SEMESTER IV - UNIT 4A

Supervised Machine Learning:

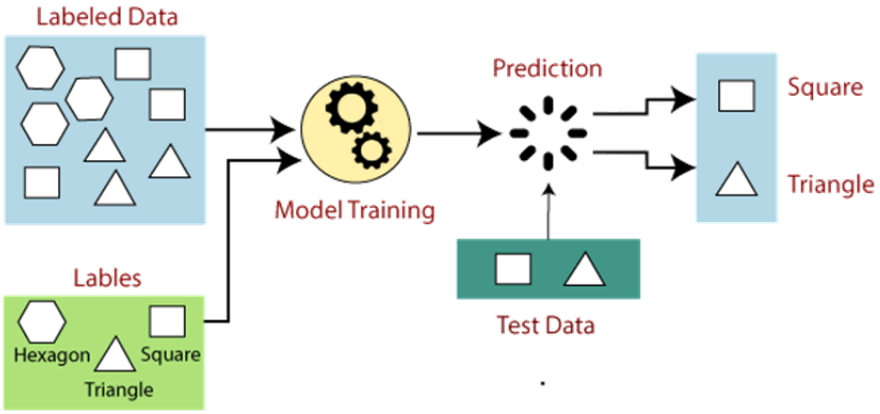
Supervised Learning হলো মেশিন লার্নিং-এর সেই ধরণ, যেখানে মেশিনগুলোকে ভালোভাবে লেবেলযুক্ত (labeled) প্রশিক্ষণ ডেটা ব্যবহার করে প্রশিক্ষণ দেওয়া হয়, এবং সেই ডেটার ভিত্তিতে মেশিনগুলো আউটপুট পূর্বানুমান করে। লেবেলযুক্ত ডেটা মানে হলো কিছু ইনপুট ডেটার সাথে ইতিমধ্যেই সঠিক আউটপুট ট্যাগ করা থাকে। Supervised Learning-এ, মেশিনগুলোকে দেওয়া প্রশিক্ষণ ডেটা এক ধরনের তত্ত্বাবধায়ক (supervisor) হিসেবে কাজ করে, যা মেশিনকে সঠিকভাবে আউটপুট পূর্বানুমান করতে শেখায়। এটি ঠিক সেই ধারণার মতো, যেমন একজন শিক্ষার্থী শিক্ষকের তত্ত্বাবধানে শেখে।

Supervised Learning হলো একটি প্রক্রিয়া, যেখানে মেশিন লার্নিং মডেলকে ইনপুট ডেটা এবং সঠিক আউটপুট ডেটা উভয়ই প্রদান করা হয়। Supervised Learning অ্যালগরিদমের মূল উদ্দেশ্য হলো একটি mapping function খুঁজে বের করা, যা ইনপুট ভেরিয়েবল (x) কে আউটপুট ভেরিয়েবল (y)-র সাথে সম্পর্কিত করে।

বাস্তব জীবনে, Supervised Learning ব্যবহার করা যায় যেমন: Risk Assessment, Image Classification, Fraud Detection, Spam Filtering ইত্যাদিতে।

Supervised Learning কিভাবে কাজ করে?

Supervised Learning-এ, মডেলগুলোকে লেবেলযুক্ত ডেটাসেট ব্যবহার করে প্রশিক্ষণ দেওয়া হয়, যেখানে মডেল প্রতিটি ধরনের ডেটা সম্পর্কে শেখে। একবার প্রশিক্ষণ প্রক্রিয়া সম্পন্ন হলে, মডেলটি পরীক্ষামূলক ডেটা (test data) এর ভিত্তিতে পরীক্ষা করা হয় (যা সাধারণত প্রশিক্ষণ সেটের একটি উপসেট), এবং এরপর এটি আউটপুট পূর্বানুমান করে। Supervised Learning-এর কাজ বোঝা সহজ হয় নিচের উদাহরণ এবং চিত্রের মাধ্যমে:



ধরে নাও আমাদের কাছে বিভিন্ন ধরনের আকৃতির একটি ডেটাসেট আছে, যার মধ্যে রয়েছে বর্গক্ষেত্র (Square), আয়তক্ষেত্র (Rectangle), ত্রিভুজ (Triangle), এবং বহুভুজ (Polygon)। এখন প্রথম ধাপ হলো প্রতিটি আকারের জন্য মডেলকে প্রশিক্ষণ দেওয়া।

- যদি প্রদত্ত আকারের চারটি বাহু থাকে এবং সব বাহু সমান হয়, তবে এটি বর্গক্ষেত্র (Square) হিসেবে লেবেল করা হবে।
- যদি প্রদত্ত আকারের তিনটি বাহু থাকে, তবে এটি ত্রিভুজ (Triangle) হিসেবে লেবেল করা হবে।
- যদি প্রদত্ত আকারের ছয়টি সমান বাহু থাকে, তবে এটি ষড়ভুজ (Hexagon) হিসেবে লেবেল করা হবে।

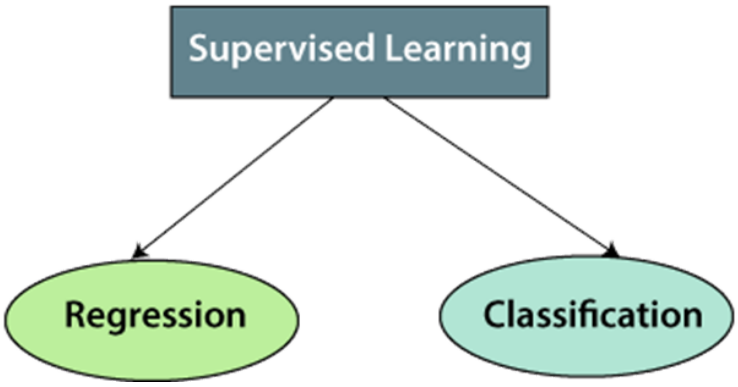
প্রশিক্ষণের পরে, আমরা টেস্ট সেট ব্যবহার করে মডেলটি পরীক্ষা করি, এবং মডেলের কাজ হলো আকৃতিটি সনাক্ত করা। মেশিনটি ইতিমধ্যেই সব ধরনের আকারে প্রশিক্ষিত, এবং যখন এটি একটি নতুন আকার খুঁজে পায়, তখন এটি বাহুগুলোর সংখ্যার ভিত্তিতে আকারটি শ্রেণিবিন্যস্ত (classify) করে এবং আউটপুট পূর্বানুমান (predict) করে।

Supervised Learning-এর ধাপসমূহ**:**

1. প্রথমে প্রশিক্ষণ ডেটাসেটের ধরন নির্ধারণ করো।
2. লেবেলযুক্ত প্রশিক্ষণ ডেটা সংগ্রহ/একত্র করো।
3. প্রশিক্ষণ ডেটাসেটকে ভাগ করো: training dataset, test dataset, এবং validation dataset।
4. প্রশিক্ষণ ডেটাসেটের ইনপুট ফিচারগুলো নির্ধারণ করো, যা যথেষ্ট তথ্য সরবরাহ করবে যাতে মডেল সঠিকভাবে আউটপুট পূর্বানুমান করতে পারে।
5. মডেলের জন্য উপযুক্ত অ্যালগরিদম নির্ধারণ করো, যেমন Support Vector Machine, Decision Tree ইত্যাদি।
6. প্রশিক্ষণ ডেটাসেটে অ্যালগরিদমটি প্রয়োগ করো। কখনও কখনও validation set ব্যবহার করা হয় নিয়ন্ত্রণ প্যারামিটার হিসেবে, যা training dataset-এর একটি উপসেট।
7. টেস্ট সেট প্রদান করে মডেলের সঠিকতা (accuracy) মূল্যায়ন করো। যদি মডেল সঠিক আউটপুট পূর্বানুমান করে, তবে এর মানে মডেলটি সঠিকভাবে কাজ করছে।

Supervised Machine Learning অ্যালগরিদমের ধরন**:**

Supervised Learning দুই ধরনের সমস্যায় বিভক্ত করা যায়:



১. **Regression**

Regression অ্যালগরিদম ব্যবহার করা হয় যখন ইনপুট ভেরিয়েবল এবং আউটপুট ভেরিয়েবলের মধ্যে সম্পর্ক থাকে। এটি ধারাবাহিক ভেরিয়েবল পূর্বানুমানের জন্য ব্যবহৃত হয়, যেমন আবহাওয়া পূর্বাভাস, মার্কেট ট্রেন্ড ইত্যাদি। কিছু জনপ্রিয় Regression অ্যালগরিদমের উদাহরণ:

- Linear Regression
- Regression Trees
- Non-Linear Regression
- Bayesian Linear Regression
- Polynomial Regression

২. **Classification**

Classification অ্যালগরিদম ব্যবহার করা হয় যখন আউটপুট ভেরিয়েবল categorical হয়, অর্থাৎ দুই বা ততোধিক শ্রেণিতে ভাগ করা যায়, যেমন হ্যাঁ-না, পুরুষ-মহিলা, সত্য-মিথ্যা ইত্যাদি। কিছু উদাহরণ:

- Spam Filtering
- Random Forest
- Decision Trees
- Logistic Regression

- Support Vector Machines

Supervised Learning-এর সুবিধা**:**

- Supervised Learning-এর সাহায্যে মডেল পূর্ব অভিজ্ঞতার ভিত্তিতে আউটপুট পূর্বানুমান করতে পারে।
- এতে বস্তু বা ক্লাসের সঠিক ধারণা পাওয়া যায়।
- Supervised Learning মডেল বিভিন্ন বাস্তব সমস্যা সমাধানে সাহায্য করে, যেমন Fraud Detection, Spam Filtering ইত্যাদি।

Supervised Learning-এর অসুবিধা**:**

- Supervised Learning মডেল জটিল কাজ পরিচালনার জন্য উপযুক্ত নয়।
- যদি টেস্ট ডেটা প্রশিক্ষণ ডেটা থেকে আলাদা হয়, মডেল সঠিক আউটপুট পূর্বানুমান করতে পারে না।
- প্রশিক্ষণ প্রক্রিয়ায় বেশ সময় ও কম্পিউটেশন লাগে।
- Supervised Learning-এর জন্য বস্তু বা ক্লাসের যথেষ্ট জ্ঞান থাকা আবশ্যক।

Bayesian Learning

Bayesian Learning-এর বৈশিষ্ট্যসমূহ:

- প্রতিটি পর্যবেক্ষণকৃত প্রশিক্ষণ উদাহরণ হাইপোথিসিসের সম্ভাব্যতা বাড়ায় বা কমায়।
- এটি এমন অ্যালগরিদমের চেয়ে বেশি নমনীয়, যা কোনো উদাহরণের সাথে অমিল পেলে হাইপোথিসিস বাতিল করে।
- Prior knowledge এবং পর্যবেক্ষণকৃত ডেটা একত্রিত করে হাইপোথিসিসের চূড়ান্ত সম্ভাব্যতা নির্ধারণ করা যায়।
- Bayesian Learning-এ Prior Knowledge দেওয়া হয়:
    - প্রতিটি সম্ভাব্য হাইপোথিসিসের জন্য prior probability নির্ধারণ করে।
    - প্রতিটি হাইপোথিসিসের জন্য পর্যবেক্ষণকৃত ডেটার probability distribution নির্ধারণ করা হয়।
- Bayesian পদ্ধতি probabilistic prediction করতে পারে।

- নতুন ইনস্ট্যান্সগুলিকে একাধিক হাইপোথিসিসের পূর্বাভাসের সাথে সম্ভাব্যতার ওজন ব্যবহার করে শ্রেণিবিন্যস্ত করা যায়।

- যদিও কিছু ক্ষেত্রে Bayesian পদ্ধতি computation-এর দিক থেকে জটিল, এটি optimal decision-making-এর মান প্রদান করে।

Bayesian পদ্ধতির সমস্যাসমূহ**:**

- অনেক প্রাথমিক সম্ভাব্যতা জানা প্রয়োজন।

- যখন এগুলো জানা থাকে না, তখন পূর্ব অভিজ্ঞতা, পূর্বের ডেটা এবং assumptions ব্যবহার করে অনুমান করা হয়।

- সাধারণ ক্ষেত্রে Bayes optimal hypothesis নির্ধারণে উচ্চ computational cost প্রয়োজন হয়। বিশেষ পরিস্থিতিতে এটি কমানো যায়।

Bayes Theorem

- মেশিন লার্নিং-এ আমরা observed training data D এর ভিত্তিতে hypothesis space H থেকে সেরা হাইপোথিসিস নির্ধারণ করতে চাই।

- Bayesian Learning-এ সেরা hypothesis মানে হলো সবচেয়ে সম্ভাব্য hypothesis, প্রদত্ত ডেটা D এবং prior knowledge অনুযায়ী।

- Bayes theorem ব্যবহার করে hypothesis-এর সম্ভাব্যতা নির্ধারণ করা যায়।

Bayes theorem-এর ধরণ**:**

- P(h): hypothesis h-এর prior probability
    - প্রশিক্ষণ ডেটা পর্যবেক্ষণ করার আগে hypothesis h কতটা সম্ভব তা নির্দেশ করে।
- P(D): training data D-এর prior probability
    - কোন হাইপোথিসিসের উপরে নির্ভর না করে D-এর সম্ভাব্যতা।
- P(h|D): D প্রদত্ত posterior probability of h
    - প্রশিক্ষণ ডেটা D দেখার পরে hypothesis h কতটা সম্ভব তা নির্দেশ করে।
- P(D|h): h প্রদত্ত D-এর সম্ভাব্যতা

- কোনো হাইপোথিসিস h সত্য হলে D-এর সম্ভাব্যতা।

Bayes Theorem সূত্র: $P(h|D)=P(D|h) \cdot P(h) / P(D)$

P(h|D) P(h) এবং P(D|h)-এর সাথে বৃদ্ধি পায়।

- P(h|D) P(D) বৃদ্ধি পাওয়ায় কমে যায়, কারণ D স্বাধীনভাবে ঘটার সম্ভাবনা বেশি হলে, D-এর মাধ্যমে h সমর্থনের প্রমাণ কমে।

Naïve Bayes Classifier Algorithm

- Naïve Bayes হলো Supervised Learning Algorithm, যা Bayes theorem-এর উপর ভিত্তি করে Classification সমস্যা সমাধান করে।
- প্রধানত text classification-এ ব্যবহৃত হয়, যেখানে high-dimensional training dataset থাকে।
- এটি simple এবং দ্রুত Classifier, যা দ্রুত পূর্বাভাস করতে পারে।
- এটি একটি probabilistic classifier, অর্থাৎ কোনো বস্তু'র সম্ভাব্যতার ভিত্তিতে পূর্বানুমান করে।
- জনপ্রিয় উদাহরণ: Spam filtration, Sentiment Analysis, Article Classification।

কেন এটি **Naïve Bayes** বলা হয়**?**

- Naïve: কারণ এটি ধরে নেয় যে প্রতিটি feature স্বাধীনভাবে ঘটছে।
    - উদাহরণ: কোনো ফলের রঙ, আকার, স্বাদ দেখে এটি আপেল কিনা নির্ধারণ করা। প্রতিটি বৈশিষ্ট্য আলাদাভাবে ফল শনাক্ত করতে অবদান রাখে।
- Bayes: কারণ এটি Bayes theorem-এর উপর ভিত্তি করে কাজ করে।

Bayes' Theorem

- Bayes' theorem কে Bayes' Rule বা Bayes' Law নামেও বলা হয়। এটি ব্যবহার করা হয় কোনো hypothesis-এর সম্ভাব্যতা নির্ধারণের জন্য, যেখানে prior knowledge থাকে। এটি conditional probability-এর উপর ভিত্তি করে কাজ করে।

