Unit - IV

① Regularization for Deep Learning :-

→ Strategies or techniques which are used to reduce the error on the test set at an expense of increased training error are collectively known as Regularization.

→ It is used to prevent overfitting in neural networks & thus improve accuracy of a DL model when facing completely new data from the problem domain.

→ Regularization can be defined as any modification we make to a learning alg. that is intended to reduce its generalization error but not its training error.

Generalization error – how well an ML model predicts new data.

Training error – error/diff btw predicted o/p & actual o/p.

→ Developing more effective regularization strategies can make a trade profitable by reducing variance significantly while not overly increasing the bias.

→ Reduce Overfitting

- Help in Generalization (generalize well to new data by avoiding complex function that fit noise)

⑧ Parameter Norm Penalties:-

→ These are regularization techniques used to add a penalty term to the loss function, encouraging the neural network to learn smaller & simpler weights.

→ This helps in reducing overfitting & improving generalization, by discouraging complex models.

→ It helps control large weight values, making the model more stable & simpler.

→ Penalty Term:-
In the loss function, the penalty term discourages complex models by adding an extra cost for large weights.

→ It basically encourages models to learn simple patterns & prevent overfitting by keeping the weights smaller & improving generalization ability.

→ The strength of the ~~penalty~~ penalty is usually controlled by a hyperparameter (λ) which decides how much weight is given to the penalty term.

→ A small λ results in weak penalty (risk of overfitting) but a large λ results in a strong penalty (risk of underfitting)

→ Types :-

1) L1 Regularization :- (Lasso)

- Adds absolute sum of weights to loss function.

  Loss = Original Loss + $\lambda * \Sigma |w|$

- Effect :- Allows for automatic detection of most relevant features in possible way
  Encourages sparsity by forcing some weights to become exactly 0.
- Best used in feature selection models.

2) L2 Regularization (Ridge) :-

- Adds sum of squares of weights to loss function.

  Loss = Og loss + $\lambda * \Sigma w^2$

- Effect :-
  Shrinks large weights towards zero but never exactly zero.
- Used in smoothing the model, to make it more stable.

3) Elastic Net Regularization :

- Combination of L1 & L2 penalties.
- Loss = Og loss + $\lambda_1 * \Sigma |w| + \lambda_2 * \Sigma w^2$
- Effect :-
  Balances sparsity and smoothness by combining benefits of L1 & L2 regularization.

→ Disadv

- Choosing right λ value can be challenging
- Over-regularization may lead to underfitting.

(08) Regularization & Under-Constrained Learning:-

→ An under-constrained problem in deep learning occurs when there are more trainable parameters (weights) than the amount of training data.

→ This situation makes the model capable of fitting the training data perfectly, often learning noise instead of useful patterns.

→ As a result, the model overfits & performs poorly on unseen data.

→ Think of it like trying to fit 100s of puzzle pieces (parameters) into a picture when you only have a small part of the picture (data).

→ Model might memorize everything, including unnecessary details or noise, instead of learning useful patterns.

→ How Regularization helps?

Regularization is like adding rules or penalties to prevent the model from becoming too complicated. These rules forces the model to focus on important patterns & ignore useless details.

→ Regularization techniques

1) L1 Regularization (Lasso) :-
- Adds a penalty for large weights by shrinking some weights to zero.
- Think of it like keeping only most important connections in the NN
- Result :- Simpler model that ignores irrelevant features.

2) L2 Regularization (Ridge) :-
- Adds a penalty for large weights but doesn't shrink them to zero.
- Think of it like gently controlling the size of all connections.
- Result :- A smoother, more stable model.

3) Dropout :-
- Randomly turns off some neurons during training.
- Imagine solving a problem without using all your brain cells at once, this forces remaining neurons to work harder.

4) Early Stopping :-
- Stops training when the model stops improving on test data.
- Prevents from learning unnecessary noise.

5) Data Augmentation :-
- Creates extra data by slightly changing the original data

⑧ Norm Penalties as Constrained Optimization:-

→ In DL, optimization means finding the best weights for a neural network to reduce the loss function.

→ Constrained optimization means solving this problem while following certain rules or limits, such as keeping the weights small.

→ Working of Norm Penalties in Constrained Optimi-

1) Regular loss Minimization:

Normally, we minimize only the loss:

Minimize $L(w)$

where, $w \rightarrow$ weights of n/w

$L(w) \rightarrow$ loss function

2) With Constraints:

In constrained optimization, we also say:

- The total size of the weights must be below a certain limit.
- This "size" is measured using norms like L_1 or L_2 norm.

