units of the target value that is being predicted.

4) MAPE:- The percentage equivalent of MAE. Each residual is scaled against the actual value.

Name	Residual operation?	Robust to outliers?
MSE	Square	No
RMSE	Square	No
MAE	Absolute value	Yes
MAPE	Absolute value	Yes

*) Logistic Regression:- Statistical model that uses a logistic function to model binary dependent variable. It models the probability that Y belongs to a particular category / class. The output of Logistic regression problem can be only 0 and 1.

How to fit?:- We use a method called "Maximum likelihood"

MATRIKAS

Types of logistic Regressions- Date No.

i) Binary logistic Regression :- The categorical response has only two possible outcomes.
Example :- Spam or Not

ii) Multinomial logistic Regression :- Three or more categories without ordering.
Ex:- predicted which food is preffered more (Veg./Nonveg./vegan)

iii) Ordinal logistic Regressions:- Three (or) more categories with ordering.
Ex:- Movie rating from 1 to 5.

Types of logistic Regressions- Date No.

i) Binary logistic Regression :- The categorical response has only two possible outcomes.
Example :- Spam or Not

ii) Multinomial logistic Regression :- Three or more categories without ordering.
Ex:- ~~Predicted~~ predicted which food is preffered more (Veg / Nonveg / vegan)

iii) Ordinal logistic Regressions :- Three (or) more categories with ordering.
Ex:- Movie rating from 1 to 5.

Lec-4 Deep learning :- Deep learning algorithms attempts to learn feature representations by using a hierarchy of multiple layers.

ANN :- (*) Single layer Perceptron (*) Single layer of artificial neurons, with each one connected to every input feature.
(*) Summation function Ex:- dot product.
(*) Bias used to shift activation function ($f(x*w) \geq 0$) to either right (or) left.

MLP :- More complex problems may require intermediate layers.

Activation function :- Non-linear ; network with one hidden layer can learn perfectly any classification problem.
Problem is to find the proper no. of neurons & learn their weights.

MATRIKAS

Linear activation function :- Not flexible enough to capture complex relationships b/w data variables.

Non-linear activation :- Makes it easier for the model to adapt as data varies.

Sigmoid activation :- Convenient for models that predict range (or) probabilities. Ex :- Range : (0, 1). Softmax function is a more generalized logistic activation function which is used for multiclass classification.

tanh : activation :- Negative input is mapped to strongly negative values and zeros inputs is mapped near to zero. Takes negative values. Range : (-1, 1).

ReLU activation :- Both the function and its derivative are monotonic. Problem is negative inputs turns to zeros immediately so can not map negative values appropriately.

Leaky ReLU Function :- Leak helps to increase the range of the ReLU function. Usually, a is 0.01 (leaky) or it may have any other values. Monotonic in nature both ReLU functions & derivatives

Forward Propagation :- In here, the input features are fed into the first layer of neural network which process the inputs and passes the result to next layer. Each layer of the network consists of multiple neurons which calculates the weighted sum of their input and apply an activation function and gives an output. The output of each neuron is passed as input to the next layer, until the final layer of network is reached. The output of final layer represents the predicted output of the neural network for given input features.

let x be the input features ; W & b → weights & bias of neuron in each layer of the network ; f activation function

ŷ output of final layer of neural network.

output of first layer → $z_1 = W1 * x + b_1$

$a_1 = f(z_1)$

output of second layer → $z_2 = W2 * a_1 + b_2$

$a_2 = f(z_2)$

Date : No.

Back propal:- The idea is to propagate the error δ back to all neurons. After forward propagation, the predicted output of neural network is compared with true output to compute error. Then error is propagated back through the network to update the weights and bias of the neurons in the output layer.

Regression loss functions:- Huber loss: If the absolute difference b/w the actual & predicted value is less than or equal to a threshold value, δ, then MSE is applied; If error is sufficiently large, then MAE is applied.

0-1 loss:- Counts how many class labels the predictions are correct. It is not differentiable hence cannot optimize efficiently. Consider alternative functions that can be optimized also referred to as surrogate loss function.

Binary Cross entropy:- Classification neural networks outputs a vector of probabilities $\hat{y}_i$.

Categorical Cross entropy:- Classification neural networks output a vector of probabilities $\hat{y}_i$:

$g \longrightarrow$ Softmax $\longrightarrow$ Cross-entropy loss

Regularization (L_2 or) L_1:- L_1 not prefered as it will end up compressing the network L_2 to make the derivate computation easier.

Regularization term adds penalties big weights, added to the objective. Weight decay value determines how dominant regularization is during gradient computation. Big weight decay coefficient → big penalties for big weights.

MATRIKAS

Data augmentation:- Increase the size of the training data wherever possible. May result in improved predicted performance. Can reduce cost of collecting and labelling data. Enables rare event prediction. Prevents data privacy issues

Early stopping:- Use validation error to decide when to stop training. Stop when the ~~monitored~~ monitored quantity has not improved after n subsequent ~~epochs~~ epochs n is patience

Validation error too high:- Training error zero, val error remains high. Overfitting. model too complex: reduce layers and hidden nodes apply early stopping & dropout. Set hard limit on training performance. Increase training examples use regularization to penalize weights if weights get too large

Stochastic Gradient Descent:- Randomly sample a small batch of data from entire dataset at each iteration and compute gradient of the cost function with respect to the parameters of the batch. The parameters are then updated using the computed gradient and the process is repeated with another randomly sampled batch

less-CNN: We need to detect the small patterns

Multilayer perceptrons:- fully connected networks, each neuron in one layer is connected to all neurons of next layer. fully connectivity makes them prone to overfitting.