Bayes' theorem-এর সূত্র: $P(A|B)=\frac{P(B|A)P(A)}{P(B)}$, যেখানে:

- P(A|B): Posterior probability — কোন hypothesis A-র সম্ভাব্যতা, প্রদত্ত ঘটনা B-র ভিত্তিতে।
- P(B|A): Likelihood probability — প্রদত্ত প্রমাণ B ঘটার সম্ভাব্যতা, যদি hypothesis A সত্য হয়।
- P(A): Prior Probability — প্রমাণ দেখা আগে hypothesis A-এর সম্ভাব্যতা।
- P(B): Marginal Probability — প্রমাণ B-এর সম্ভাব্যতা।

Naïve Bayes Classifier-এর কার্যপ্রণালী

Naïve Bayes Classifier-এর কাজকে একটি উদাহরণের মাধ্যমে বোঝা যায়:

ধরে নাও আমাদের কাছে একটি Weather dataset আছে এবং সংশ্লিষ্ট target variable হলো “Play”। এই dataset ব্যবহার করে আমাদের নির্ধারণ করতে হবে, কোনো নির্দিষ্ট দিনে আবহাওয়ার উপর ভিত্তি করে খেলা করা উচিত কি না। এই সমস্যা সমাধানের ধাপগুলো হলো:

1. প্রদত্ত dataset-কে frequency tables-এ রূপান্তর করো।
2. প্রদত্ত feature-এর সম্ভাব্যতা বের করে Likelihood table তৈরি করো।
3. এরপর Bayes theorem ব্যবহার করে posterior probability হিসাব করো।

সমস্যা উদাহরণ**:** যদি আবহাওয়া sunny হয়, তাহলে খেলোয়াড় খেলবে কি না?
সমাধান**:** প্রথমে নিচের dataset-টি বিবেচনা করো:

	Outlook	**Play**
0	Rainy	Yes
1	Sunny	Yes

2	Overcast	Yes
3	Overcast	Yes
4	Sunny	No
5	Rainy	Yes
6	Sunny	Yes
7	Overcast	Yes
8	Rainy	No
9	Sunny	No
10	Sunny	Yes
11	Rainy	No

12	Overcast	Yes
13	Overcast	Yes

আবহাওয়া শর্তের জন্য ফ্রিকোয়েন্সি টেবিল:

Weather	**Yes**	**No**
Overcast	5	0
Rainy	2	2
Sunny	3	2
Total	10	5

আবহাওয়ার অবস্থার জন্য সম্ভাব্যতা সারণি:

Weather	**No**	**Yes**	
Overcast	**0**	**5**	**5/14= 0.35**

Rainy	**2**	**2**	**4/14=0.29**
Sunny	**2**	**3**	**5/14=0.35**
All	**4/14=0.29**	**10/14=0.71**	

Bayes' Theorem এর প্রয়োগ**:**

P(Yes|Sunny)= P(Sunny|Yes)*P(Yes) / P(Sunny)

P(Sunny|Yes)= 3/10= 0.3

P(Sunny)= 0.35

P(Yes)=0.71

So, P(Yes|Sunny) = 0.3*0.71/0.35= 0.60

P(No|Sunny)= P(Sunny|No)*P(No) / P(Sunny)

P(Sunny|NO)= 2/4=0.5

P(No)= 0.29

P(Sunny)= 0.35

So, P(No|Sunny)= 0.5*0.29/0.35 = **0.41**

তাহলে উপরের গণনা থেকে আমরা দেখতে পাচ্ছি যে
P(Yes|Sunny) > P(No|Sunny)
অতএব, রৌদ্রোজ্জ্বল দিনে (Sunny day) খেলোয়াড় খেলতে পারে।

Naïve Bayes Classifier-এর সুবিধাসমূহ **(Advantages):**

○ Naïve Bayes একটি দ্রুত এবং সহজ মেশিন লার্নিং অ্যালগরিদম যা ডেটাসেটের শ্রেণি (class) পূর্বাভাস দিতে ব্যবহৃত হয়।
○ এটি Binary এবং Multi-class Classification উভয়ের জন্যই ব্যবহার করা যায়।

○ এটি অন্যান্য অ্যালগরিদমের তুলনায় Multi-class prediction-এ ভালো কাজ করে।
○ এটি টেক্সট ক্লাসিফিকেশন (Text Classification) সমস্যায় সবচেয়ে জনপ্রিয়, যেমন ইমেইল স্প্যাম ফিল্টারিং।

Naïve Bayes Classifier-এর অসুবিধাসমূহ (Disadvantages):

○ Naïve Bayes ধরে নেয় যে সব ফিচার একে অপরের থেকে স্বতন্ত্র (independent), তাই এটি ফিচারগুলির মধ্যে সম্পর্ক (relationship) শিখতে পারে না।

Naïve Bayes Classifier-এর প্রয়োগক্ষেত্র (Applications):

○ এটি Credit Scoring-এ ব্যবহৃত হয়।
○ এটি Medical Data Classification-এ ব্যবহৃত হয়।
○ এটি Real-time Prediction-এ ব্যবহৃত হতে পারে, কারণ Naïve Bayes একটি eager learner।
○ এটি Text Classification যেমন Spam Filtering এবং Sentiment Analysis-এ ব্যবহৃত হয়।

Machine Learning-এ Decision Tree:

Decision Tree হলো একটি বহুমুখী এবং সহজবোধ্য অ্যালগরিদম, যা ভবিষ্যদ্বাণীমূলক মডেলিং (predictive modeling)-এর জন্য ব্যবহৃত হয়। এটি ইনপুট ডেটার উপর ভিত্তি করে সিদ্ধান্ত গ্রহণের কাঠামো তৈরি করে, ফলে এটি Classification এবং Regression — উভয় কাজেই প্রযোজ্য।

এই অধ্যায়ে Decision Tree-এর উপাদান, পরিভাষা, গঠন প্রক্রিয়া, সুবিধা এবং এর প্রয়োগ ও শেখার অ্যালগরিদম নিয়ে আলোচনা করা হয়েছে।

Decision Tree কীভাবে কাজ করে**:**

Decision Tree একটি Supervised Learning Algorithm, যা ইনপুট ডেটার উপর ভিত্তি করে ফলাফল পূর্বাভাস দিতে ব্যবহৃত হয়। এটি একটি গাছের মতো কাঠামো (tree-like structure), যেখানে—

- প্রতিটি Internal Node একটি বৈশিষ্ট্য (attribute)-এর উপর ভিত্তি করে সিদ্ধান্ত নেয়,
- প্রতিটি Branch কোনো নির্দিষ্ট attribute-এর মান প্রকাশ করে, এবং
- প্রতিটি Leaf Node চূড়ান্ত সিদ্ধান্ত বা পূর্বাভাস (prediction) উপস্থাপন করে।

Decision Tree অ্যালগরিদম Supervised Learning বিভাগের অন্তর্ভুক্ত এবং এটি Classification ও Regression উভয় সমস্যার সমাধানে ব্যবহার করা যায়।

Decision Tree-এর পরিভাষাসমূহ **(Terminologies):**

● Root Node: গাছের সর্বোচ্চ নোড, যা প্রাথমিক সিদ্ধান্ত বা ফিচারকে প্রকাশ করে, যেখান থেকে গাছের শাখাগুলি শুরু হয়।

● Internal Nodes (Decision Nodes): সেই নোডগুলো যেগুলি কোনো বৈশিষ্ট্যের মান অনুযায়ী সিদ্ধান্ত গ্রহণ করে এবং অন্য নোডের সাথে সংযুক্ত থাকে।

● Leaf Nodes (Terminal Nodes): শেষ প্রান্তের নোড, যেখানে সিদ্ধান্ত বা পূর্বাভাস নির্ধারিত হয়।

● Branches (Edges): নোডগুলির মধ্যে সংযোগ, যা দেখায় সিদ্ধান্তগুলো কীভাবে নেওয়া হয়েছে।

● Splitting: কোনো নোডকে দুটি বা ততোধিক উপ-নোডে ভাগ করার প্রক্রিয়া, নির্দিষ্ট বৈশিষ্ট্য ও সীমা (threshold)-এর উপর ভিত্তি করে।

● Parent Node: যে নোড থেকে বিভাজন (split) শুরু হয়, সেটিই প্যারেন্ট নোড।

● Child Node: প্যারেন্ট নোড থেকে বিভাজনের ফলে তৈরি হওয়া নতুন নোড।

● Decision Criterion: এমন নিয়ম বা শর্ত, যা নির্ধারণ করে কীভাবে ডেটা ভাগ হবে (যেমন: Information Gain বা Gini Index অনুযায়ী)।

● Pruning: Decision Tree থেকে অপ্রয়োজনীয় শাখা বা নোড অপসারণের প্রক্রিয়া, যাতে ওভারফিটিং (overfitting) রোধ হয় এবং সাধারণীকরণ (generalization) উন্নত হয়।

Decision Tree কীভাবে গঠিত হয় **(Formation Process):**

Decision Tree তৈরির প্রক্রিয়ায় ডেটাকে ধাপে ধাপে বিভিন্ন বৈশিষ্ট্যের মান অনুযায়ী ভাগ করা হয়।
প্রতিটি Internal Node-এ এমন একটি বৈশিষ্ট্য নির্বাচন করা হয়, যা Information Gain বা Gini Impurity এর মতো মানদণ্ড অনুযায়ী সর্বোত্তম বিভাজন প্রদান করে।

এই Splitting Process চলতে থাকে যতক্ষণ না কোনো Stopping Criterion পূর্ণ হয়, যেমন— সর্বোচ্চ গভীরতা (maximum depth) পৌঁছে যাওয়া, অথবা কোনো Leaf Node-এ ন্যূনতম সংখ্যক উদাহরণ (minimum number of instances) থাকা।

কেন **Decision Tree** ব্যবহার করা হয়**?**

মেশিন লার্নিং-এ Decision Tree ব্যাপকভাবে ব্যবহৃত হয় নিম্নলিখিত কারণগুলোর জন্যঃ

● Decision Tree অত্যন্ত বহুমুখী (versatile) — কারণ এটি জটিল সিদ্ধান্ত গ্রহণের প্রক্রিয়াকে সহজভাবে ব্যাখ্যা করতে পারে এবং সহজে ব্যাখ্যা করা যায় (interpretable)।

● এর hierarchical (ক্রমবিন্যাসযুক্ত) কাঠামো বিভিন্ন কারণ ও ফলাফলের মধ্যে সম্পর্ককে তুলে ধরে জটিল সিদ্ধান্তের পরিস্থিতি সহজভাবে উপস্থাপন করতে সাহায্য করে।

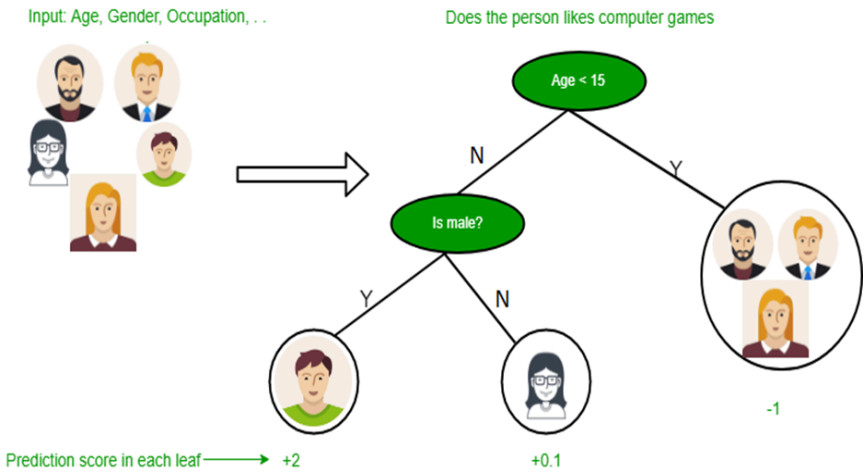
● Decision logic বা সিদ্ধান্ত গ্রহণের যুক্তি সম্পর্কে সহজবোধ্য ধারণা প্রদান করে বলে Decision Tree classification এবং regression কাজের ক্ষেত্রে বিশেষভাবে উপকারী।

● এটি সংখ্যাগত (numerical) এবং বিষয়ভিত্তিক (categorical) — উভয় ধরণের ডেটা নিয়েই দক্ষভাবে কাজ করতে পারে এবং স্বয়ংক্রিয়ভাবে উপযুক্ত ফিচার বেছে নেওয়ার ক্ষমতার কারণে বিভিন্ন ডেটাসেটে সহজে মানিয়ে নিতে পারে।

● Decision Tree সহজে ভিজুয়ালাইজেশন (visualization) করার সুযোগ দেয়, যার ফলে মডেলের অভ্যন্তরীণ সিদ্ধান্ত গ্রহণ প্রক্রিয়া বোঝা এবং ব্যাখ্যা করা সহজ হয়।

Decision Tree-এর পদ্ধতি **(Decision Tree Approach)**

Decision Tree সমস্যার সমাধানের জন্য একটি গাছের কাঠামো (tree representation) ব্যবহার করে, যেখানে— প্রতিটি leaf node একটি class label বা শ্রেণিকে উপস্থাপন করে এবং প্রতিটি internal node-এ ডেটার বৈশিষ্ট্যগুলো (attributes) প্রদর্শিত হয়।
Decision Tree-এর সাহায্যে আমরা যেকোনো Boolean function (যার মান True বা False হতে পারে) discrete attributes-এর উপর ভিত্তি করে উপস্থাপন করতে পারি।



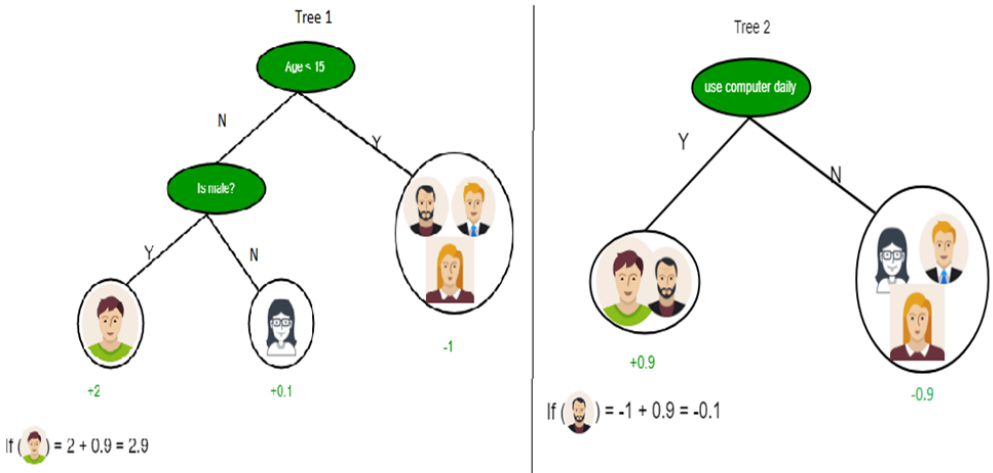
নীচে Decision Tree ব্যবহারের সময় আমরা যে কয়েকটি ধারণা (assumptions) গ্রহণ করি তা উল্লেখ করা হলোঃ

● প্রাথমিকভাবে আমরা পুরো training set-টিকে root node হিসেবে বিবেচনা করি।

● ফিচারের মানগুলো categorical (বিষয়ভিত্তিক) হওয়াই অধিক পছন্দনীয়। যদি ফিচারের মান continuous (অবিচ্ছিন্ন) হয়, তবে মডেল তৈরির আগে সেগুলোকে discretize (বিভাগীকরণ) করা হয়।

● Attribute values (বৈশিষ্ট্যের মান)-এর ভিত্তিতে রেকর্ডগুলোকে **recursive (পুনরাবৃত্তিমূলক)**ভাবে ভাগ করা হয়।