For ex:

$\|w\|_1 \leq c$ (L_1 constraint)

This prevents the network from having very large weights, which helps reduce overfitting.

→ Recasting Constraints as Penalties:

Instead of setting strict limits, we add a penalty to the loss, if the weights are too large:

$$L(w) + \lambda ||w||$$

where;

$L(w)$ → original loss function

$||w||$ → weight size (L1 or L2)

λ → controls how much importance given to penalty.

→ Types of Norm Penalties:-

1) L1 Regularization

- Adds the absolute value of weights.

$$L(w) + \lambda \Sigma |w_i|$$

- Effect: Makes some weights exactly zero, leading to simpler models.

2) L2 Regularization

- Adds the square of the weights.

$$L(w) + \lambda \Sigma w_i^2$$

- Effect:- Shrinks weights towards zero but never makes them exactly zero.

→ Uses:-

- Prevents Overfitting
- Improves Generalization
- Simpler Models.

→ Ex:-

Think of training a NN like driving a car on a winding road. Without constraints, you might drive dangerously fast. Constrained optimization is like putting a speed limit - it forces you to drive safely.

⑧ Data Augmentation:-

→ It is a technique used in ML & DL to artificially increase the size & diversity of a training dataset by applying transformations to existing data without changing their labels.

→ It is a way to create more training data for a model by slightly changing existing data.

→ This helps the model learn better & perform well on new, unseen data.

→ When you don't have enough data, it creates extra examples.

→ It help the model understand diff variations of data & help in reducing overfitting.

→ Common Data Augmentation Techniques for Images:

1) Geometric Transformations:
   - Rotation
   - Flipping
   - Scaling (Enlarging or shrinking)
   - Cropping

2) Color Space Transformations:
   - Brightness Adjustment
   - Contrast Adjustment
   - Color shifting

3) Noise Injection (Gaussian noise)
Adding random noise to simulate real-world imperfections.

4) Affine Transformation :-
- Transforming position of pixels while preserving straight lines.
- Includes translation, shearing (break off), & rotations.

5) Cutout :-
- Masking out random patches of image to n/w to focus on less obvious features.

→ DA for Text Data :-
1) Synonym Replacement
2) Random Insertion into sentences
3) Back translation (another lang. & back to og lang.)

→ For Time-Series Data :-
1) Time Warping (Stretching & compressing signal)
2) Jittering (add random noise)
3) Permutation (shuffle ~~segmentation~~ segments of data)

→ Tools used for DA :-
1) Keras & ImageDataGenerator
2) Albumentation (Img Augmentation library)
3) NLTK, Spacy (Text)

(v) Noise Robustness :-

→ Noise Robustness refers to the ability of a DL model to maintain its performance even in the presence of noisy or imperfect data.

→ Noise can come in various forms, such as random errors, mislabeled data, or corrupted data, and can negatively impact a model's ability to learn generalizable patterns.

→ Models that are not robust to noise may become unstable, leading to erratic behaviour or poor performance.

→ Common Sources of Noise in DL :-

1) Label Noise : Errors in labels of training data (Ex:- img labeled as "cat" when its a dog)

2) I/P Noise : Sensor errors or transmission noise (Ex:- pixel noise in images, bg noise in audio)

3) Model Overfitting :- Overfitting to noisy data patterns rather than generalizable patterns.

→ Techniques to Improve Noise Robustness :-

1) Data Augmentation (Simulate noisy cond & provides training samples)

2) Regularization Techniques (randomly dropping units)

3) Robust loss Functions :-

. Using loss functions that are less sensitive to outliers or noise.

Ex:- Huber loss rather than Standard squared Error.

4) Noise Injection:-

- Adding noise deliberately during training to make the model more resilient to it.
  Ex:- Gaussian noise

5) Denoising Autoencoders:-

- A type of neural network used to remove noise from data. The model is trained to reconstruct clean data from noisy i/ps.

6) Ensemble learning:-

- Combining multiple models to reduce the impact of noise on any single model.
- Bagging & Boosting: These techniques use multiple models to improve the final prediction & reduce variance by noise.

7) Adversarial Training:-

- Involves training the model on adversarial examples, designed to trick the model to make it more robust to noise.