Hierarchical patterns:- Assemble patterns of increasing complexity. using smaller & simpler patterns captured by filters

MATRIKAS

Date No.

Whole CNN:-

Image → Convolution → MaxPooling → Convolution → MaxPooling → Flattened → fully connected feed forward Network

(Can repeat many times)

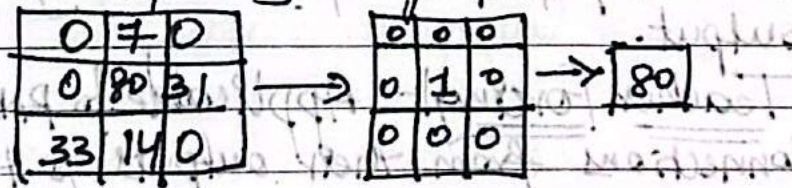
* Maxpooling takes the max value in a 2x2 (non overlapping)
* Avg. Pooling takes the avg in a 2x2 (non overlapping)

* Dark Background :- maxpooling selects the brightest pixels
* Light Background :- minpooling selects the darkest features
* Image Smoothning :- average pooling.

* Why pooling? → Subsampling pixels will not change the object. We subsample the pixels to make image smaller. fewer parameters to characterize image.
Reduced no. of connections; Shared weights on the edges; Maxpooling further reduces the complexity.
* Flattening image refers to convert image from 2D to 1D by stacking the pixel values row by row (or) column by column. It is often done as preprocessing step before feeding into a ML model.
* ReLU is applied after each convolution: faster & moreover effective training by mapping the negative values to Zeros and maintaining positive values.
* Final layer: Softmax on the previous layer, yielding the probability of output being of a particular class.

MATRIKAS

Training a CNN: Two steps:- forward phase:- the input is passed through the network. each layer will cache any input and intermediate values needed for the backward phase
Backward phase:- Gradients are backpropogated and weights are updated.
Cross entropy loss is used as the loss function.
Training a CNN: maxpooling:- A maxpooling layer cannot be trained as it doesnot have any weights. We can propogate gradients by doubling the width & height of the gradient by assigning each value to where the original max value was in the corresponding pooled block.
Training a CNN: Convolution:- Changing the filter weights affect the entire output image for that filter. Every output pixel uses every pixel weights during convolution. Increasing any of the filter weights by 1 would increase the output by the value of the corresponding image pixel. The derivative of a specific output pixel (i,j) with respect to a specific filter weight is just the corresponding image pixel value.



Deploying a Pretrained CNN:-
*) Pretrain a neural network model on a source dataset
*) Create a new neural model by:
- copying the pretrained model
- removing the output layer
- adding an output layer
- initializing randomly the model parameters of the new output layer
*) Train new model on the target dataset
*) The output layers will be trained from scratch, while the

Parameters of all other layers are fine-tuned based on the parameters of the source model.

Date No.

Lec-6:- RNN : Basic RNN tasks :-

*) Predict the future from the past

*) Map the past into a fixed-length vector $h^{(t)}$

$(x^{(t)}, x^{(t-1)}, x^{(t-2)}, \ldots, x^{(2)}, x^{(1)}) \longrightarrow h^{(t)}$

*) How much of the past should one store/model in $h^{(t)}$

Three common vanilla RNNs :-

*) Version 1 : *) output at each time step

*) Recurrent Connection b/w hidden units

*) Version 2 : *) output at each time step

*) Recurrent Connection only b/w from the output at one time step to the hidden units at the next

*) Version 3 ; *) Recurrent Connection b/w hidden units

*) Read an entire sequence & produce a single output

Version 2 :- no need to compute the output for previous time step first why? We can give the actual output.

*) Teacher Forcing :- Applicable to RNNs that have connections from their outputs to their hidden states at the next time step. At training time, we feed the correct output $y^{(t)}$ drawn from the training set as input to $h^{(t+1)}$

Parroting

During training time, we will feed in the intended output to the next hidden state and teach it how to produce the output time t & compare it with intended output of time t. We can add noise to it cause it's not always be the exact output.

MATRIKAS

Input output Scenarios:

Date No.

- Single - Single → Feed-forward Network
- Single - Multiple → Image captioning (Image only)
- Multiple - Single → Sentiment classification
- Multiple - Multiple → Translation ; Image captioning (Previous words too).

Injecting the Image to RNN:

- Init-inject :- Image features are used to initialize the hidden state of the RNN. The image features are passed through a fully connected layer to obtain an initial hidden state, which is used to initialize the hidden state of RNN.
- Pre inject:- The image features are concatenated with the input sequence at each time step before being fed into the RNN. This allows RNN to take the image features into account when processing the sequence.
- Par-inject:- The image features are passed through a parrakel network that runs alongside of RNN. The output of the parrakel network is then concatenated with hidden state of RNN at each time step.
- Merge inject:- the image features and the hiddenstate of the RNN are combined using a merge layer, such as concatenation before being passed through a fully connected layer to obtain the new hidden state.

MATRIKAS

Date No.

Back propagation through time (BPTT)

- Treat unfolded network as one big feed-forward network
- This network takes in the whole sequence as input
- We compute the gradients through the usual Back propagation
- We update ~~weights~~ shared weights

Turncated BPTT:-

- Run forward & backward propagation through segments of the sequence instead of the whole sequence
- Back propagation through a segment & make a gradient step on the weights
- Next Batch :- Still have the hidden state from the previous Batch & carry the hidden state forward.