● Root node বা internal node নির্ধারণের জন্য আমরা পরিসংখ্যানভিত্তিক পদ্ধতি (statistical methods) ব্যবহার করি, যাতে কোন বৈশিষ্ট্যটি সিদ্ধান্তের জন্য সবচেয়ে উপযুক্ত তা নির্ধারণ করা যায়।



যেমনটি তুমি উপরের ছবিতে দেখতে পাচ্ছো, Decision Tree কাজ করে Sum of Product form-এ, যাকে Disjunctive Normal Form (DNF) নামেও ডাকা হয়। উপরের চিত্রে আমরা মানুষের দৈনন্দিন জীবনে কম্পিউটারের ব্যবহারের পূর্বাভাস (prediction) করছি।

Decision Tree-তে প্রধান চ্যালেঞ্জগুলোর একটি হলো — প্রতিটি স্তরে (level) Root Node-এর জন্য সঠিক Attribute (বৈশিষ্ট্য) নির্ধারণ করা। এই প্রক্রিয়াকে বলা হয় Attribute Selection (বৈশিষ্ট্য নির্বাচন)। Decision Tree-তে Attribute Selection করার জন্য দুটি জনপ্রিয় মাপকাঠি ব্যবহৃত হয়ঃ

1. Information Gain (তথ্য লাভ)
2. Gini Index (গিনি সূচক)

1. Information Gain (তথ্য লাভ)

যখন আমরা Decision Tree-র কোনো Node ব্যবহার করে Training Data-কে ছোট ছোট উপসেটে (subsets) ভাগ করি, তখন সেই ডেটার Entropy (অর্থাৎ অনিশ্চয়তা) পরিবর্তিত হয়।
এই Entropy পরিবর্তনের পরিমাণই হলো Information Gain।

ধরা যাক,

- S হলো ইনস্ট্যান্সগুলির (instances) একটি সেট,
- A হলো একটি বৈশিষ্ট্য (attribute),
- Sv হলো S-এর একটি উপসেট (subset),

- এবং v হলো বৈশিষ্ট্য A-এর একটি নির্দিষ্ট মান।

এখানে Values(A) হলো A বৈশিষ্ট্যের সমস্ত সম্ভাব্য মানের সেট। তাহলে,

Information Gain(S, A) = Entropy(S) - Σ [(|Sv| / |S|) * Entropy(Sv)]

Entropy (এন্ট্রপি**)**

Entropy হলো কোনো এলোমেলো চলকের (random variable) অনিশ্চয়তার একটি পরিমাপ।
এটি কোনো ডেটাসেটের অবিশুদ্ধতা (impurity) বা অসংগতি (disorder) নির্দেশ করে।
Entropy যত বেশি হয়, সেই ডেটায় তথ্যের পরিমাণও (information content) তত বেশি থাকে। Decision Tree-তে Entropy-এর সূত্র হলোঃ

$$Entropy = \sum_{i=1}^{c} -P_i * log_2(P_{i)}$$

এখানে,

- C হলো শ্রেণির (class) সংখ্যা,
- এবং p_i হলো সেই সম্ভাবনা (probability) যে কোনো উদাহরণ class i-এর অন্তর্ভুক্ত।

এই সূত্রটি একটি ডেটাসেট বা Decision Tree-এর কোনো Node-এর মধ্যে থাকা অবিশুদ্ধতা (impurity) বা অসংগতি (disorder) পরিমাপ করে।

সূত্র বিশ্লেষণ **(Formula Breakdown):**

- H(S): সেট S-এর Entropy — এটি ডেটার অনিশ্চয়তা বা বিশৃঙ্খলার পরিমাণ নির্দেশ করে।
- ∑ (Sigma): যোগফল চিহ্ন — প্রতিটি শ্রেণির (class) জন্য প্রাপ্ত ফলাফলগুলোকে যোগ করতে হবে।
- C: ডেটাসেটে মোট আলাদা শ্রেণির সংখ্যা।
  উদাহরণস্বরূপ, যদি দুটি শ্রেণি থাকে “Yes” এবং “No”, তবে C=2।
- p_i: কোনো উদাহরণ class i-এর অন্তর্ভুক্ত হওয়ার সম্ভাবনা। এটি নির্ণয় করা হয় — **class i-**এর উদাহরণের সংখ্যা কে সেটের মোট উদাহরণের সংখ্যা দিয়ে ভাগ করে।

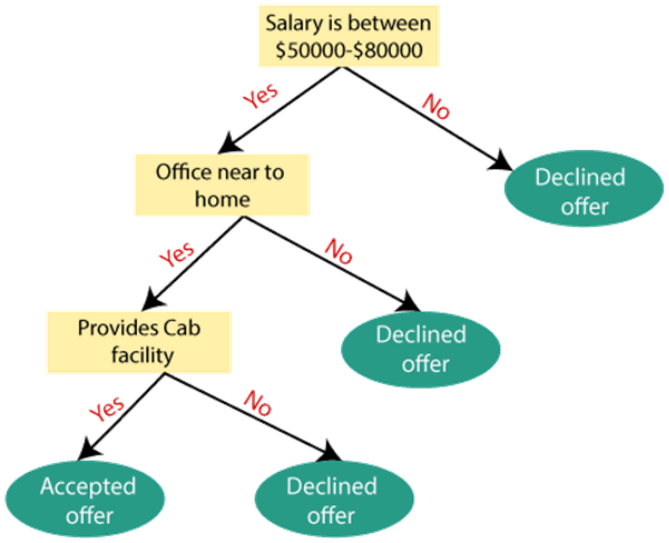
অর্থাৎ, Entropy যত বেশি হবে, সেই ডেটাসেট তত বেশি অবিশুদ্ধ (impure) বা অসংগঠিত (disordered) হবে।
আর Entropy যত কম হবে, সেটি তত বেশি বিশুদ্ধ (pure) হবে।

উদাহরণ **(Example):**

ধরা যাক, একজন প্রার্থী (candidate) একটি চাকরির অফার পেয়েছে এবং সে সিদ্ধান্ত নিতে চায় —
সে অফারটি গ্রহণ করবে নাকি প্রত্যাখ্যান করবে।

এই সমস্যার সমাধান করতে Decision Tree নিম্নলিখিতভাবে কাজ করে —

- Root Node-এ থাকে Salary attribute (বেতন)।
- Root Node বিভক্ত হয় পরবর্তী Decision Node-এ (যেমন অফিসের দূরত্ব বা *distance from office* অনুযায়ী) এবং একটি Leaf Node-এ (যেখানে চূড়ান্ত সিদ্ধান্ত থাকে)।
- পরের Decision Node আবার ভাগ হয় — একদিকে Cab Facility (যাতায়াত সুবিধা)-এর উপর ভিত্তি করে আরেকটি Decision Node এবং অন্যদিকে একটি Leaf Node।
- শেষ পর্যন্ত Decision Node দুটি Leaf Node-এ বিভক্ত হয় —
    1. Accepted Offer (অফার গ্রহণ)
    2. Declined Offer (অফার প্রত্যাখ্যান)



উদাহরণ **(Example):**

ধরা যাক সেট **X = {a, a, a, b, b, b, b, b}**

- মোট উদাহরণ (Total instances): ৮
- **b**-এর উদাহরণ সংখ্যা (Instances of b): ৫
- **a**-এর উদাহরণ সংখ্যা (Instances of a): ৩

Entropy=[( + )]

$=-[0.375(-1.415)+0.625(-0.678)]$

$=-(-0.53-0.424)$

$=0.954$

Information Gain ব্যবহার করে **Decision Tree** তৈরি করার মূল ধাপসমূহ**:**

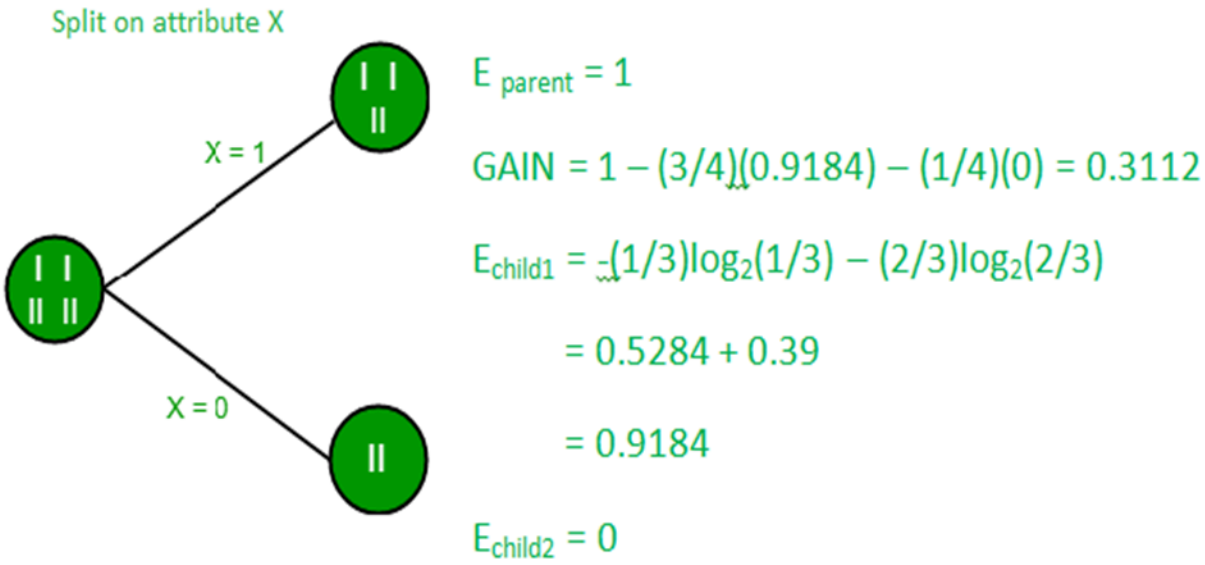
● শুরু করো সমস্ত training instances দিয়ে, যা root node-এর সাথে যুক্ত থাকবে।
● প্রতিটি নোডের জন্য কোন attribute ব্যবহার করা হবে তা Information Gain-এর মাধ্যমে নির্বাচন করো।
● লক্ষ্য করো: কোনো root-to-leaf path-এ একই discrete attribute দুইবার ব্যবহার করা যাবে না।
● প্রতিটি subtree পুনরাবৃত্তিমূলকভাবে (recursively) তৈরি করো, সেই subset-এর উপর যা ঐ path অনুযায়ী tree-তে classify হবে।
● যদি সবই positive বা সবই negative instances থাকে, তাহলে নোডটিকে সেই অনুযায়ী “yes” বা “no” লেবেল দাও।
● যদি কোনো attribute বাকি না থাকে, তাহলে নোডটিকে বাকি training instances-এর majority vote অনুযায়ী লেবেল করো।
● যদি কোনো instance বাকি না থাকে, তাহলে নোডটিকে parent node-এর training instances-এর majority vote অনুযায়ী লেবেল করো।

উদাহরণ **(Example):**
এবার Information Gain ব্যবহার করে নিম্নলিখিত ডেটার জন্য Decision Tree আঁকা যাক। Training Set: ৩টি বৈশিষ্ট্য (features) এবং ২টি শ্রেণি (classes)।

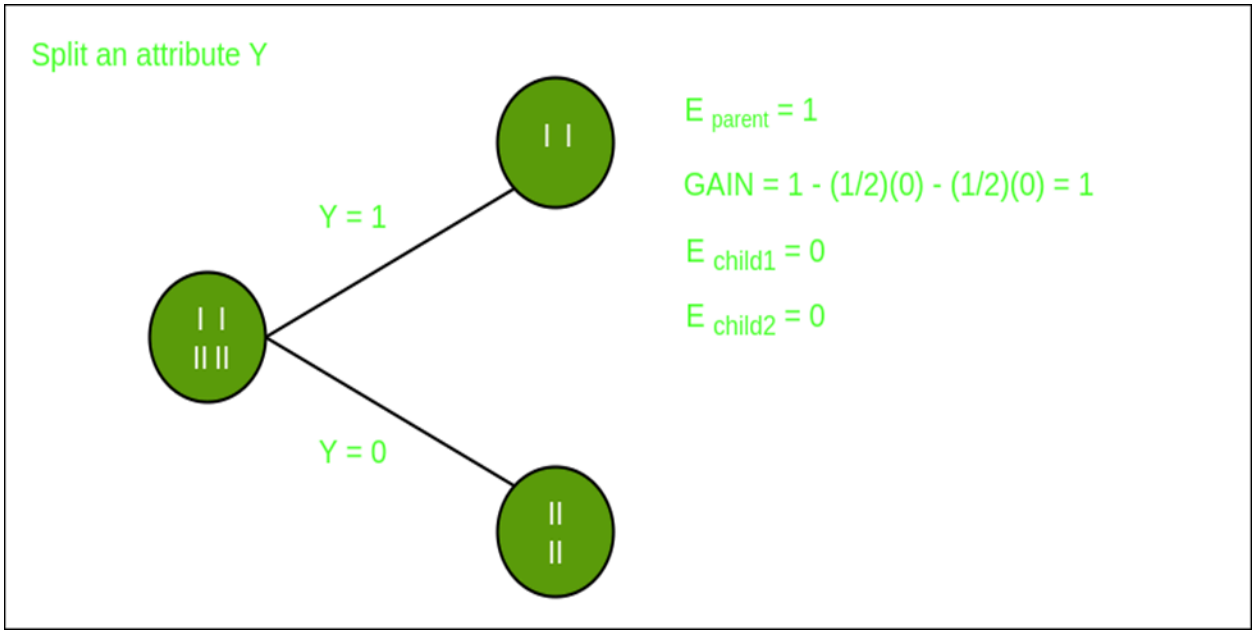
X	**Y**	**Z**	**C**
1	1	1	I

1	1	0	I
0	0	1	II
1	0	0	II

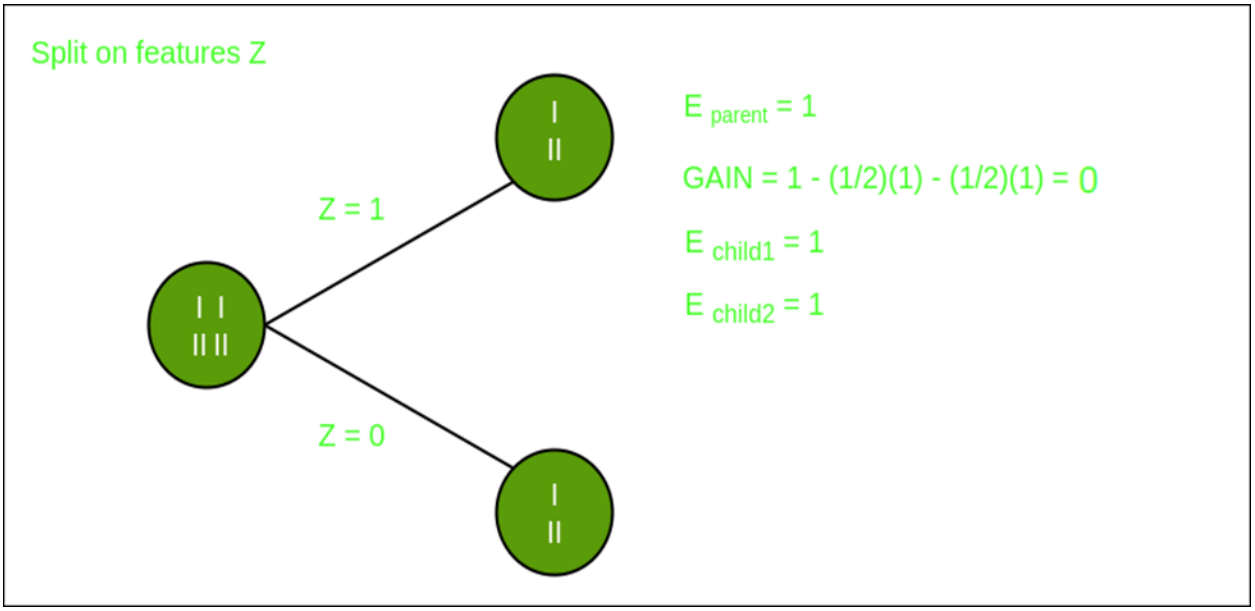
এখানে আমাদের কাছে ৩টি বৈশিষ্ট্য (features) এবং ২টি আউটপুট শ্রেণি (output classes) আছে। Information Gain ব্যবহার করে Decision Tree তৈরি করতে, আমরা প্রতিটি feature-কে বিবেচনা করব এবং প্রতিটি feature-এর জন্য Information Gain হিসাব করব।