8) Early Stopping:-

- By monitoring validation performance, you can stop training when the model starts to overfit, reducing the chances of fitting to noisy data.

→ Challenges:-

1) Identifying True Patterns vs. Noise
2) Trade-off b/w Noise & Model complexity [too much leads to underfitting]

(16) Semi-Supervised Learning:-

→ It is a type of Machine learning that combines both labeled & unlabeled data for training.

→ It lies b/w supervised & unsupervised learning.

→ It uses a small amount of labeled data & a large amount of unlabeled data to improve learning accuracy & generalization.

→ Collecting labeled data is costly & time-consuming so it can be used as it also uses a large set of unlabeled data.

→ By using both methods, it provides better results and has improved accuracy.

→ Working:-

1) It uses label propogation, which helps the labels propogate from labeled data to unlabeled based on similarities & similar data points. This helps it for the models to capture the structure of data and make accurate predictions.

→ Techniques Used:-

i) Self-Training:-

- Model is first trained on small labeled data sets.
- It then predicts labels for unlabeled data, which are added to training sets.

- This process is repeated, gradually increasing no. of labeled samples.

2) Co-training:-

- Two models are trained on different views (subsets) of the data.
- Each model labels the unlabeled data for the other model, help each other improve.

3) Generative Models:-

- These models assume that the data is generated by a mixture of distributions, and use both labeled & unlabeled data to learn parameters of distributions.

4) Graph-Based Methods:-

- This methods models data as a graph, where labels are propogated across the graph to infer labels for unlabeled data.

5) Consistency Regularization:-

- The model is trained to o/p same prediction when small noises/disturbances are applied to i/p data to handle unlabled data in a more robust way.

→ Adv:-
Better Performance, Cost-Effective

→ Disadv:-
Quality of Unlabeled data, More complex

(4) Multi-task learning:-

→ MTL is a type of ML technique where a model is trained to perform multiple tasks simultaneously.

→ In DL, MTL refers to training a neural n/w to perform multiple tasks by sharing some of n/w's layers & parameters across tasks.

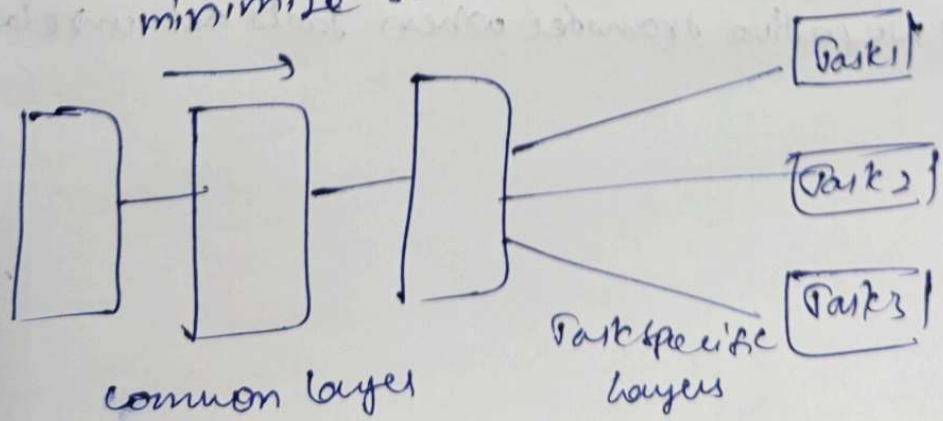
→ In MTL, the goal is to improve the generalization performance of the model by leveraging the information shared across tasks.

working :-

→ Shared layers :-
A common set of neural network layers is used to learn features shared among all tasks.

→ Task-specific layer :-
Each task has its own unique layer to capture task-specific details.

→ Loss Functions :-
The total loss is the combination of losses from each task, which the n/w tries to minimize simultaneously.



→ Let $L_1, L_2, \ldots, L_T$ be loss functions for T tasks

The total loss L_{Total} is

$$L_{Total} = \sum_{t=1}^{T} w_i L_i$$

where w_i → weights to balance importance of each task

Ex:- Consider a self-driving car with following tasks:

1) Detecting objects
2) Estimating distances to obstacles
3) Predicting the speed limit

All these can be learned using a single neural network in an MTL framework.

→ Apps:-
NLP, computer vision, speech Recognition

→ Adv:-
1) Better generalization due to shared learning
2) Improved performance for related tasks.
3) Efficient use of computational resources.