Vanishing Gradients:- The effect of change in L to the weight in some layers becomes so small due to increase model complexity with more hidden units, that it becomes zero after a certain point.

Exploding Gradients:- a large increase in the norm of the gradient during training due to an explation of long-term components, can result in the gradients grow exponentially.

Gradient scaling:- normalizing the error gradient vector such that the vector norm equals a defined value, such as 0.1

Gradient clipping:- forcing the gradient value to a specific minimum (or) maximum value if the gradient exceeds an expected range.

- In some cases large ~~num~~ range of error gradients is permitted in the output layer compared to hidden layers.

MATRIKAS

Gated RNNs:- Creates paths through time that have derivatives that neither vanish or explode. Most effective sequence models used in practical applications: i) Long short-term Memory ii) Gated Recurrent Unit.

* Allows the network to accumulate information over a long duration
* Once this information is used the neural network can forget old state
* Instead of when to clear the state manually, neural network learns to decide when to do it.

LSTM:- Uses this idea of constant error flow to create a constant error carousel which ensures that gradient do not decay.
Memory cell that acts as an accumulator over time. Instead of computing a new state as a matrix product with the old state, it rather computes the difference b/w them. Gradients are well behaved

Structure:- Each cell has the input and output of an ordinary recurrent network but also more parameters and a system of gating units that controls the flow of information.

* The sigmoid layer outputs numbers between 0-1 which determines how much each component should be let through
* x gate: point wise multiplication
  + gate: Point wise addition

Gated Recurrent unit:- * A single gating unit that combines the forget and input into a single update gate.
* It also merges the cell state and hidden state.
* This is simpler than LSTM
* There are many other variants too.
* Has fewer parameters than an LSTM & has been shown to outperform LSTM on some tasks

MATRIKAS

LSTM vs GRU:- Date No.

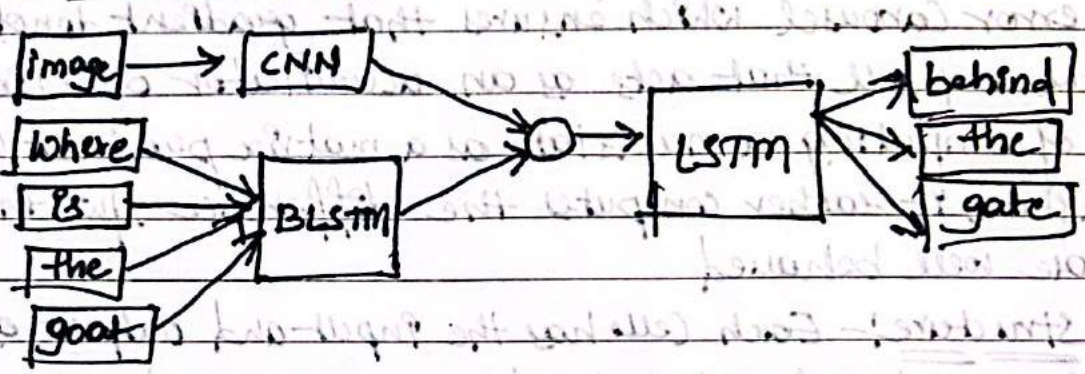
*) GRU uses lesser training parameters and hence needs less memory.

*) GRU executes faster than LSTM whereas LSTM is more accurate on larger datasets and longer sequences.

*) One can choose LSTM when dealing with long sequences and accuracy is concerned.

*) GRU is used when we have less memory availability.

Multimodal learning:-



Lec 7:-

Autoencoders: Neural networks that are designed to learn encodings/features spaces from data. How? By reproducing the input from a learned encoding.

Encoder:- Compresses the input into a latent space of usually smaller dimensionality.

Decoder:- Reconstructs the input from the latent space with as close as possible.

*) Unlike PCA we can achieve the non-linearity by using activation functions. It has been shown that without activation function achieve PCA capacity:
$x \rightarrow f(x) \rightarrow z \rightarrow g(z) \rightarrow \hat{x}$

Given data x and we would like to learn function f (encoder) and g (decoder) where $f(x) = s(Wx+b) = z$ & $g(z) = s(W'z+b') = \hat{x}$

Z is a latent space representation, S is a non linearity function

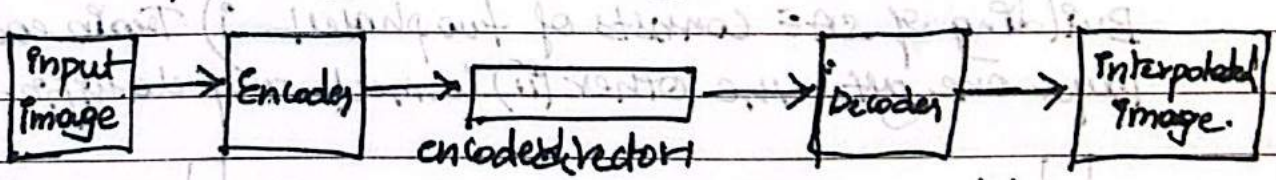
*) Train the autoencoders using gradient decent as any other NN using traditional squared error loss function. If input is interpreted as bit vectors then cross entropy

*) Two types of autoencoders :- Undercomplete, overcomplete.

Undercomplete:- Hidden layer is smaller than input layer. Hidden nodes capture good features of the training set distribution.

Over Complete:- Hidden layer is larger than input, no compression in hidden layers, no guarantee that nodes will capture good features, a latent space of higher dimensionality models a more complex distribution.