Feature X-এর উপর ভিত্তি করে বিভাজন **(Split)**



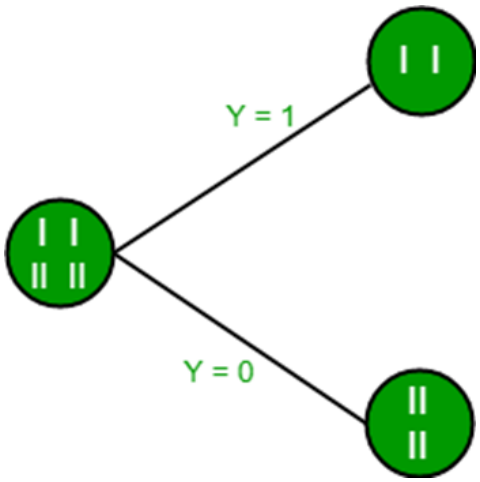
Feature Y-এর উপর ভিত্তি করে বিভাজন **(Split)**



Feature Z-এর উপর ভিত্তি করে বিভাজন **(Split)**

উপরের ছবিগুলো থেকে আমরা দেখতে পাচ্ছি যে Feature Y-এর উপর বিভাজন (split) করার সময় Information Gain সর্বাধিক। তাই Root Node-এর জন্য সবচেয়ে উপযুক্ত বৈশিষ্ট্য হলো Feature Y। এখন আমরা দেখতে পাচ্ছি যে Feature Y দিয়ে ডেটাসেট বিভাজন করার সময়, child node-এ লক্ষ্য ভেরিয়েবল (target variable)-এর একটি

pure subset থাকে। তাই আমাদের আর ডেটাসেটকে আরও ভাগ করার প্রয়োজন নেই। উপরের ডেটাসেটের জন্য চূড়ান্ত Decision Tree এরূপ হবে:



2. Gini Index

- Gini Index হলো একটি পরিমাপ, যা দেখায় যে একটি এলোমেলোভাবে নির্বাচিত উপাদান কতবার ভুলভাবে শনাক্ত হবে।
- এর মানে, কম Gini Index-যুক্ত attribute-কে বেশি প্রাধান্য দেওয়া উচিত।
- Sklearn-এ Gini Index-এর জন্য “Gini” ক্রাইটেরিয়া সমর্থিত, এবং ডিফল্ট মান হিসেবে এটি “gini” নেয়।
- Gini Index হিসাবের সূত্র (Formula) নিচে দেওয়া হলো।

Gini Index-এর সূত্র**:**

$$\text{Gini}(S) = 1 - \sum_{i=1}^{c} p_i^2$$

Gini Index হলো একটি পরিমাপ, যা দেখায় কোনো বণ্টনের (distribution) অসাম্যতা বা অশুদ্ধতা (impurity) কতটা। এটি Decision Tree এবং অন্যান্য মেশিন লার্নিং অ্যালগরিদমে ব্যবহার করা হয়। এর মান সাধারণত 0 থেকে 0.5 এর মধ্যে থাকে, যেখানে 0 নির্দেশ করে একটি শুদ্ধ (pure) সেট (সব উদাহরণ একই শ্রেণির), এবং 0.5 নির্দেশ করে একটি সর্বাধিক অশুদ্ধ (maximally impure) সেট (উদাহরণগুলো সমানভাবে বিভিন্ন শ্রেণিতে বিতরণ)।

Gini Index-এর অতিরিক্ত বৈশিষ্ট্যসমূহ:

- এটি হিসাব করা হয় প্রতিটি সম্ভাব্য ফলাফলের বর্গমূল (squared probabilities) যোগ করে এবং 1 থেকে বিয়োগ

করে।

- কম Gini Index মানে একটি বেশি homogeneous বা pure distribution, এবং বেশি Gini Index মানে heterogeneous বা impure distribution।
- Decision Tree-তে, Gini Index ব্যবহার করা হয় বিভাজনের (split) মান যাচাই করতে, যেখানে Parent node-এর impurity এবং Child node-এর weighted impurity-এর পার্থক্য পরিমাপ করা হয়।
- Entropy-এর মতো অন্যান্য impurity measure-এর তুলনায়, Gini Index দ্রুত হিসাব করা যায় এবং class probability-এর পরিবর্তনের প্রতি বেশি সংবেদনশীল।
- একটি অসুবিধা হলো, Gini Index প্রায়ই এমন split পছন্দ করে যা সমান আকারের child node তৈরি করে, যদিও তা classification accuracy-এর জন্য সর্বোত্তম নাও হতে পারে।
- ব্যবহারিকভাবে, Gini Index বা অন্যান্য impurity measure-এর মধ্যে পছন্দ নির্ভর করে নির্দিষ্ট সমস্যা এবং dataset-এর উপর এবং প্রায়শই পরীক্ষা ও tuning প্রয়োজন হয়।

Decision Tree Algorithm-এর উদাহরণ

Forecasting Activities Using Weather Information

- Root Node: পুরো dataset
- Attribute: “Outlook” (sunny, cloudy, rainy)
- Subsets: Overcast, Rainy, Sunny
- Recursive Splitting: উদাহরণস্বরূপ, sunny subset-কে humidity অনুযায়ী আরও ভাগ করা।
- Leaf Nodes: Activities যেমন “swimming,” “hiking,” “staying inside”।

Decision Tree গঠন প্রক্রিয়া**:**

1. পুরো dataset-কে Root Node হিসেবে ধরো।
2. Information Gain-এর মাধ্যমে best attribute নির্বাচন করো। সূত্র:

Information Gain = Entropy(parent) − [Weighted average] × Entropy(children)

Best attribute-এর জন্য একটি নতুন internal node তৈরি করো এবং root node-এর সাথে সংযুক্ত করো। উদাহরণ: যদি best attribute হয় “Outlook” (sunny, overcast, rainy), তাহলে নতুন node-এর লেবেল হবে “Outlook”।

3. Best attribute-এর মান অনুযায়ী dataset-কে subsets-এ ভাগ করো।
4. প্রতিটি subset-এর জন্য steps 1-4 পুনরাবৃত্তি করো যতক্ষণ না সব উদাহরণ একই শ্রেণিতে চলে আসে বা আর split সম্ভব না হয়।

5. প্রতিটি subset-কে leaf node দিয়ে লেবেল করো, যেখানে একই শ্রেণির উদাহরণ থাকে।

6. Decision Tree ব্যবহার করে prediction করো, root node থেকে leaf node পর্যন্ত traversal করে। উদাহরণ: Outlook sunny এবং humidity high হলে, path অনুসরণ করে leaf node “swimming” এ পৌঁছাবে।

ID3 Algorithm:

Iterative Dichotomiser 3 (ID3) হলো Decision Tree গঠনের একটি পরিচিত অ্যালগরিদম। এটি প্রতিটি node-এ best attribute নির্বাচন করে তথ্য লাভ (information gain) অনুযায়ী tree তৈরি করে। লক্ষ্য হলো final subsets möglichst homogeneous করা।

ID3-এর কাজ করার পদ্ধতি:

1. **Best Attribute** নির্বাচন**:** Entropy এবং Information Gain ব্যবহার করে attribute নির্বাচন করা হয়।

2. **Tree Nodes** তৈরি**:** নির্বাচনকৃত attribute অনুযায়ী dataset subdivide করা হয়। প্রতিটি subset-এর জন্য আবার best attribute নির্বাচন করা হয়।

3. **Stopping Criteria:** Recursion চলতে থাকে যতক্ষণ না সব উদাহরণ একই শ্রেণির হয় বা attribute শেষ হয়ে যায়।

4. **Missing Values Handle** করা**:** Attribute mean/mode substitution বা majority class ব্যবহার করা যেতে পারে।

5. **Tree Pruning:** Overfitting কমানোর জন্য অপ্রয়োজনীয় node সরানো। C4.5-এর মতো variation pruning অন্তর্ভুক্ত করে।

Decision Tree-এর সুবিধা**:**

- সহজে বোঝা যায় এবং interpret করা যায়।
- Numerical এবং categorical data উভয় পরিচালনা করতে পারে।
- Decision-making এর জন্য feature importance সম্পর্কে insight দেয়।
- Missing values এবং outliers সহজে handle করতে পারে।
- Classification এবং regression উভয় কাজে প্রয়োগযোগ্য।

Decision Tree-এর অসুবিধা**:**

- Overfitting-এর সম্ভাবনা থাকে।
- ডেটার ছোট পরিবর্তনের প্রতি সংবেদনশীল, training data represent না করলে limited generalization।
- Imbalanced data থাকলে bias-এর সম্ভাবনা।

Pruning: Optimal Decision Tree:

Pruning হলো অপ্রয়োজনীয় node সরিয়ে optimal decision tree তৈরি করার প্রক্রিয়া। Too-large tree → overfitting, ছোট tree → important feature capture না করা।

প্রধান pruning technique:
- Cost Complexity Pruning
- Reduced Error Pruning

Underfitting এবং Overfitting

Machine Learning model-এর performance এবং prediction error-এর উপর নির্ভর করে।

- **Bias:** Bias হলো সেই ত্রুটি (error), যা Learning Algorithm-এর অতিরিক্ত সরল অনুমানের (overly simplistic assumptions) কারণে ঘটে। এই অনুমানগুলো মডেলকে সহজে বোঝা এবং শেখা সম্ভব করে তোলে, কিন্তু ডেটার মূল জটিলতা (underlying complexities) ধরতে পারে না। এটি সেই ত্রুটি, যা মডেল input এবং output-এর মধ্যে সঠিক সম্পর্ক সঠিকভাবে উপস্থাপন করতে ব্যর্থ হওয়ার কারণে ঘটে। যদি কোনো মডেল training এবং testing উভয় ডেটায়ই খারাপ পারফরম্যান্স প্রদর্শন করে, তাহলে এর মানে হলো high bias, যা সরল মডেলের কারণে ঘটে এবং underfitting নির্দেশ করে। **High bias → underfitting**।
- **Variance:** অন্যদিকে, Variance হলো সেই ত্রুটি (error), যা মডেলের training data-এর পরিবর্তনের প্রতি সংবেদনশীলতার (sensitivity) কারণে ঘটে। এটি নির্দেশ করে মডেলের ভিন্ন training instance-এর জন্য prediction-এর পরিবর্তনশীলতা (variability)। যখন কোনো মডেল training data-এর noise এবং random fluctuations শিখে মূল প্যাটার্ন (underlying pattern) না শিখে, তখন high variance দেখা দেয়। ফলস্বরূপ, মডেল training data-তে ভালো পারফরম্যান্স দেখায়, কিন্তু testing data-তে খারাপ পারফরম্যান্স দেয়, যা overfitting নির্দেশ করে। **High variance → overfitting**।

Machine Learning-এ Underfitting:

যখন কোনো statistical model বা machine learning algorithm ডেটার জটিলতা ধরতে যথেষ্ট শক্তিশালী নয়, তখন বলা হয় মডেলটি underfitting করেছে। এটি নির্দেশ করে মডেলটি training data-কে কার্যকরভাবে শেখার অক্ষমতা, যার ফলে training এবং testing উভয় ডেটাতেই পারফরম্যান্স খারাপ হয়। সহজভাবে বলতে গেলে, underfit মডেল নতুন বা unseen উদাহরণে অসঠিক ফলাফল দেয়। এটি সাধারণত ঘটে যখন খুব সরল মডেল ব্যবহার করা হয় এবং অতিরিক্ত সরল অনুমান করা হয়।

Underfitting সমাধানের জন্য:

- বেশি complex models ব্যবহার করতে হবে।
- feature representation উন্নত করতে হবে।
- regularization কমাতে হবে।

নোট**:** Underfitting মডেল-এর ক্ষেত্রে High Bias এবং Low Variance থাকে।

Underfitting-এর কারণসমূহ**:**

1. মডেল খুব সরল, তাই ডেটার জটিলতা ধরতে অক্ষম।
2. Training-এর জন্য ব্যবহারকৃত input features লক্ষ্য ভেরিয়েবলকে প্রভাবিত করা underlying factors-এর যথেষ্ট প্রতিনিধিত্ব করে না।
3. Training dataset-এর আকার অপর্যাপ্ত।
4. Overfitting প্রতিরোধে অতিরিক্ত regularization ব্যবহার করা হয়েছে, যা মডেলকে ডেটা ধরতে বাধা দেয়।
5. Features properly scale করা হয়নি।

Underfitting কমানোর কৌশলসমূহ**:**

1. মডেলের জটিলতা (complexity) বাড়ানো।
2. Feature সংখ্যা বাড়ানো এবং feature engineering করা।
3. ডেটা থেকে noise সরানো।
4. Training-এর সময়কাল বা epochs বাড়ানো।

Machine Learning-এ Overfitting:

যখন মডেল testing data-এ সঠিক prediction দিতে পারে না, তখন বলা হয় মডেলটি overfitted।

- মডেল যখন খুব বেশি ডেটা দিয়ে train করা হয়, তখন এটি noise এবং inaccurate data- থেকেও শিখতে শুরু করে।
- Testing data-এ high variance দেখা দেয় এবং মডেল ডেটা সঠিকভাবে categorize করতে পারে না, কারণ এতে অতিরিক্ত বিস্তারিত এবং noise থাকে।

- Overfitting-এর প্রধান কারণ হলো non-parametric এবং non-linear methods, যেগুলো মডেল তৈরি করার ক্ষেত্রে বেশি স্বাধীনতা রাখে এবং তাই কখনও কখনও অযৌক্তিক মডেল তৈরি করতে পারে।

Overfitting প্রতিরোধের সমাধান**:**

- Linear data হলে linear algorithm ব্যবহার করা।
- Decision Tree ব্যবহার করলে maximal depth নির্ধারণ করা।

সারসংক্ষেপে, Overfitting হলো এমন সমস্যা যেখানে training data-এর performance এবং unseen data-এর performance পার্থক্য থাকে।

Overfitting-এর কারণসমূহ**:**

1. High Variance এবং Low Bias।
2. মডেল খুব জটিল।
3. Training data-এর আকার।

Overfitting কমানোর কৌশলসমূহ**:**

1. Training data-এর মান উন্নত করা, যাতে model শুধুমাত্র গুরুত্বপূর্ণ patterns শিখে এবং noise বা অপ্রয়োজনীয় feature-এর সাথে fit না হয়।
2. Training data-এর পরিমাণ বাড়ানো, যাতে model unseen data-এ generalize করতে পারে।
3. Model complexity কমানো।
4. Training-এর সময় early stopping ব্যবহার করা (যখন loss বাড়তে শুরু করে training থামানো)।
5. Ridge Regularization এবং Lasso Regularization ব্যবহার।
6. Neural Network-এ dropout ব্যবহার করা overfitting কমাতে।

উপসংহার **(Conclusion):**

Decision Tree হলো machine learning-এর একটি গুরুত্বপূর্ণ টুল, যা input data-এর ভিত্তিতে outcomes model এবং predict করে tree-এর মতো কাঠামোর মাধ্যমে।

- এগুলো interpretability, versatility, এবং সহজ visualisation প্রদান করে, যার ফলে categorization এবং regression উভয় কাজেই গুরুত্বপূর্ণ।

- Decision Tree-এর সুবিধা হলো সহজে বোঝা যায়, কিন্তু এগুলো overfitting-এর মতো চ্যালেঞ্জের মুখোমুখি হতে পারে।

- Decision Tree-এর terminologies এবং formation process বোঝা কার্যকরভাবে বিভিন্ন পরিস্থিতিতে এগুলো ব্যবহার করার জন্য অপরিহার্য।

SEMESTER IV - UNIT 4B

আনসুপারভাইজড লার্নিং কী?