→ Disadv:-
1) complex model design
2) Negative transfer when tasks are unrelated.

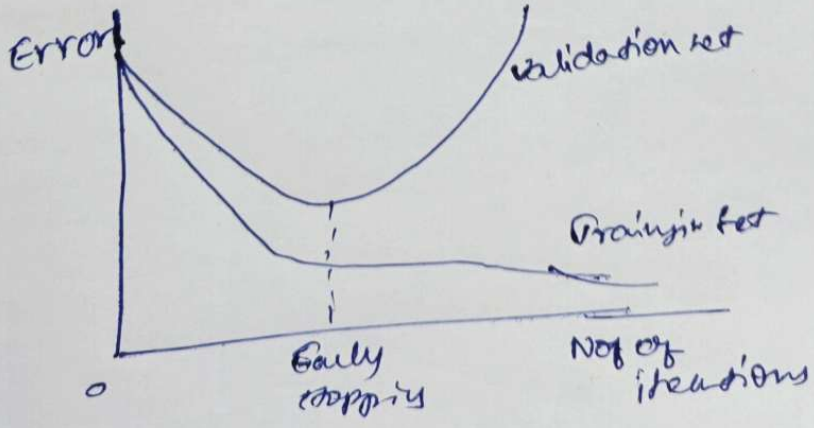
⑧ Early Stopping:-

→ Early Stopping is a regularization technique used in training deep learning models to prevent overfitting.

→ It stops the training process when the model's performance on validation dataset starts to degrade, indicating that the model has begun to memorize the training data instead of generalizing from it.

→ Working:-

1) Split the dataset into training & validation sets [evaluate model's performance].
2) Train the model & monitor its performance on the validation set after each epoch.
3) Record the best validation performance during training.
4) Stop training if the validation performance doesn't improve for a specified no. of epochs (patience).



→ Adv:-

- Reduces computation time & costs.
- Prevents overfitting.
- Simple & easy to implement

→ Disadv:-

- Requires careful tuning of patience value
- May stop training too early in some cases.

④ Parameter Tying & Parameter Sharing:-

Aspects	P. Tying	P. Sharing
Definition	same set of weights is used for diff parts of network	same weights are used across multiple layers or nodes
Purpose	To reduce the no. of parameters & enforce symmetry	To reduce complexity & improve generalization
Common Ex:	weights in CNN filters across diff image regions.	weights in CNN filters or RNN weights across time st
Memory Efficiency	Significantly reduces the no. of parameters.	Reduces no. of unique parameters stored.
Training Efficiency	Speeds up training due to fewer parameters.	Speeds up training due to fewer parameters to learn
Regularization Effect	Helps prevent overfitting by limiting model capacity.	Helps prevent overfitting by reducing the no. of parameters.
Constraint	Parameters are enforced to be identical across parts.	Parameters are shared but can be used at diff layers/times
Model complex	less complex by limiting no. of parameters	less complex by minimizing no. of parameters
Use	Primarily used in CNNs.	Primarily used in both CNNs & RNNs

④ Sparse Representations :-

→ These refer to the idea of encoding data in a way that most of the elements in the representation are zero or close to zero, while only a small number of elements are non-zero.

→ The goal is to represent the data with fewer active features, making the representation more efficient and easier to understand.

→ Sparsity :- It means that most values in dataset or vector are zero or near-zero.
Eg :- In a vector of size 1000, only a few values may be non-zero, making the representation sparse.

→ Sparse Coding :- It is an approach where data is represented in terms of few basis elements or features, which makes it easier to model & understand complex data.

→ Sparse Representations are used in various NN models, particularly in autoencoders & CNNs, to reduce computational costs & avoid overfitting.

→ In NN, sparsity can be enforced by adding regularization terms, which encourages the model to use a small no. of active features during training.

→ Adv:-

- Help reduce memory usage & computation time as majority elements are 0
- It is easily interpretable
- Reduces overfitting

→ Apps:-

Image compression, NLP, NN optimisation

⑫ Ensemble Methods:-

→ Ensemble Methods are techniques that combine multiple ML models to improve overall performance.

→ These methods are based on the idea that combining the predictions of several models will result in better generalization & reduced overfitting.

→ The idea is to leverage the "wisdom of the crowd" by combining predictions from several base models to make final prediction.

→ The ensemble method's goal is to reduce errors, variance, & bias, depending on the specific methods used.