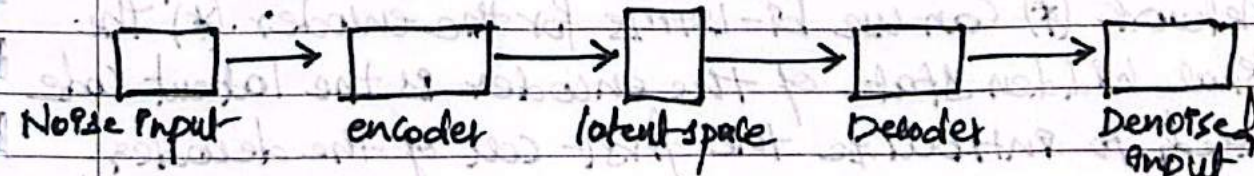
Image fading:- Take two random images and generate the encoding by passing them through encoders. The encoding are flat vectors of length 128 each. Generate interpolations b/w the encodings. Generate images for each of the interpolation by passing them through decoder.



Denoising autoencoder:- A more robust model.

Encode the input & do not mimic the identity function

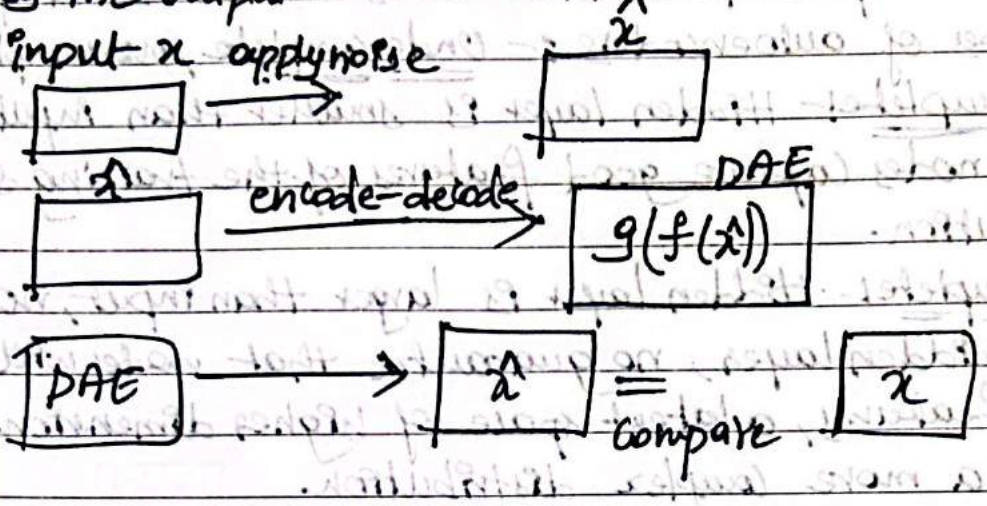
Undo the effect of corruption process applied to the input.



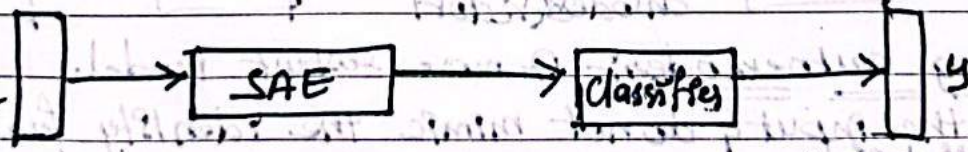
*) Instead of trying to mimic the identity function by minimizing, A DAE instead minimizes

MATRIKAS

Denoising autoencoders — process:- Using the corrupted No.
version of the input data as input to the network, & are trained to produce the original uncorrupted input as the output.



Stacked auto encoders:- we want harness the feature extraction quality of an AE; we can build a deep classifier, whose input is the output of a SAE.
Benifit:- Our deep models are not randomly selected but rather smartly selected.
Building of SAE consists of two phases:- i) Train each AE layer one after the other (ii) connect any classifier



Autoencoders with RNNs:- *) LSTM for encoders & decoders network (*) Can use Bi-LSTMs for the encoder. (*) the final hidden state of the encoder is the latent code used to initialize the first cell of the decoder.
*) At training time: the decoder receives the ground truth previous word as input at each time step
*) At inference time:- The decoder receives the predicted word at the previous time step.

MATRIKAS

Date No.

Limitation of autoencoders:-

*) language translation: encoder + decoder learn the translation tasks together.

**) Encoder: Reads & encodes a source sentence into a fixed-length vector; Decoder: outputs a translation from the encoded vector

*) A Neural network needs to be able to compress all the necessary information of a source sentence into a fixed length vector this may make it difficult to cope up with long sentences → Solution is Attention Mechanism.

Attention Mechanism [Bahdanau 2015]:- In autoencoders, attention Mechanism is can be used to help autoencoders learn to focus on the most important parts and ignoring the irrelevant parts. This also helps autoencoder to identify patterns in the input data, which can be used to generate more meaningful representations of the data. The autoencoder does not encode a whole input sentence into a single fixed length vector. It encodes the input sentence into a sequence of vectors and choose a subset of these vectors adaptively, while decoding the translation.

Decoder:- implements an attention mechanism by soft-searching through annotations generated by the encoder.

Encoder:- Uses a BiRNN to capture richer context from the input sentence by producing annotations that are concatenations of the forward & Backward ~~propagation~~ hidden states at each step.

How does decoder pay attention to annotations?

It uses an alignment model with a trainable feed forward network to score how relevant each annotation is for the decoders hidden state. It calculates weights of each annotation using the alignment score. The decoder receives

as input at time point i a context vector that is a weighted sum of the annotations.

Date . . . . No.