যেমন নামটি নির্দেশ করছে, আনসুপারভাইজড লার্নিং হলো একটি মেশিন লার্নিং কৌশল যেখানে মডেলগুলোকে প্রশিক্ষণ ডেটাসেট ব্যবহার করে সরাসরি তত্ত্বাবধানে রাখা হয় না। বরং, মডেলগুলো নিজেই প্রদত্ত ডেটা থেকে লুকানো প্যাটার্ন এবং তথ্য খুঁজে বের করে। এটি মানুষের মস্তিষ্কে নতুন কিছু শেখার প্রক্রিয়ার সঙ্গে তুলনা করা যায়।

সংজ্ঞা: আনসুপারভাইজড লার্নিং হলো এমন একটি মেশিন লার্নিং-এর ধরণ যেখানে মডেলগুলো লেবেলবিহীন ডেটাসেট ব্যবহার করে প্রশিক্ষিত হয় এবং কোনো তত্ত্বাবধানে না থেকে ডেটার উপর কাজ করতে পারে।

আনসুপারভাইজড লার্নিং সরাসরি রিগ্রেশন বা ক্লাসিফিকেশন সমস্যায় প্রয়োগ করা যায় না, কারণ সুপারভাইজড লার্নিং-এর মতো আমাদের কাছে ইনপুট ডেটা থাকলেও তার সাথে সম্পর্কিত আউটপুট ডেটা নেই। আনসুপারভাইজড লার্নিং-এর মূল লক্ষ্য হলো ডেটাসেটের অন্তর্নিহিত কাঠামো খুঁজে বের করা, ডেটাগুলোকে মিলের ভিত্তিতে গ্রুপ করা এবং সেই ডেটাসেটকে সংক্ষিপ্ত আকারে উপস্থাপন করা।

উদাহরণ: ধরুন আনসুপারভাইজড লার্নিং অ্যালগরিদমকে বিভিন্ন ধরণের বিড়াল এবং কুকুরের ছবি দেওয়া হয়েছে। অ্যালগরিদম কখনোও প্রদত্ত ডেটাসেট নিয়ে প্রশিক্ষিত হয়নি, অর্থাৎ এটি ডেটাসেটের বৈশিষ্ট্য সম্পর্কে কোনো ধারণা রাখে না। আনসুপারভাইজড লার্নিং অ্যালগরিদমের কাজ হলো নিজেই ছবির বৈশিষ্ট্য শনাক্ত করা। অ্যালগরিদম এই কাজটি সম্পন্ন করবে ছবিগুলোকে মিলের ভিত্তিতে গ্রুপ বা ক্লাস্টারে ভাগ করে।

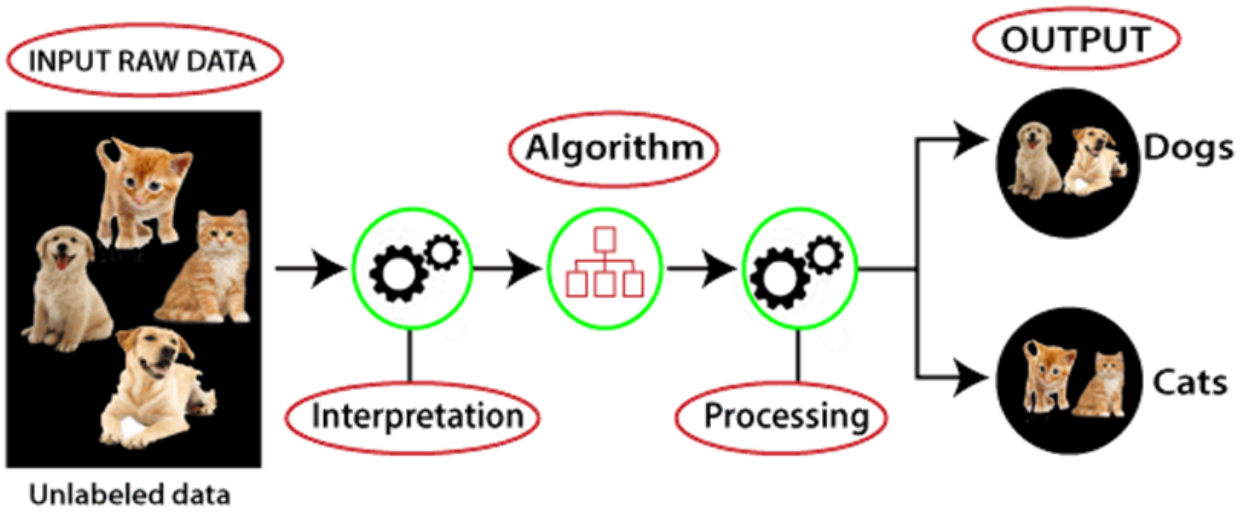
কেন আনসুপারভাইজড লার্নিং ব্যবহার করা হয়?

নিচে কিছু মূল কারণ উল্লেখ করা হলো যা আনসুপারভাইজড লার্নিং-এর গুরুত্ব ব্যাখ্যা করে:

- আনসুপারভাইজড লার্নিং ডেটা থেকে মূল্যবান অন্তর্দৃষ্টি (insights) খুঁজে বের করতে সহায়ক।
- এটি মানুষের মতো নিজের অভিজ্ঞতার মাধ্যমে চিন্তা করার প্রক্রিয়ার সাথে মিল রয়েছে, যা এটিকে প্রকৃত AI-এর আরও কাছাকাছি করে।
- আনসুপারভাইজড লার্নিং লেবেলবিহীন এবং শ্রেণিবিহীন ডেটার উপর কাজ করে, যা এটিকে আরও গুরুত্বপূর্ণ করে তোলে।
- বাস্তব জগতে সব সময় ইনপুট ডেটার সাথে সংশ্লিষ্ট আউটপুট পাওয়া যায় না, তাই এমন পরিস্থিতি সমাধানের জন্য আনসুপারভাইজড লার্নিং প্রয়োজন।

আনসুপারভাইজড লার্নিং-এর কাজ করার পদ্ধতি:

আনসুপারভাইজড লার্নিং-এর কাজ করার প্রক্রিয়া নিচের চিত্রের মাধ্যমে বোঝা যায়:

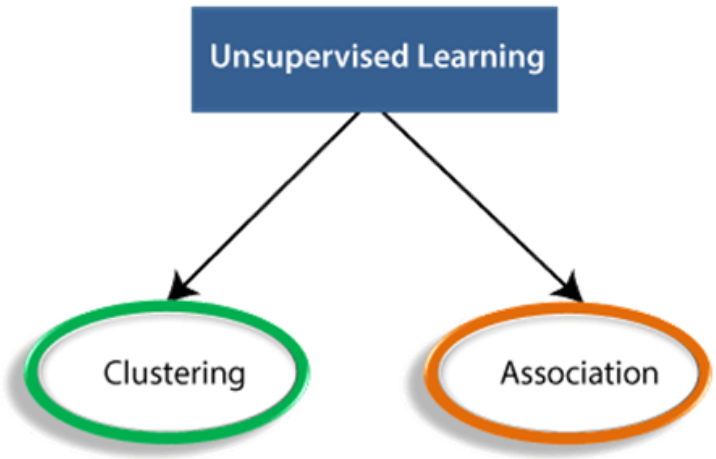


এখানে আমরা একটি লেবেলবিহীন ইনপুট ডেটা নিয়েছি, অর্থাৎ এটি কোনোভাবে শ্রেণিবদ্ধ নয় এবং সংশ্লিষ্ট আউটপুটও দেওয়া হয়নি। এখন এই লেবেলবিহীন ইনপুট ডেটা মেশিন লার্নিং মডেলে প্রশিক্ষণের জন্য ফিড করা হয়। প্রথমে, মডেল কাঁচা ডেটা বিশ্লেষণ করে ডেটার লুকানো প্যাটার্ন খুঁজে বের করবে এবং তারপর উপযুক্ত অ্যালগরিদম প্রয়োগ করবে, যেমন K-Means ক্লাস্টারিং, Decision Tree ইত্যাদি।

একবার উপযুক্ত অ্যালগরিদম প্রয়োগ করা হলে, অ্যালগরিদমটি ডেটার অবজেক্টগুলোকে তাদের মিল এবং ভিন্নতার ভিত্তিতে গ্রুপে ভাগ করবে।

আনসুপারভাইজড লার্নিং অ্যালগরিদমের ধরন:

আনসুপারভাইজড লার্নিং অ্যালগরিদমকে মূলত দুই ধরণের সমস্যায় ভাগ করা যেতে পারে:



ক্লাস্টারিং **(Clustering):** ক্লাস্টারিং হলো একটি পদ্ধতি যেখানে অবজেক্টগুলোকে এমনভাবে গ্রুপ বা ক্লাস্টারে ভাগ করা হয় যাতে সবচেয়ে বেশি মিল থাকা অবজেক্টগুলো একই গ্রুপে থাকে এবং অন্য গ্রুপের অবজেক্টের সাথে কম বা কোনো মিল না থাকে। ক্লাস্টার বিশ্লেষণ (Cluster Analysis) ডেটার অবজেক্টগুলোর মধ্যে সাধারণ বৈশিষ্ট্য খুঁজে বের করে এবং সেই বৈশিষ্ট্যের উপস্থিতি ও অনুপস্থিতির ভিত্তিতে অবজেক্টগুলোকে শ্রেণীবদ্ধ করে।

অ্যাসোসিয়েশন **(Association):**অ্যাসোসিয়েশন রুল হলো আনসুপারভাইজড লার্নিং-এর একটি প দ্ধতি যা বড় ডেটাবেসের ভেরিয়েবলগুলোর মধ্যে সম্পর্ক খুঁজে বের করতে ব্যবহৃত হয়। এটি নির্ধারণ করে কোন আইটেমগুলো ডেটাসেটে একসাথে উপস্থিত থাকে। অ্যাসোসিয়েশন রুল মার্কেটিং স্ট্র্যাটেজি আরও কার্যকর করে। যেমন, যারা X আইটেম (ধরা যাক রুটি) কিনে, তারা সাধারণত Y আইটেম (বাটার/জ্যাম) ও কিনে। অ্যাসোসিয়েশন রুলের একটি সাধারণ উদাহরণ হলো মার্কেট বাস্কেট অ্যানালিসিস (Market Basket Analysis)।

নিচে কিছু জনপ্রিয় আনসুপারভাইজড লার্নিং অ্যালগরিদমের তালিকা দেওয়া হলো:

- K-Means ক্লাস্টারিং

- হায়ারার্কিকাল ক্লাস্টারিং (Hierarchical Clustering)
- অ্যানোমালি ডিটেকশন (Anomaly Detection)
- নিউরাল নেটওয়ার্কস (Neural Networks)
- প্রিন্সিপাল কম্পোনেন্ট অ্যানালিসিস (Principal Component Analysis)
- ইনডিপেনডেন্ট কম্পোনেন্ট অ্যানালিসিস (Independent Component Analysis)
- এপ্রিওরি অ্যালগরিদম (Apriori Algorithm)
- সিঙ্গুলার ভ্যালু ডিকম্পোজিশন (Singular Value Decomposition)

আনসুপারভাইজড লার্নিং-এর সুবিধা:

- আনসুপারভাইজড লার্নিং ব্যবহার করা হয় আরও জটিল কাজের জন্য, কারণ এখানে লেবেলযুক্ত ইনপুট ডেটা নেই।
- আনসুপারভাইজড লার্নিং প্রাধান্য পায়, কারণ লেবেলবিহীন ডেটা পাওয়া সহজ, লেবেলযুক্ত ডেটার তুলনায়।

আনসুপারভাইজড লার্নিং-এর অসুবিধা:

- আনসুপারভাইজড লার্নিং প্রকৃতিগতভাবে সুপারভাইজড লার্নিং-এর চেয়ে বেশি কঠিন, কারণ এখানে সংশ্লিষ্ট আউটপুট নেই।
- আনসুপারভাইজড লার্নিং-এর ফলাফল তুলনামূলকভাবে কম সঠিক হতে পারে, কারণ ইনপুট ডেটা লেবেলযুক্ত নয় এবং অ্যালগরিদম আগেই আউটপুট জানে না।

আনসুপারভাইজড লার্নিং-এর ব্যবহার:

আনসুপারভাইজড লার্নিং বিভিন্ন সমস্যার সমাধান করতে ব্যবহার করা যেতে পারে, যেমন:

- অ্যানোমালি ডিটেকশন **(Anomaly Detection):** আনসুপারভাইজড লার্নিং ডেটার অস্বাভাবিক প্যাটার্ন বা ব্যত্যয় চিহ্নিত করতে পারে, যা ফ্রড, হ্যাকিং বা সিস্টেম ত্রুটি শনাক্ত করতে সহায়ক।
- বৈজ্ঞানিক আবিষ্কার **(Scientific Discovery):** বৈজ্ঞানিক ডেটায় লুকানো সম্পর্ক ও প্যাটার্ন খুঁজে নতুন

হাইপোথিসিস এবং অন্তর্দৃষ্টি পাওয়া যায়।

- রেকমেন্ডেশন সিস্টেম **(Recommendation Systems):** ব্যবহারকারীর আচরণ ও পছন্দের প্যাটার্ন এবং মিল খুঁজে প্রোডাক্ট, সিনেমা বা মিউজিক সাজেশন দেওয়া যায়।
- কাস্টমার সেগমেন্টেশন **(Customer Segmentation):** একই ধরনের বৈশিষ্ট্যের গ্রাহকরা কোন গ্রুপে পড়ে তা চিহ্নিত করে ব্যবসায়িক মার্কেটিং এবং গ্রাহক সেবা উন্নত করা যায়।
- ইমেজ অ্যানালাইসিস **(Image Analysis):** ছবির কনটেন্ট অনুযায়ী ছবিগুলো গ্রুপ করা যায়, যা ইমেজ ক্লাসিফিকেশন, অবজেক্ট ডিটেকশন এবং ইমেজ রিট্রিভালের কাজ সহজ করে।

সুপারভাইজড এবং আনসুপারভাইজড লার্নিং-এর তুলনা:

প্যারামিটার	সুপারভাইজড মেশিন লার্নিং	আনসুপারভাইজড মেশিন লার্নিং
ইনপুট ডেটা	অ্যালগরিদমগুলো লেবেলযুক্ত ডেটা ব্যবহার করে প্রশিক্ষিত হয়	অ্যালগরিদমগুলো লেবেলবিহীন ডেটার উপর কাজ করে
কম্পিউটেশনাল জটিলতা	সহজ পদ্ধতি	জটিল কম্পিউটেশনাল প্রক্রিয়া
সঠিকতা	খুব সঠিক	কম সঠিক
ক্লাসের সংখ্যা	ক্লাসের সংখ্যা জানা থাকে	ক্লাসের সংখ্যা জানা থাকে না
ডেটা বিশ্লেষণ	অফলাইন বিশ্লেষণ ব্যবহার করে	ডেটার রিয়েল-টাইম বিশ্লেষণ ব্যবহার করে
ব্যবহৃত অ্যালগরিদম	লিনিয়ার ও লজিস্টিক রিগ্রেশন, র‍্যান্ডম ফোরেস্ট, মাল্টি-ক্লাস ক্লাসিফিকেশন, ডিসিশন ট্রি, সাপোর্ট ভেক্টর মেশিন, নিউরাল নেটওয়ার্ক ইত্যাদি	K-Means ক্লাস্টারিং, হায়ারার্কিকাল ক্লাস্টারিং, KNN, অ্যাপ্রিওরি অ্যালগরিদম ইত্যাদি
আউটপুট	কাঙ্ক্ষিত আউটপুট দেওয়া হয়	কাঙ্ক্ষিত আউটপুট দেওয়া হয় না