→ Some key ensemble methods are:

1) Bagging (Bootstrap Aggregating):-

- It helps to reduce variance & prevent overfitting and is mainly used for high variance models like decision trees.
- Working
  a) Data sampling (Checks all data which can appear multiple times or not at all)
  b) Model Training
  c) Prediction Aggregation
     (For classification, prediction made by voting)
     (For regression, averaging votes)

2) Boosting:-

- It aims to reduce both bias & variance and works by sequentially training models, where each new model tries to correct the errors made by previous models.
- Working:-
  a) Sequential learning
  b) Weighted Voting (better models are given more influence in final decision)

3) Stacking:-

- Here, multiple diff types of models are trained, & a meta-model is used to combine their predictions.
- Working:-
  a) Multiple Models (instead of using same model multiple time, uses diff. models)
  b) Meta-Model

4) Voting:-

- Voting is simple ensemble method where predictions from multiple models are combined, and class with most votes is selected.

→ Bagging vs Boosting:-

Aspects	Bagging	Boosting
Concept	Trains models in parallel on diff data subsets	Trains models sequentially, focusing on errors from previous models
Data Sampling	Uses random sampling with replacement.	Models are trained on entire dataset
Training	Trained independently in parallel	Models are trained sequentially
Effect on bias & variance	Reduces variance	Reduces both variance & bias
Risk of overfitting	Less prone	More prone
Parallelism	Easily	Can't be parallelized
Final Prediction	Majority voting (Classification) Averaging (Regression)	Weighted voting
Ex:-	Random Forest, Bagged Decision Trees	AdaBoost, Gradient Boosting, XGBoost

④ DropOut:-

→ It is a regularization technique used in NNs to prevent overfitting during training.

→ Overfitting occurs when a model learns not only general patterns but also noise in data leading to poor performance on new data.

→ In dropout, randomly selected neurons are dropped out (set to zero) during each training iteration.

→ This means that during each train step, a diff subset of neurons is used, preventing the nlw from becoming too reliant.

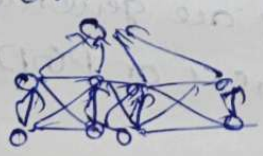
→ working

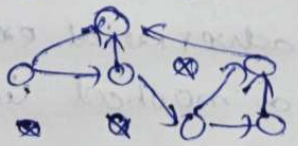
- During training, for each forward pass, dropout randomly deactivates a certain percentage (20% to 50% typically) of neurons.
- These neurons that are dropped out, don't contribute to the forward pass or back propogation, meaning their weights are not updated during that iteration.
- At test time, no neurons are dropped, but the weights are scaled by the dropout rate to account for the neurons that were removed during training.

→ For example, if 50% of neurons are dropped during training, the weights of neurons used during testing are multiplied by 0.5.

→ Adv:-

Prevents Overfitting, Improves Performance on unseen data, provides efficient regularization.





⑧ Adversarial Training:-

→ It is a technique used in ML & DL to improved the robustness of a model by training it on adversarial examples.

→ These are i/ps that are intentionally modified to deceive the model into making incorrect predictions or classifications.

→ The aim of adversarial training is to make the model more resistant to such adversarial attacks; improving its generalization & robustness in real world scenarios.

→ Adversarial examples are i/ps to a model that are slightly altered in a way that they cause model to make incorrect predictions.

common techniques to generate these example include:
Fast Gradient Sign Method (FGSM).
Projected Gradient Descent (PGD)

→ Adversarial Attacks are those attacks where the attacker makes small, carefully chosen changes to the i/p data to mislead the model.

→ Working:-

1) Generate Adversarial Examples:-
. First, adversarial examples are generated using a method like FGSM or PGD.

2) Include Adversarial Examples in Training:-
The generated examples are added to the training set, and the model is trained on this set.

3) Model Optimization:-
The model learns to correctly classify both adversarial & clean examples. This process also involves min a loss function.

4) Evaluate Model Performance:-
After training, the model's performance is evaluate on a test set containing both clean and adversarial data. The goal is to ensure that model can classify both types correctly.

→ Challenges:-
- Computational cost is high
- Risk of overfitting

⑫ Tangent Distance:-
→ It is a concept commonly used in the context of Neural Networks, especially in tasks like pattern recognition & image classification.
→ It is typically used to compare or measure the difference b/w two vectors or data points in a feature space.
→ It accounts for the geometric structure of the data, particularly for the data that has nonlinear relationships.