Decoder at step i:- takes the previous state, current state to generate the current hidden state. Which is why we use previous state to calculate the alignment scores.

Transformers:- First to relying entirely on self attention to compute the representations of it's input & output without using any RNN & CNN units.

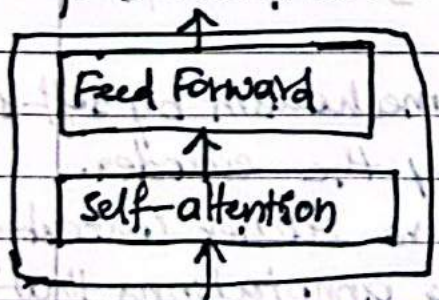
Encoder:- All are identical in structure but don't share similar weights.

*) self attention layer helps the encoder look at other words in the input sentence as it encodes a specific word.

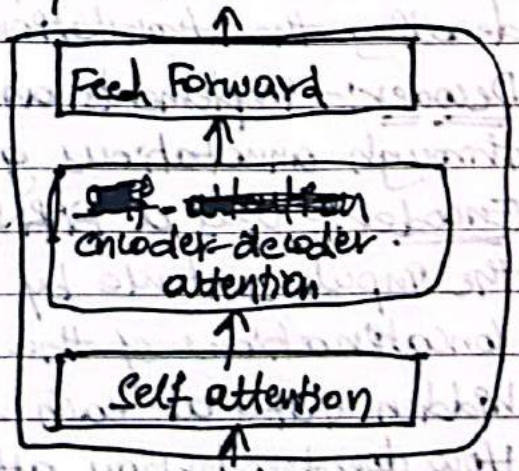
*) Output of self attention layer is directly fed into a feed forward neural network.

*) The same feed forward network is independently applied to each position.

Decoder:- has both the same layers but in b/w them it has another layer (attention layer) that helps decoder focus on relevant parts of input sentence.



Encoder



Decoder

MATRIKAS

Encoding:- *) Each encoder receives a list of vector, each of 512 bits
*) Each embedding flows through each of the two layers of encoder
*) These paths have dependencies when in self attention layer, but not through feed forward NN, which is the same for all input vectors
*) Hence the latter can be executed in parallel.

Self attention:- Consider a statement: "The animal didn't cross the street because it is too tired". The self attention layer solves this by associating "it" with "animal".

Step 1:- Define three vectors for each input embedding by multiplying with three matrices. (i) Query vector (ii) Key vector (iii) Value vector: the default dimensionality of these three vectors is 64.

Step 2:- Self attention score: The dot product of the query vector with key vectors with respective word. Self attention score for the word in the position #1 is $q_1 * k_1$ & second is $q_2 * k_2$, etc. The score defines how much focus to place on other parts of input sentence as we encode a word at a certain position.

Step 3-4:- Normalization: Divide the scores by 8, this leads to have more stable gradients. Then pass the result through a softmax operation.

Step 5-6:- Scaling & Summarization:- *) Multiply each value vector by softmax score. Maintain the values of the word(s) we want to focus on & drown out irrelevant word(s). Output of self attention layer at each position: sum of the weighted value vectors.

MATRIKAS

Multiple heads:- It focus on different positions. It offers attention layer multiple representation subspaces. It tries to catch different definitions of relevance. If we add all attention head to the picture, it is hard to interpret.

(*) How do we account for the order of the words?
- The transformer adds a vector to each input embedding.
- These vector follows a specific pattern that the model learns, which help it to determine the position of each word in the sequence.
- Adding these values to the embedding provides meaningful distance b/w the vectors once they are projected into Q/K/V & during dot product attention.

(*) Layer Normalization:- So that each feature has the same average and standard deviation. Normalizes the input across the features.

Decoder:- (*) The encoder starts by processing input sequence.

(*) The output of the top encoder is then transformed into a set of attention vectors K & V.

(*) These are then used by each decoder in its "encoder-decoder attention" layer helping the decoder to focus on appropriate places in the input sequence.

(*) Encoder-decoder layer:- Works just like multihead self attention, except it creates its query matrix from the layer below it, & takes the key & values matrix from output of the encoder stack.

Self attention layer:- only allowed to attend to earlier positions in the output sequence by masking future positions before the softmax step in self attention calculation.

The final layers - (*) The decoder stack outputs a vector of floats

(*) How do we turn that into a word?