প্রশিক্ষণ ডেটা	প্রশিক্ষণ ডেটা ব্যবহার করে মডেল তৈরি করা হয়	কোনো প্রশিক্ষণ ডেটা ব্যবহার করা হয় না
জটিল মডেল	সুপারভাইজড লার্নিং-এর চেয়ে বড় এবং জটিল মডেল শেখা সম্ভব নয়	আনসুপারভাইজড লার্নিং-এ বড় এবং জটিল মডেল শেখা সম্ভব
মডেল	আমরা মডেল পরীক্ষা করতে পারি	আমরা মডেল পরীক্ষা করতে পারি না
ডাকা হয়	সুপারভাইজড লার্নিংকে ক্লাসিফিকেশনও বলা হয়	আনসুপারভাইজড লার্নিংকে ক্লাস্টারিংও বলা হয়
উদাহরণ	উদাহরণ: অপটিক্যাল ক্যারেক্টার রিকগনিশন (OCR)	উদাহরণ: কোনো ছবিতে মুখ সনাক্ত করা
সুপারভিশন	মডেল প্রশিক্ষণের জন্য সুপারভিশন প্রয়োজন	মডেল প্রশিক্ষণের জন্য কোনো সুপারভিশন প্রয়োজন হয় না

মেশিন লার্নিং-এ ক্লাস্টারিং:

ক্লাস্টারিং বা ক্লাস্টার বিশ্লেষণ হলো একটি মেশিন লার্নিং কৌশল, যা লেবেলবিহীন ডেটাসেটকে গ্রুপে ভাগ করে। এটিকে সংজ্ঞায়িত করা যায় এইভাবে: "একটি পদ্ধতি যার মাধ্যমে ডেটা পয়েন্টগুলোকে বিভিন্ন ক্লাস্টারে ভাগ করা হয়, যেখানে একই ধরনের ডেটা পয়েন্ট একই গ্রুপে থাকে এবং অন্য গ্রুপের ডেটার সঙ্গে কম বা কোনো মিল থাকে না।" ক্লাস্টারিং লেবেলবিহীন ডেটাসেটে কিছু সাদৃশ্যপূর্ণ প্যাটার্ন যেমন আকার, মাপ, রঙ, আচরণ ইত্যাদি খুঁজে বের করে এবং সেই অনুযায়ী ডেটাকে ভাগ করে। এটি একটি আনসুপারভাইজড লার্নিং পদ্ধতি, তাই অ্যালগরিদমকে কোনো সুপারভিশন দেওয়া হয় না।

ক্লাস্টারিং কিছুটা ক্লাসিফিকেশন অ্যালগরিদমের মতো, তবে প্রধান পার্থক্য হলো ব্যবহার করা ডেটাসেটের ধরন। ক্লাসিফিকেশনে লেবেলযুক্ত ডেটা ব্যবহৃত হয়, আর ক্লাস্টারিং-এ লেবেলবিহীন ডেটা ব্যবহৃত হয়।

ক্লাস্টারিং ও ক্লাসিফিকেশন-এর পার্থক্য

বিষয়	Clustering	Classification

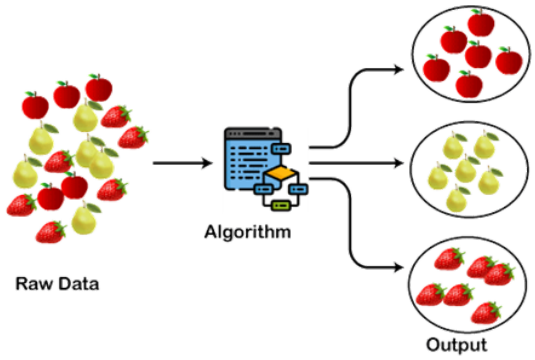
লার্নিং টাইপ	Unsupervised Learning	Supervised Learning
ডেটার ধরন	লেবেলবিহীন (Unlabeled Data)	লেবেলযুক্ত (Labeled Data)
আউটপুট	নতুন ক্লাস্টার বা গ্রুপ তৈরি করে	নির্দিষ্ট শ্রেণিতে ভাগ করে
উদ্দেশ্য	ডেটার প্যাটার্ন বা গঠন নির্ণয়	পূর্বনির্ধারিত শ্রেণি পূর্বাভাস
উদাহরণ	গ্রাহক বিভাজন	ফলের ধরন নির্ধারণ (আপেল, আম, কলা)

উদাহরণ: ধরা যাক, আমরা একটি শপিং মলে গিয়েছি। আমরা লক্ষ্য করি যে একই ধরনের ব্যবহারযোগ্য জিনিসগুলো একসাথে গ্রুপে রাখা হয়েছে। যেমন, টি-শার্টগুলো একটি সেকশনে, ট্রাউজার অন্য সেকশনে, সবজি সেকশনে আপেল, কলা, আম ইত্যাদি আলাদা আলাদা সেকশনে রাখা থাকে। এটি আমাদের জিনিসগুলো সহজে খুঁজে পেতে সাহায্য করে। ক্লাস্টারিং কৌশলও একইভাবে কাজ করে।

ক্লাস্টারিং কৌশলের ব্যবহার:

- মার্কেট সেগমেন্টেশন
- স্ট্যাটিস্টিকাল ডেটা বিশ্লেষণ
- সামাজিক নেটওয়ার্ক বিশ্লেষণ
- ইমেজ সেগমেন্টেশন
- অ্যানোমালি ডিটেকশন ইত্যাদি

এই সাধারণ ব্যবহারের পাশাপাশি, আমাজন তার রিকমেন্ডেশন সিস্টেমে ক্লাস্টারিং ব্যবহার করে ব্যবহারকারীর পূর্বের সার্চ অনুযায়ী প্রোডাক্ট সাজেস্ট করার জন্য। নেটফ্লিক্সও এই কৌশল ব্যবহার করে মুভি ও ওয়েব-সিরিজের রিকমেন্ডেশন দেয়। নীচের চিত্রটি ক্লাস্টারিং অ্যালগরিদমের কাজ বোঝায়। আমরা দেখতে পাচ্ছি বিভিন্ন ফল বিভিন্ন গ্রুপে ভাগ হয়েছে, যেখানে একই ধরনের বৈশিষ্ট্যযুক্ত ফল একসাথে রয়েছে।



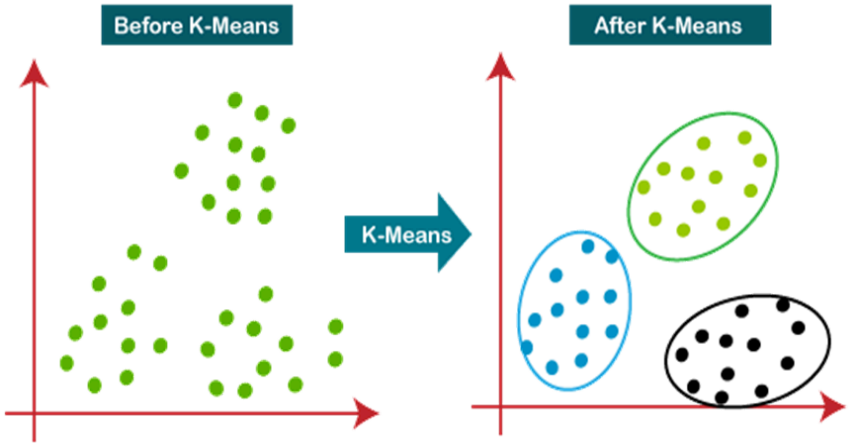
K-Means অ্যালগরিদম কী?

K-Means Clustering হলো একটি Unsupervised Learning (অপর্যবেক্ষিত শিক্ষণ) অ্যালগরিদম, যা অলেবেলড ডেটাসেটকে বিভিন্ন ক্লাস্টারে ভাগ করে। এখানে K মানে হলো পূর্বনির্ধারিত ক্লাস্টারের সংখ্যা যা তৈরি করতে হবে। যেমন, যদি K = 2 হয়, তবে দুটি ক্লাস্টার তৈরি হবে, আর যদি K = 3 হয়, তবে তিনটি ক্লাস্টার তৈরি হবে, এভাবে চলতে থাকে। এটি একটি পুনরাবৃত্তিমূলক (iterative) অ্যালগরিদম, যা অলেবেলড ডেটাসেটকে K সংখ্যক ভিন্ন ক্লাস্টারে ভাগ করে, এমনভাবে যে প্রতিটি ডেটা পয়েন্ট শুধুমাত্র একটি ক্লাস্টারের অন্তর্ভুক্ত হয় যার সাথে তার বৈশিষ্ট্য সবচেয়ে বেশি মেলে। এই অ্যালগরিদম আমাদেরকে ডেটাকে বিভিন্ন গ্রুপ বা শ্রেণিতে বিভক্ত করার সুযোগ দেয়, এবং এটি স্বয়ংক্রিয়ভাবে অলেবেলড ডেটাসেটে থাকা অজানা ক্যাটাগরি বা গ্রুপগুলো শনাক্ত করতে সক্ষম, কোনো প্রশিক্ষণ ছাড়াই।

এটি একটি centroid-based (কেন্দ্র-ভিত্তিক) অ্যালগরিদম, যেখানে প্রতিটি ক্লাস্টারের সাথে একটি centroid (কেন্দ্রবিন্দু) যুক্ত থাকে। এই অ্যালগরিদমের মূল উদ্দেশ্য হলো প্রতিটি ডেটা পয়েন্ট এবং তার সংশ্লিষ্ট ক্লাস্টারের কেন্দ্রবিন্দুর মধ্যে দূরত্বের যোগফলকে সর্বনিম্ন করা। অ্যালগরিদমটি অলেবেলড ডেটাসেটকে ইনপুট হিসেবে নেয়, তারপর সেটিকে K সংখ্যক ক্লাস্টারে বিভক্ত করে, এবং এই প্রক্রিয়াটি পুনরায় পুনরায় চালায় যতক্ষণ না পর্যন্ত এটি সবচেয়ে ভালো ক্লাস্টার বিন্যাস খুঁজে পায়। এই অ্যালগরিদমে K-এর মানটি পূর্বনির্ধারিত হতে হবে।

K-Means Clustering Algorithm প্রধানত দুটি কাজ সম্পাদন করে:
○ K সংখ্যক কেন্দ্রবিন্দুর (centroids) জন্য সেরা মান নির্ধারণ করা, যা একটি পুনরাবৃত্তিমূলক প্রক্রিয়ার মাধ্যমে সম্পন্ন হয়।
○ প্রতিটি ডেটা পয়েন্টকে তার নিকটতম কেন্দ্রবিন্দুর সাথে যুক্ত করা। যে ডেটা পয়েন্টগুলো কোনো নির্দিষ্ট কেন্দ্রবিন্দুর কাছাকাছি থাকে, তারা একত্রে একটি ক্লাস্টার গঠন করে।

ফলে, প্রতিটি ক্লাস্টারের মধ্যে থাকা ডেটা পয়েন্টগুলোর মধ্যে কিছু মিল বা সাদৃশ্য থাকে, এবং এক ক্লাস্টার অন্য ক্লাস্টার থেকে ভিন্ন বা দূরে অবস্থান করে। নীচের চিত্রটি K-Means Clustering Algorithm-এর কার্যপ্রণালীকে ব্যাখ্যা করে:



K-Means অ্যালগরিদম কীভাবে কাজ করে?

K-Means অ্যালগরিদমের কাজের ধাপগুলো নিচে ব্যাখ্যা করা হলোঃ

ধাপ-১: ক্লাস্টারের সংখ্যা নির্ধারণ করার জন্য একটি K মান নির্বাচন করো।

ধাপ-২: যেকোনো র‍্যান্ডম K পয়েন্ট বা সেন্ট্রয়েড (centroids) নির্বাচন করো। (এগুলো ইনপুট ডেটাসেটের বাইরেও হতে পারে।)

ধাপ-৩: প্রতিটি ডেটা পয়েন্টকে তার নিকটতম সেন্ট্রয়েডের সাথে যুক্ত করো, এর ফলে K সংখ্যক পূর্বনির্ধারিত ক্লাস্টার তৈরি হবে।

ধাপ-৪: প্রতিটি ক্লাস্টারের variance (বিচ্যুতি) হিসাব করো এবং প্রতিটি ক্লাস্টারের নতুন কেন্দ্রবিন্দু (centroid) নির্ধারণ করো।

ধাপ-৫: তৃতীয় ধাপটি পুনরাবৃত্তি করো, অর্থাৎ প্রতিটি ডেটা পয়েন্টকে তার নতুন নিকটতম সেন্ট্রয়েডের সাথে পুনরায় যুক্ত করো।

ধাপ-৬: যদি কোনো পুনঃনিয়োগ (reassignment) ঘটে, তাহলে ধাপ-৪ এ ফিরে যাও, অন্যথায় FINISH এ যাও।

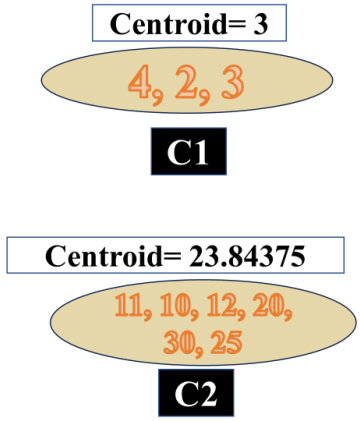
ধাপ-৭: এখন মডেলটি প্রস্তুত।

K-means ক্লাস্টারিং-এর সংখ্যাগত উদাহরণ:

K Means Clustering Example

- **Data Points: { 2,4, 10,12,3,20,30,11,25 }**
- **K = 2**
- Initial cluster centroids: **M1 = 4, M2 = 11 (Random)**

Centroid	Data Points	Distance to M1	Distance to M2	Cluster
(M1=4, M2=11)	2	2	9	C1
(M1=3, M2=11)	10	7	1	C2
(M1=3, M2=10.5)	12	9	1.5	C2
(M1=3, M2=10.75)	3	0	7.75	C1
(M1=3, M2=10.75)	20	17	9.25	C2
(M1=3, M2=15.375)	30	27	14.625	C2
(M1=3, M2=22.6875)	25	22	2.3125	C2



K-Means Clustering-এর সহজ ব্যবহার

• ব্যবসা ক্ষেত্রে: গ্রাহকদের কেনাকাটার আচরণের ভিত্তিতে আলাদা গ্রুপ তৈরি করা।

• শিক্ষাক্ষেত্রে: শিক্ষার্থীদের পারফরম্যান্স অনুযায়ী গ্রুপ করে উন্নয়নমূলক পরিকল্পনা নেওয়া।

• চিকিৎসাক্ষেত্রে: রোগীর রিপোর্ট বিশ্লেষণ করে রোগের ধরন বা ঝুঁকি চিহ্নিত করা।

• চিত্র প্রক্রিয়াকরণে: একই রঙ বা আকৃতির পিক্সেলগুলোকে একত্র করে ইমেজ কম্প্রেশন করা।