→ It will be useful when we want to measure the dissimilarity between two datapoints, but with more emphasis on their directional changes rather than just the straight line distance.

→ The tangent distance b/w 2 vectors X_1 & X_2 can be:

$$d_{tan}(X_1, X_2) = |\|X_1\| - \|X_2\||$$

→ This distance metric considers the angular difference b/w the vectors rather than just Euclidian distance, which may be misleading for data that is not linearly seperable.

→ This distance improves classification of non linear data like image recognition tasks. (Cims, speech)

→ It also helps with Dimensionality Reduction.

→ This distance allows the model to capture more complex patterns in data, leading to better performance than traditional distance measures.

④ Tangent Prop (Propogation):-

→ It is a method used in NNs, particularly in training deep nws, to deal with the vanishing gradient problem.

→ It was introduced to improve training by modifying the way gradients are propogated through the nw, especially deep architectures.

→ The vanishing gradient problem is that, when training a neural n/w with many layers, the signals used to update weights (gradients) in earlier layers become extremely small or vanish as they are passed backward in backpropagation hindering the learning process.

→ Transet Prop modifies this by using tangent vector to propogate the gradients.

→ Instead of solely relying on the backpropogation, it introduces another mechanism that helps to keep the gradients from diminishing.

→ Working:-
It focuses on adjusting the tangent direction of the activation function. This method computes the tangent of activation function, which helps preserve magnitude of gradient, ensuring learning rate does not shrink so much.

⑩ Manifold:-

→ A manifold in ML is a mathematical concept that refers to a space where each small piece of the space resembles a Euclidean space (flat space), but the global structure may be curved or complex.

→ Many real-world data, like images, audio, or even text, lie on a manifold within a higher-dimensional space.

→ This means that even though the data may appear to be in a high dimensional space, it can actually be represented in a much lower-dimensional subspace (manifold) that captures essential structure of data.

→ In ML, Manifold learning is a technique used to learn the underlying structure of the data. The goal is to map high dimensional data onto a lower-dimensional manifold that preserves imp properties of the data.

→ Deep neural networks often try to learn representations that capture the underlying manifold of data.

→ The idea is that, by learning good representation, the nw can make better predictions or classifications, and ideally align the representation with manifold structure of data.

→ Ex:
- Earth is 3D
- But when you stand on ground & see it feels like walking on a flat 2D surface (manifold)
- Similarly, each picture has thousands of pixel values (making it higher-dimensional)
- Shapes of pixels (0, 1) follow simple patterns, represented as lower dimensions (manifold)

(v) Tangent Classifier:-

→ It is an ML algorithm that works based on the idea of using tangent space to classify data.

→ It is particularly known for its nonlinear classification problems.

→ Imagine your data points are not in a straight line but scattered across a curved surface or space (manifold)

→ The key idea behind Tangent Classifier is that it learns the linear approximation of the data in a tangent space at a given point, which is then used for classification.

→ The tangent space provides a linear approximation to that manifold at each data point.

→ Linear approximation is a method used to estimate the value of a function near a given point using a straight line. If you zoom in close enough to a curve, it starts to look like a straight line which can be used to approximate the curves.

→ Working:-

1) Learn the manifold:-

First the alg learns a representation of the manifold from training data, using Principal component analysis (PCA)

2) Local linear Approximation:-

At each data point, the algorithm approximates local structure of data using linear model (like SVM).

2) Classification:-

After learning the manifold and local linear models, the classifier can predict the class of new, unseen data by mapping it to appropriate tangent space.

→ Adv:-

- Can handle nonlinear boundaries
- Flexible with complex data

→ App:-

- Image Recognition, Speech Recognition, Biomedical data Analysis

→ Disadv:

- Computational complexity
- Requirement of Proper Manifold Learning.

→ Ex:-

- Take earth as a manifold:- where it is curved and data points lie.
- Country borders as Decision Boundaries:-

  Borders seperate countries (classes) on Earth, like seperating data classes.

- Tangent Space:- Imagine standing at a point on Earth. The tangent plane is like a flat area around you, helping to understand local structure.
- Tangent classifier:- It uses the local flat area (tangent plane) and traverse through the land (tangent space) to figure out the border.
- Nonlinear Boundaries:- Even if borders are curved, the tangent plane helps classify data by using local-straight line approximations.