- final layer followed by a softmax layer

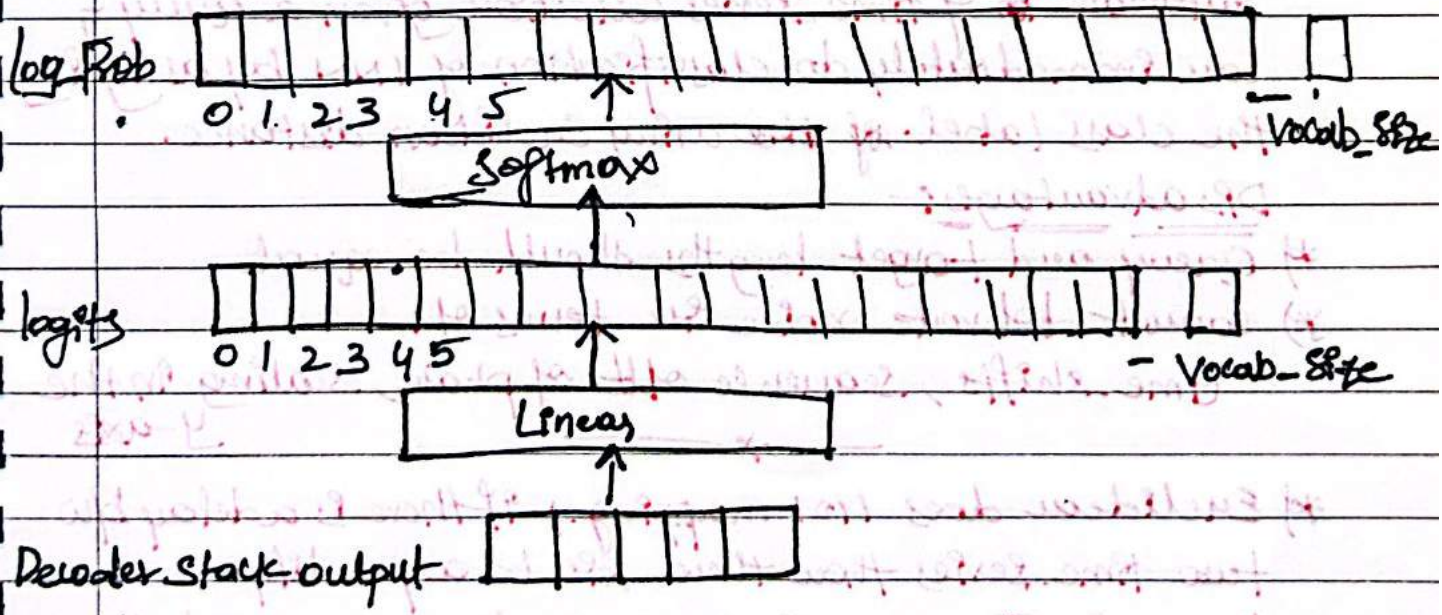
Linear layer - a simple fully connected neural network that projects the vector of the stack of decoders into a large vector called logits vector.

(*) Assuming the model knows 10,000 unique words learned from the training set

(*) Logits vector: 10,000 cells

(*) The softmax layer then turns the scores into probabilities

(*) the cell with highest probability is chosen, and the word associated with it is produced as output for this time step.



Loss function - translate "merci" to "thanks".

*) use one-hot encoding to represent the output vocabulary

*) Cross-entropy loss (or) Kullback-Leibler divergence

MATRIKAS

*) How do we use transformers for classification?
*) Add an FNN and a softmax layer on top.

Lec-8 Time Series

Date : No.

Data Series:- Sequence of points ordered along some dimension

*) How do we compute the distance b/w two time series? How do we define a similarity?

The Lp norm/Euclidean distance and after apply K-means as clustering:- View each time series as a point in the n-dimensional space (n=length); Define the distance b/w time series X and Y

The Euclidean distance:- Euclidean distance b/w two time series & ranking them according to the distance low to high

And the disadvantages are query & target-lengths should be equal

A distance function, then we can perform k-nn search in database of time series collection given a query & can immediately do classification of 1NN by assigning the class label of the using Euclidean distance.

Disadvantages:-

*) Query and target lengths should be equal
*) Cannot tolerate noise in terms of:
- time shifts, sequence off of phase, scaling in the y-axis

*) Euclidean does 1 to 1 mapping., if there is a delay b/w two time series then there will be a penalty.

Dynamic Time wrapping:- DTW allows sequences to be stretched along the time axis
- Insert 'stutters' into sequence
- Then compute Euclidean distance
- If does it in minimum way does that cost of the matching is minimized using Dynamic programming

MATRIKAS

Computing DTW:-

Date No.

- DTW computed by dynamic Programming (DP)
- Given two sequences. $X = \{x_1, x_2, \ldots, x_n\}$; $Y = \{y_1, y_2, \ldots, y_m\}$
- We an $n \times m$ DP matrix f to store the scores
- Computing the difference of the two points that are corresponding to the cell (Pairwise difference of two time series points) and add the minimum value looking at three adjacent cells & the final distance

Equations $f(i,j) = \|x_i - y_j\| + \min \begin{cases} f(i,j-1) \\ f(i-1,j) \\ f(i-1,j-1) \end{cases}$ is the value of the top right cell.

- Each cell (i,j) is a pair of indices
- Cell values are computed using some norm, $p=2$
- first cell $= (x_1 - y_1)^2$; We then moved diagonal then the second cell $= (x_2 - y_2)^2 + (x_1 - y_1)^2 \Rightarrow$ Pairwise (adding the previous cell value to the second cell when it's computed) So this identical to Euclidean but without square root. that means we are doing just one to one mapping. When it is diagonal then the equation will be $f(i-1, j-1)$.
- Actually DTW allows any path not the diagonal path.
- Pairwise distance of the particular cell and then look on to the left side of the cell and downwards to cell & check the values in the cell and finds the smallest value and propagate it to add to the cell value. If we go from the left we are applying a y-stutter if we are go from the bottom we apply x-stutter that means we are repeating the same y value and x value respectively if we go diagonal we are doing one to one mapping, not repeating anything. The top-right cell contains final output & it is the DTW score. If for each cell we store also their path (or) the transition (the adjacent cell having smallest value)

We can propagate back to the initial cell.

Properties of DTW:- i) Boundary Conditions (ii) Continuity (iii) Monotonicity.

Boundary conditions:- The first two points of time series and the last two points time series should match no matter what.

Continuity:- We should not skip any time point. Each point should atleast match other point other time series & viceversa.

Monotonicity:- The path should always go to the right and upwards. It cannot go back (or) down.

*) What happens if it go backward (or) down?

- there will be a crossover in the matching of points

*) How can we speed DTW?

- Global constraints:- Applying global constraints (or) apply bands around the diagonal.