উপসংহার

Unsupervised Learning এমন একটি প্রযুক্তি যা ডেটা থেকে নিজেরাই সম্পর্ক ও প্যাটার্ন আবিষ্কার করে। Clustering তার অন্যতম ব্যবহারিক দিক, যা ব্যবসা, স্বাস্থ্য, শিক্ষা, এবং গবেষণায় বহুল ব্যবহৃত। K-Means Algorithm হলো এর সবচেয়ে জনপ্রিয় ও সহজবোধ্য রূপ। সঠিকভাবে ব্যবহার করলে Unsupervised Learning ভবিষ্যতে তথ্য বিশ্লেষণ ও সিদ্ধান্ত গ্রহণে গুরুত্বপূর্ণ ভূমিকা রাখবে।

অনুশীলনী (**Excersises**): [নম্বর- ২]

- Supervised learning কী? সংজ্ঞা দাও।
- Supervised learning-এর প্রধান দুইটি কাজের নাম কি?
- Supervised learning-এর জন্য training data-তে কী ধরনের লেবেল থাকে?
- Supervised learning-এর দুটি প্রধান algorithm-এর নাম দাও।
- Classification এবং Regression-এর মধ্যে পার্থক্য লিখ।
- Probabilistic classifier কী? সংক্ষেপে লিখ।
- Bayes theorem-এর মূল সূত্র লিখ।
- Conditional independence বলতে কী বোঝায়?
- Naive Bayes classifier কীভাবে কাজ করে? সংক্ষেপে ব্যাখ্যা কর।
- Naive Bayes classifier-এর একটি সাধারণ application লিখ।
- Sentiment classification task-এ Naive Bayes classifier কীভাবে ব্যবহার করা হয়?
- Add-one smoothing কী এবং কেন এটি Naive Bayes-এ ব্যবহার করা হয়?
- Naive Bayes classifier-এর সুবিধা দুটি লিখ।
- Naive Bayes classifier-এর অসুবিধা একটি লিখ।
- Decision Tree Learning-এর জন্য Entropy কী এবং কেন ব্যবহৃত হয়?
- Information Gain কী এবং এটি কিভাবে attribute-এর কার্যকারিতা মাপতে সাহায্য করে?
- ID3 Algorithm কী এবং এটি decision tree কীভাবে তৈরি করে সংক্ষেপে লিখ।
- Overfitting কী এবং কেন decision tree-তে হয়?
- Reduced Error Pruning কীভাবে decision tree-এর overfitting কমায়?
- Continuous-valued attributes কিভাবে discretize করা হয় information gain-based method ব্যবহার করে?

- Unsupervised Learning কী? সংক্ষেপে লিখো।
- Supervised Learning কী? সংক্ষেপে লিখো।
- Supervised Learning এবং Unsupervised Learning-এর মধ্যে প্রধান পার্থক্য কী?
- Clustering কী? সংক্ষেপে ব্যাখ্যা করো।
- কেন Clustering একটি Unsupervised Learning পদ্ধতি? সংক্ষেপে লিখো।
- K-Means Clustering কী? সংক্ষেপে লিখো।
- K-Means অ্যালগরিদমে ‘K’ মানের গুরুত্ব কী?
- K-Means অ্যালগরিদমে প্রতিটি ক্লাস্টারের সাথে কি যুক্ত থাকে?
- K-Means অ্যালগরিদমের মূল লক্ষ্য কী? সংক্ষেপে লিখো।
- K-Means অ্যালগরিদমে ডেটা পয়েন্টগুলিকে কীভাবে ক্লাস্টারে ভাগ করা হয়?
- Clustering-এর বাস্তব জীবনের একটি উদাহরণ দাও।
- Clustering-এর আরও একটি বাস্তব উদাহরণ লিখো।
- Classification কী? সংক্ষেপে লিখো।
- Clustering এবং Classification-এর মধ্যে পার্থক্য কী?
- Unsupervised Learning-এর আরেকটি উদাহরণ দাও।
- K-Means অ্যালগরিদমের ধাপগুলির মধ্যে প্রথম ধাপ কী?
- K-Means অ্যালগরিদমে পুনঃনিয়োগ (reassignment) কেন প্রয়োজন হয়?
- K-Means অ্যালগরিদম কবে শেষ হয়? সংক্ষেপে লিখো।
- K-Means ব্যবহার করে কোন ধরনের সমস্যা সমাধান করা যায়? সংক্ষেপে লিখো।
- Clustering এবং Unsupervised Learning এর প্রয়োজনীয়তা সংক্ষেপে লিখো।

অনুশীলনী: **[নম্বর- ৩]**

- Supervised Learning-এর প্রধান দুটি কাজ কি? উদাহরণসহ ব্যাখ্যা কর।
- Supervised Learning-এর জন্য training data-তে কী ধরনের লেবেল থাকে এবং কেন?
- কিছু জনপ্রিয় Supervised Learning Algorithm-এর নাম লিখ।
- Classification এবং Regression-এর মধ্যে পার্থক্য ব্যাখ্যা কর।
- Probabilistic classifier কী এবং এটি কীভাবে prediction করে?
- Bayes theorem-এর মূল সূত্র লিখ এবং সংক্ষেপে ব্যাখ্যা কর।
- Conditional independence কী এবং Naive Bayes-এ এটি কেন গুরুত্বপূর্ণ?
- Naive Bayes classifier কীভাবে কাজ করে? উদাহরণসহ ব্যাখ্যা কর।
- Naive Bayes classifier-এর সুবিধা দুটি এবং অসুবিধা দুটি লিখ।
- Sentiment classification task-এ Naive Bayes classifier কীভাবে ব্যবহার করা হয়?
- Add-one smoothing কী এবং এটি Naive Bayes classifier-এ কেন ব্যবহার করা হয়?
- Probabilistic classifier-এর ব্যবহার ক্ষেত্র উল্লেখ কর।
- Naive Bayes classifier-এর জন্য dataset তৈরি করার সময় কোন ধরনের preprocessing দরকার হয়?
- Decision Tree Learning-এর জন্য Entropy কী এবং এটি কীভাবে purity বা impurity মাপতে ব্যবহার হয়? সূত্রসহ ব্যাখ্যা কর।
- Information Gain কী এবং এটি কীভাবে attribute-এর কার্যকারিতা measure করতে সাহায্য করে? সূত্রসহ ব্যাখ্যা কর।
- ID3 Algorithm কী এবং এটি decision tree কীভাবে তৈরি করে? উদাহরণসহ ব্যাখ্যা কর।
- Overfitting কী এবং Decision Tree-তে এটি কেন ঘটে? উদাহরণসহ ব্যাখ্যা কর।
- Reduced Error Pruning কী এবং এটি Decision Tree-তে overfitting কমাতে কীভাবে সাহায্য করে?

- Continuous-valued attributes কিভাবে discretize করা হয় Information Gain-based method ব্যবহার করে? শুধুমাত্র binary split উল্লেখ করে উদাহরণসহ ব্যাখ্যা কর।
- Unsupervised Learning কী? উদাহরণসহ ব্যাখ্যা করো।
- Supervised Learning এবং Unsupervised Learning-এর মধ্যে পার্থক্য বিশদভাবে লিখো।
- Clustering কী এবং এটি কীভাবে কাজ করে? সংক্ষেপে উদাহরণসহ ব্যাখ্যা করো।
- কেন Clustering একটি Unsupervised Learning পদ্ধতি? উদাহরণসহ ব্যাখ্যা করো।
- K-Means Clustering Algorithm কী এবং এটি কীভাবে কাজ করে? সংক্ষেপে ব্যাখ্যা করো।
- K-Means অ্যালগরিদমের ধাপগুলো ব্যাখ্যা করো।
- Clustering এবং Classification-এর মধ্যে পার্থক্য উদাহরণসহ ব্যাখ্যা করো।
- Clustering-এর বাস্তব জীবনের তিনটি উদাহরণ উল্লেখ করো।
- K-Means ব্যবহার করে কোন ধরনের সমস্যা সমাধান করা যায়? সংক্ষেপে উদাহরণসহ লিখো।
- Unsupervised Learning এবং Clustering-এর ব্যবহার এবং গুরুত্ব সংক্ষেপে ব্যাখ্যা করো।

SEMESTER IV - UNIT 5

কৃত্রিম নিউরাল নেটওয়ার্ক **(ANN)-**এর জীববৈজ্ঞানিক প্রেরণা

কৃত্রিম নিউরাল নেটওয়ার্ক (Artificial Neural Networks বা ANNs)-এর গবেষণা মূলত জীববৈজ্ঞানিক শেখার পদ্ধতি থেকে অনুপ্রাণিত। দেখা গেছে, জীবিত প্রাণীর শেখার প্রক্রিয়া জটিল এবং পরস্পর সংযুক্ত নিউরনের (neuron) জালের ওপর নির্ভরশীল।

- কৃত্রিম নিউরাল নেটওয়ার্ক তৈরি হয় অসংখ্য ঘনভাবে সংযুক্ত সরল ইউনিট বা সেল দিয়ে, যেখানে প্রতিটি ইউনিট কিছু বাস্তব মানের ইনপুট (real-valued inputs) নেয় — যা অনেক সময় অন্য ইউনিটের আউটপুটও হতে পারে — এবং একটি বাস্তব মানের আউটপুট তৈরি করে, যা আবার অন্য অনেক ইউনিটের ইনপুট হিসেবে কাজ করে।

- মানব মস্তিষ্কে আনুমানিক 101110^{11}1011 (এক লক্ষ কোটি) নিউরন রয়েছে, এবং প্রতিটি নিউরন গড়ে প্রায় 10410^{4}104 (দশ হাজার) অন্যান্য নিউরনের সঙ্গে সংযুক্ত।

- নিউরনগুলির কার্যকলাপ অনেক সময় অন্য নিউরনের সংযোগের মাধ্যমে দমিত (inhibited) হয়।

কৃত্রিম নিউরাল নেটওয়ার্কের কাঠামো ও কার্যপদ্ধতি

জৈবিক গঠন থেকে অনুপ্রেরণা**:**
কৃত্রিম নিউরাল নেটওয়ার্ক মানুষের মস্তিষ্কের জৈবিক গঠনের ওপর ভিত্তি করে তৈরি, যেখানে মস্তিষ্ক একটি জটিল আন্তঃসংযুক্ত নিউরন নেটওয়ার্ক নিয়ে গঠিত।

Structure (গঠন**):**

ANN অনেকগুলো ঘনভাবে সংযুক্ত সাধারণ ইউনিট বা "নিউরন" নিয়ে গঠিত, যেগুলি ইনপুট গ্রহণ করে, তা প্রক্রিয়া করে এবং আউটপুট তৈরি করে।

Function (কার্য**):**

মানব মস্তিষ্কে নিউরনরা একে অপরকে বৈদ্যুতিক সংকেত পাঠিয়ে তথ্য প্রক্রিয়া করে। একইভাবে, ANN-এর প্রতিটি নিউরন ইনপুট গ্রহণ করে, তা প্রক্রিয়া করে, এবং ফলাফল বা আউটপুট তৈরি করে।

Learning (শেখার প্রক্রিয়া**):**

ANN-গুলি শেখার ক্ষমতার জন্য বিখ্যাত।
জীববৈজ্ঞানিক সিস্টেমে অভিযোজন (adaptation) একটি দীর্ঘমেয়াদি প্রক্রিয়া, যা বহু প্রজন্ম ধরে ঘটে — যেখানে

জীবেরা ধীরে ধীরে তাদের পরিবেশের সঙ্গে মানিয়ে নেয়।
কিন্তু ANN-এ শেখা (learning) একক সিস্টেমের মধ্যেই দ্রুত সম্পন্ন হয় — অর্থাৎ একটি নেটওয়ার্ক নিজে নিজেই প্রশিক্ষিত হতে পারে।

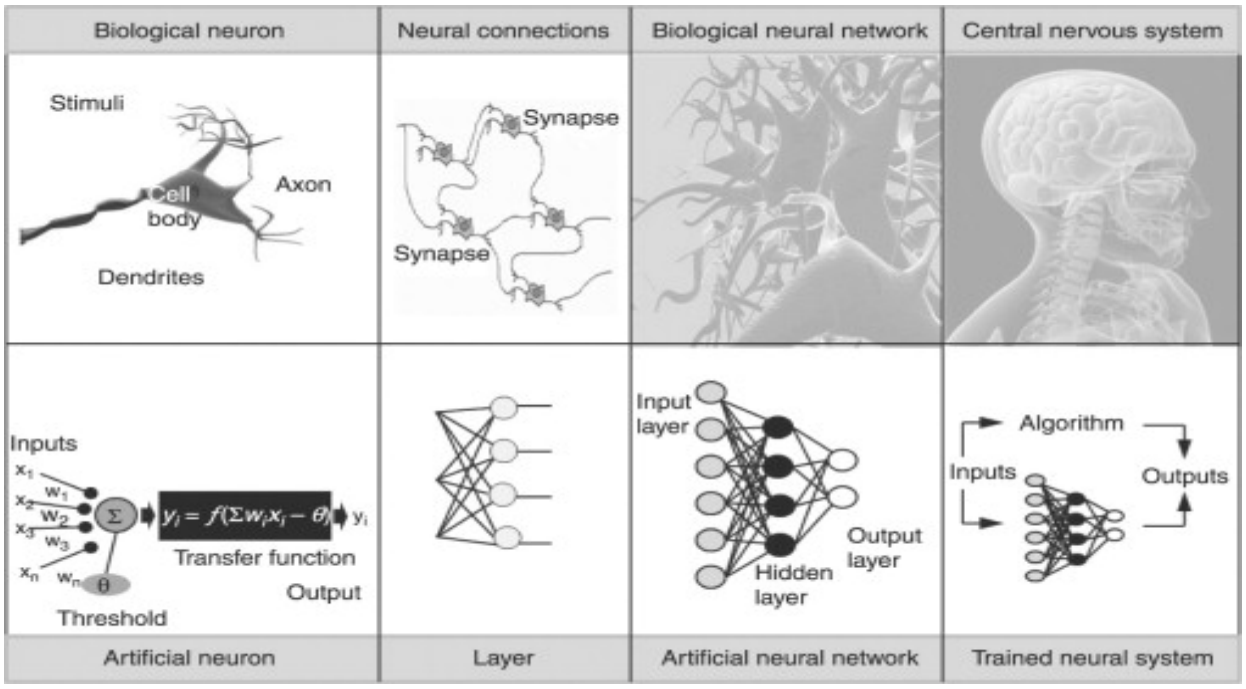
কৃত্রিম নিউরাল নেটওয়ার্কের অন্যান্য বৈশিষ্ট্য

- **Activation Function (**অ্যাক্টিভেশন ফাংশন**):**
  প্রতিটি নিউরনের একটি অ্যাক্টিভেশন ফাংশন থাকে, যা একটি গাণিতিক সমীকরণ — এটি নির্ধারণ করে কখন একটি নিউরন সক্রিয় হবে বা আউটপুট পাঠাবে।

- **Connections (**সংযোগ**):**
  প্রতিটি সংযোগের সঙ্গে একটি নির্দিষ্ট **weight (**ওজন**)** যুক্ত থাকে, যা নির্দেশ করে সেই ইনপুটের গুরুত্ব কতটুকু।

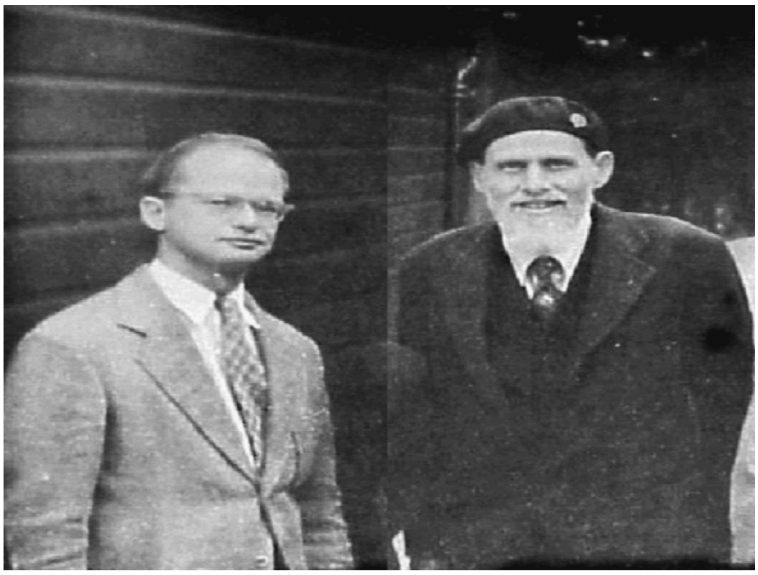
- **Threshold (**থ্রেশহোল্ড**):**
  নিউরন ইনপুটগুলির ওজনযুক্ত যোগফল (weighted sum) গণনা করে এবং সেটি একটি নির্দিষ্ট থ্রেশহোল্ডের সঙ্গে তুলনা করে সিদ্ধান্ত নেয় — আউটপুট তৈরি করবে কিনা।

কৃত্রিম নিউরাল নেটওয়ার্ক (ANN) মানব মস্তিষ্কের জৈবিক গঠন দ্বারা অনুপ্রাণিত , যা আন্তঃসংযুক্ত নিউরনের একটি জটিল নেটওয়ার্ক দ্বারা গঠিত.