*) We can apply how the working path can move. It can only move inside band & it cannot go outside the band and we have band of fixed length of "r" and we can also have band that is rigid at the beginning and as it moves it becomes more liberal and again it becomes rigid. "Itakura" fits better and mostly used

Pros & Cons:- Advantages:- *) Target & Query lengths may not be equal length

*) Can tolerate: time shifts, sequences out of phase, scaling in the y-axis (how?) (Z-normalization)

Disadvantages:- *) Computational complexity; may not be handel missing or noisy values;

*) If we have noise, noise will go through the matching and it propogate. add it to the sum of the Euclidean distance

MATRIKAS

Z-normalization:- Needed when interested in detected trends and not absolute values. If we want to detect trends. If we care only about trend not fact that they are different in y axis, what we can do is Z-normalize the whole time series or each one of them. Take one time series, compute the mean & standard deviation and for each and every value in time series we subtract the mean and divide by standard Deviation.

* We observe patterns are similar but we are using two different scales for two patterns for two time series. If we immediately run Euclidean (or) DTW we will get high values. If we apply Z-normalization it comes to the same scales.

If care about patterns then Z-normalization is must.

~~Longest Subsequence~~

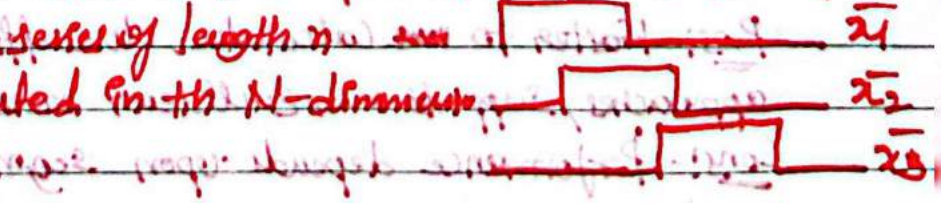
Longest Common Subsequence:- Using Dynamic programming but what it does, it checks the pair of points. If the diff b/w the pair of points is less than a small value then counts 1 to the match. So we add 1 to the score. If it is higher than that then we do not add anything to the match but we propogate the value from either the left (or) bottom. In this we can avoid which have high difference.

Time Series Summarization techniques:-

Piecewise Aggregate Approximation (PAA):- Represent the original time series of length n as a sequence of box basis functions, each box begins of the same length resulting in a new time series of length $N < n$.

X: original time series of length n

It can be represented in the N-dimensional space.



MATRIKAS

Date No.

Example: Let X = [1 3 7 1 4 4 4 5 3 7]; X can be mapped from its original dimension n=9 to a lower dimension, eg N=3, as follows: [1 3 7 1 4 4 4 5 3 7]
[↓ 1 ↓ 4 ↓ 5]

What does it do (Paws):- It takes original time series & it breaks it down into segments non overlapping and it computes average of within its segment.

Pros:- *) Very fast to compute (*) As efficient as other approaches (*) Supports non-Euclidean measures

Cons:- It assumes as fixed length non-overlapping summary window over the time series.

*) The solution to PAA is APCA:- It does the opposite to PAA, If we have a new data & do simple PAA we will have a fixed weight for each segment that we have to define then we compute average/mean of the within segment

The problem:- We need two representa, length & value. We need to come up exactly with one algorithm that we want to use for segmentation and there are different time series segmentation algorithms.

Where should you do this segmentation where should you stop one segment & where one segment is begin?

So we have to define how much fine grained we want it to be, how slow

*) It is not commonly used because harder to implement

Pros:- Faster to calculate, More efficient than other approaches, Supports non Euclidean measures

Cons:- Performance depends upon segmentation technique used.

MATRIKAS

Symbolic Approximation (SAX):- Similar principle to PAA, start with original time series & use segments to represent it. But it represents segments with symbols (rather than real numbers) small memory footprint.

Definition of how it works:- We define these green lines (cates), we decide how many cates we want and then for each band the cates we assign a letter (discreate value).

Procedural:- *) First, we have to z-normalize & apply PAA to it by defining no. of segments. Next, discretize into symbols

*) How do we define the bands?

We are making an assumption that the time series we are having is a Guassian distribution. Then we can divide the Guassian distribution into areas of equal size.

*) How do we decide the alphabet? We will decide it, we will try different alphabet size to the task we are solving and see which alphabet size solves the problem & we want to map the time series to SAX before we apply classification then we try diff. size of alphabets which performs well.

Problems:- *) The symantic of those values

Time series classification:-

If we use 1-NN we can use any distance measures;

Pros:- accurate, Robust, Simple; Cons:- time and space complexity; results are not interpretable.

How about feature-based classification? Use shapelets as "attributes" or "features" for splitting a node in the decision tree.

Shapelets:- time series subsequence, maximally representative of a class, discriminative from other classes.

MATRIKAS

Date: No.

Shapelets are short time series.

*) If we have a shapelet how do you quantify it? So, any subsequence is a feature and what is the feature value? It can use many diff ways to represent. One way to take the shapelet is: use a sliding window and run DTW, Euclidean dist, less for each slide take the minimum one will be the best match.

*) If we have a set of shapelets we can build a decision tree (or) a shapelet tree. How does it work? It works same as the regular Decision tree. How can apply? So, we have the dictionary and then each node is decision node. could be binary. If the best match is higher than some value, if yes you go to one side of the tree. If not we go other side of the tree. And keep on testing every shapelet until we find the decision node.

How do we define shapelet?

*) We apply any transformation (SPK, PAA) & do KNN.

*) As soon as we find shapelet, select top k most informative shapelets as features and learn any suitable classifier (svm, Random forest) using the transformed dataset.

*) Take random no of shapelets & cluster them, take means of those clusters by using K-means & start petrub those K-shapelets to improve them & check their utility in terms of classification.

Rocket:- *) Keep similar (or) better classification results on huge time series datasets and its much faster than any other.

working:- It initializes a set of convolution kernels that are random. The convolution of each kernel with an

MATRIKAS

Input time series probabilities & feature vector. They don't define kernels, they don't train them. Next, they look at only two features they look at is the proportional positive value after convolution is applied & also maxpooling. And then they concatinate these two value. total no. of kernals & they feed it to the regressor.

*) Dilated convolution:- If we have 3x3 size kernel & then take image & can kernel across the image it we will get a number by doing dot product the normal convolution. So if we apply dilation what it does is it, instead of move with dilation of 2 that make our convolution area wider, actually add row (or) column after each and every point of the convolution kernel. It very useful cause manage with noise & time series of different scales that's why dilation is needed for time series

4.9 Interpretability & explainability:

4.10 RL :- where we learn through experience of trail & error method

*) The art of learning, so there is an intelligent agent which interacts with environment & it learns by trail & error & gradually learns how to take decisions.

Learn to

Goal:- Make optimal sequence of decisions in an unknown environment by Trial & Error.

*) What do the AI agent learn? Our goal is to maximize the total accumulated reward over a period of time by taking the sequence of the rewards we get.

*) Three Main Paradigms :- i) Multiarmed Bandits (ii) Contextual Bandits (iii) Markov Decision process

Armed
Multi Bandits:- No change in environment, the environment remains static, the state doesn't change we just take some actions & based on it we get some rewards & we learn which actions are good.
Contextual Bandits:- Environment has some context (or) state but there is no co-relation from one state to another state.
Markov Decision Process:- Has both states as well as dynamics b/w one state to another state. The next state is dependant on current state.
Main Characteristics of RL :- i) Optimization (ii) Exploration, Exploitation (iii) Delay Reward (iv) Generalization.
Optimum
How do we design rewards? We have to design in such a way that what the behaviour the agent is learning is what we want it to learn.
Exploration - Exploitation Trade off:- When an agent is interacting with the environment it has to decide whether i have to use my knowledge or should i explore more to get new knowledge.
Delayed Reward:- There can be two approaches while we are learning :- i) Greedy approach (ii) patient approach.
Greedy:- Just alway take the action which is giving you the best reward immediately.
Patient:- Taking an action that will lead to better accumulated reward on the long run, not immediately.
Generalization:- In which each pixel combination is a new state for the agent. We train a few states and we can generalize for a large set of states & this achieved using deep Reinforcement learning.

MATRIKAS

Where we train a weight vector of the neural network & the weight vector is sufficient to give result for any state

Multi Armed Bandits (setup):- do what we learn is how the exploration and exploitation trade off is done. we need to explore more to get more experience about the environment.

Action-values:- How to explore? and how to exploit?

~~we need some value which can help~~ The action value tells us, this action has this much value: If you take this action this is the average reward you get.

If you already know that for this action this is the reward I am going to get ~~from~~ that is optimal action.

* When we actually want to exploit (or) explore? When we want to exploit, whatever Q values which we have till now we will see which action gives us the best value and we will take that action. When we want to explore we will take a random action & then we will see what's the reward & update the Q value.

Bayes Algorithms which helps in exploration exploitation:-

*) (i) ~~Greedy~~ (ii) Upper Confidence bound algorithm.

ϵ-Greedy:- If we want to explore we do it with the probability ϵ (epsilon). We decide some probability at the beginning. if we want to explore we will take an random action and update the Q values based on the reward we get. When we want to exploit we will choose the max value what we received till now. Probability of exploitation is $1-\epsilon$

Problems:- Inefficient exploration:- ϵ is static. It explores with the same probability from starting to end

Inefficient exploitation:- Spend too much/little time on clearly bad/good actions

MATRIKAS

Upper Confidence Bounds :- i) we want to take an action only if the reward for the action is large enough (ii) when we are not confident about that action.

Principle's for UCB:- i) optimism in face of uncertainty
ii) statistical confidence Bounds

Date No.

i) optimism in face of uncertainty:- Acts as if the environment is as plausible as possible. If unsure, then overestimate. $\hat{Q}(a)$

ii) statistical Upper Confidence Bounds:- ~~we can use some statistical bounds~~ The average value of the random variable is bigger than the particular value these inequalities followed by that. ~~These upper confidence bounds help to capture the~~

* How it is applied? In each iteration we choose the action which maximize the UCB value & update Q value based on the reward we get.

* How it works? if you are sure the UCB value is high if you are not sure the UCB value is high.

* How good they are performing? This can be done by using regret analysis.

* Regret Analysis:- UCB is more efficient than ϵ-greedy as time increases in ϵ-greedy and lower time in UCB.

~~Contextual Bandits~~ Markov Decision Process:- we have a state and we have relationship b/w one state and other state

~~Process:-~~ * The environment is in one state and the agent takes action and get reward. As a result of this action the agent moves to next state, next action, next reward. This trajectory.

* Given current states and the history of actions taken till now, what is the probability moving to next state. To some dynamics involved in that.

* Markovian assumptions:- The current state is enough to give the ~~abbreviation~~ information of whole history of the past.

Deep RL: Generalization: The no of features increase the state space increases exponentially that's called curse of dimensionality

* We will learn the approx Q values and using the approx based RL algorithms they are proven to be more effective.

Date No.