কৃত্রিম নিউরাল নেটওয়ার্ক (ANN) গঠিত হয়েছে জীববৈজ্ঞানিক নিউরনের গঠন ও কার্যপ্রণালী থেকে অনুপ্রাণিত হয়ে। যেভাবে একটি জীববৈজ্ঞানিক নিউরন একটি নির্দিষ্ট থ্রেশহোল্ড-এ পৌঁছালে সক্রিয় হয় বা "ফায়ার" করে, সেই ধারণা থেকেই **McCulloch** এবং **Pitts (1943)** একটি সরল গাণিতিক নিউরন মডেল প্রস্তাব করেন।

একটি নিউরনের সরল গাণিতিক মডেল **(McCulloch** এবং **Pitts, 1943)**



McCulloch এবং Pitts মডেল, যা ১৯৪৩ সালে প্রবর্তিত হয়েছিল, ছিল নিউরনের প্রথমদিকের গাণিতিক মডেলগুলির একটি, যা মানব মস্তিষ্কের জীববৈজ্ঞানিক কার্যপ্রণালী থেকে অনুপ্রাণিত।
এই মডেলটি একটি সরল দ্বিমূলক **(binary)** নিউরন উপস্থাপন করে, যা একটি থ্রেশহোল্ড প্রক্রিয়া-এর ওপর ভিত্তি করে কাজ করে — অর্থাৎ নিউরনটি হয় "সক্রিয় হয়" (আউটপুট দেয় ১) অথবা "নিষ্ক্রিয় থাকে" (আউটপুট দেয় ০)।

McCulloch এবং **Pitts** নিউরন মডেলের প্রধান উপাদানসমূহ

1. ইনপুট ও ওজন **(Inputs and Weights):**

নিউরন একাধিক ইনপুট গ্রহণ করে, এবং প্রতিটি ইনপুটের সঙ্গে একটি নির্দিষ্ট ওজন **(weight)** যুক্ত থাকে।
এই ওজন নির্দেশ করে প্রতিটি ইনপুটের শক্তি বা গুরুত্ব কতটুকু, যা আউটপুট নির্ধারণে ভূমিকা রাখে।

2. ওজনযুক্ত যোগফল **(Weighted Sum):**

প্রতিটি ইনপুট x_i তার সংশ্লিষ্ট ওজন w_i এর সঙ্গে গুণ করা হয়। এরপর নিউরন সব ইনপুটের ওজনযুক্ত যোগফল হিসাব করে —

Weighted Sum= $\sum w_i x_i$

3. থ্রেশহোল্ড ফাংশন **(Threshold Function):**

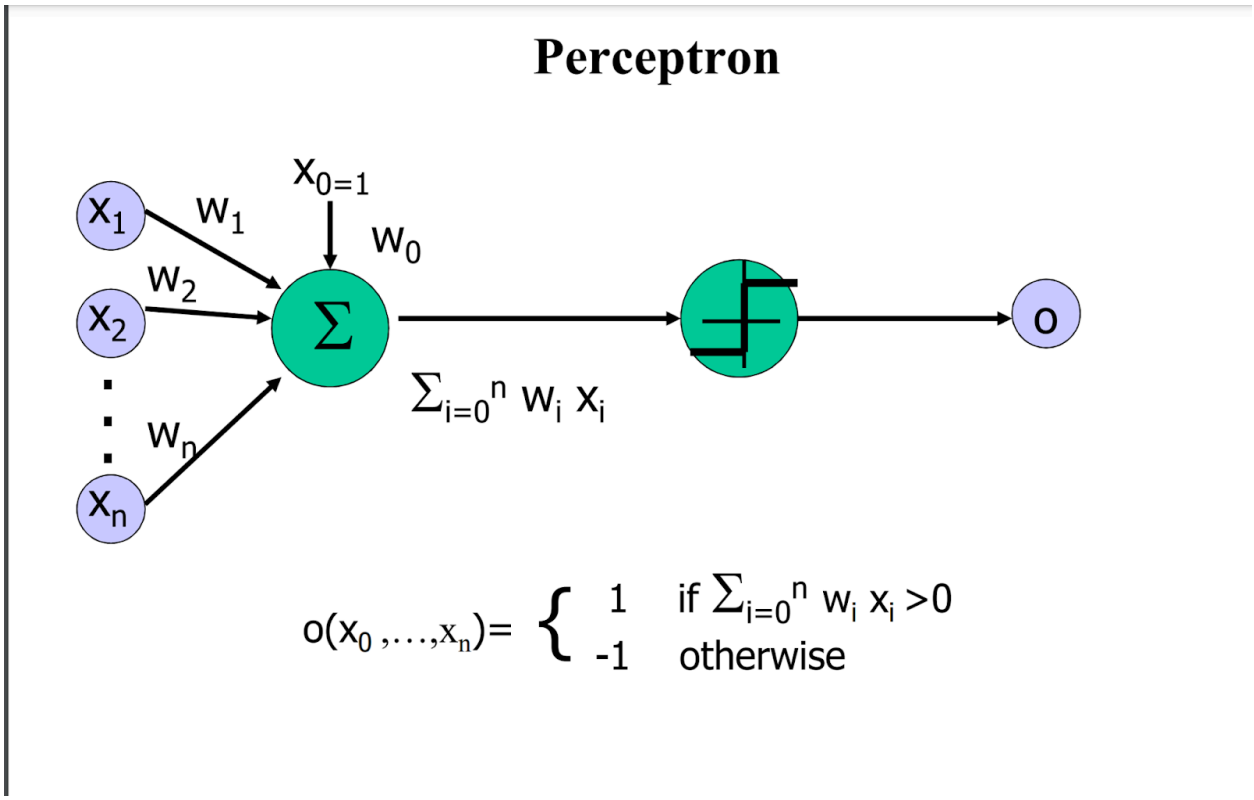
নিউরনের একটি নির্দিষ্ট থ্রেশহোল্ড মান **(θ)** থাকে।

- যদি ইনপুটগুলির ওজনযুক্ত যোগফল থ্রেশহোল্ড মানের সমান বা বেশি হয়, তবে নিউরন “সক্রিয় হয়” (fires) এবং আউটপুট দেয় ১।

- আর যদি ওজনযুক্ত যোগফল থ্রেশহোল্ডের কম হয়, তবে নিউরন সক্রিয় হয় না এবং আউটপুট দেয় ০।

Mathematically, this is represented as:

$$y = \begin{cases} 1 & \text{if } \sum_i w_i x_i \geq \theta \\ 0 & \text{if } \sum_i w_i x_i < \theta \end{cases}$$



ব্যাখ্যা ও গুরুত্ব **(Interpretation and Significance)**

দ্বিমূলক আউটপুট **(Binary Output):**

এই মডেলের দ্বিমূলক আউটপুট একটি সহজ অন-অফ সুইচ-এর মতো কাজ করে, যা মানব মস্তিষ্কে নিউরনের ফায়ারিং প্রক্রিয়া-র অনুরূপ।

যৌক্তিক ক্রিয়া **(Logical Operations):**

McCulloch এবং Pitts মডেলটি মৌলিক যৌক্তিক ফাংশন যেমন **AND**, **OR**, এবং **NOT** সম্পাদন করতে পারে। এটি নিউরাল নেটওয়ার্ক মডেলের ভিত্তি গঠনে একটি গুরুত্বপূর্ণ পদক্ষেপ ছিল।

সীমাবদ্ধতা **(Limitations):**

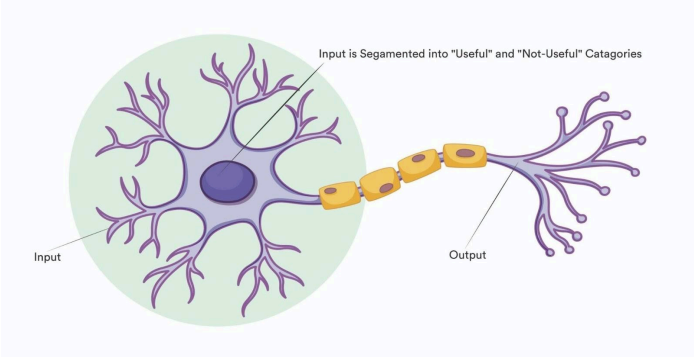
যদিও এই মডেলটি যুগান্তকারী ছিল, তবুও এর কিছু সীমাবদ্ধতা রয়েছে —
এটি শুধুমাত্র রৈখিকভাবে পৃথকযোগ্য **(linearly separable)** সমস্যা সমাধান করতে সক্ষম এবং জটিল, অরৈখিক **(nonlinear)** সিদ্ধান্ত সীমানা নির্ধারণ করতে পারে না।
এই সীমাবদ্ধতাই পরবর্তীতে আরও উন্নত মডেল তৈরির পথ খুলে দেয়, যেখানে ক্রমাগত অ্যাক্টিভেশন ফাংশন **(continuous activation functions)** এবং বহুস্তরীয় আর্কিটেকচার **(multi-layer architectures)** ব্যবহার করা হয়।

একটি পারসেপ্ট্রন হল একটি একক-স্তর নিউরাল নেটওয়ার্ক যা ইনপুটকে দুটি বিভাগে শ্রেণীবদ্ধ করে । এটি এক ধরনের লিনিয়ার ক্লাসিফায়ার যা ভবিষ্যদ্বাণী করার জন্য একটি বৈশিষ্ট্য ভেক্টরের সাথে ওজনের একটি সেটকে একত্রিত করতে একটি রৈখিক ভবিষ্যদ্বাণীকারী ফাংশন ব্যবহার করে।

এটা কি করে	ইনপুটগুলিকে দুটি বিভাগে শ্রেণীবদ্ধ করে
এটা কিভাবে কাজ করে	প্রশিক্ষণের ডেটার উপর ভিত্তি করে সিদ্ধান্তের সীমানার ওজন শিখে
এটা কি ফেরত	একটি একক বাইনারি আউটপুট, হয় 1 বা 0।
সুবিধা	বাস্তবায়ন, প্রশিক্ষণ, এবং বুঝতে সহজ
সীমাবদ্ধতা	প্রশিক্ষণের অসুবিধার কারণে গভীর শিক্ষায় সাধারণত ব্যবহৃত হয় না

অ্যাক্টিভেশন ফাংশনের ধারণা: থ্রেশহোল্ড ফাংশন এবং সিগময়েড ফাংশন

সংজ্ঞা: কৃত্রিম নিউরাল নেটওয়ার্কে (Artificial Neural Network) প্রতিটি নিউরন তার ইনপুটগুলির একটি ওজনযুক্ত যোগফল **(weighted sum)** গণনা করে এবং প্রাপ্ত মানটি একটি নির্দিষ্ট ফাংশনের মাধ্যমে প্রেরণ করে, যাকে অ্যাক্টিভেশন ফাংশন **(activation function)** বলা হয়।



মানুষের মস্তিষ্ক ও কৃত্রিম নিউরাল নেটওয়ার্কের তুলনা

মানুষের ক্ষেত্রে, আমাদের মস্তিষ্ক বাইরের পরিবেশ থেকে ইনপুট গ্রহণ করে, সেই ইনপুট নিউরনের মাধ্যমে প্রক্রিয়া করে এবং নিউরনের টেইল (axon) সক্রিয় করে প্রয়োজনীয় সিদ্ধান্ত তৈরি করে।
একইভাবে, কৃত্রিম নিউরাল নেটওয়ার্কে আমরা ইনপুট দিই যেমন — ছবি (images), শব্দ (sounds), সংখ্যা (numbers) ইত্যাদি।
এরপর কৃত্রিম নিউরনে সেই ইনপুট প্রক্রিয়া করা হয়, এবং একটি নির্দিষ্ট অ্যালগরিদম সঠিক আউটপুট নিউরন স্তর **(final neuron layer)** সক্রিয় করে ফলাফল তৈরি করে।

কেন অ্যাক্টিভেশন ফাংশনের প্রয়োজন?

একটি অ্যাক্টিভেশন ফাংশন নির্ধারণ করে যে কোনো নিউরন সক্রিয় **(activated)** হবে কিনা।
অর্থাৎ, এটি কিছু সরল গাণিতিক প্রক্রিয়ার মাধ্যমে নির্ধারণ করে যে নিউরনে প্রবেশ করা ইনপুটটি নেটওয়ার্কের ভবিষ্যদ্বাণী (prediction) প্রক্রিয়ায় প্রাসঙ্গিক **(relevant)** নাকি অপ্রাসঙ্গিক **(irrelevant)**।

অ্যাক্টিভেশন ফাংশনের মূল উদ্দেশ্য হলো —
একটি কৃত্রিম নিউরাল নেটওয়ার্কে অরৈখিকতা **(non-linearity)** যুক্ত করা এবং একটি স্তরে দেওয়া ইনপুটগুলির একটি সংগ্রহ থেকে আউটপুট তৈরি করা।

অ্যাক্টিভেশন ফাংশনের ধরন

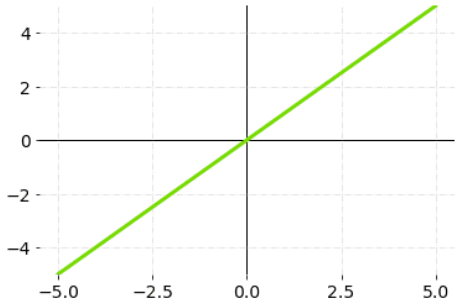
অ্যাক্টিভেশন ফাংশনকে প্রধানত তিনটি ভাগে ভাগ করা যায় —

1. **Linear Activation Function (**রৈখিক অ্যাক্টিভেশন ফাংশন**)**
2. **Binary Step Function (**দ্বিমূলক ধাপ ফাংশন**)**
3. **Non-linear Activation Functions (**অরৈখিক অ্যাক্টিভেশন ফাংশন**)**

Linear Activation Function (রৈখিক অ্যাক্টিভেশন ফাংশন**)**

রৈখিক অ্যাক্টিভেশন ফাংশনকে অনেক সময় আইডেন্টিটি অ্যাক্টিভেশন ফাংশন **(identity activation function)** বলা হয়, কারণ এটি ইনপুটের সঙ্গে সমানুপাতিক **(proportional)**।

- এর মানের পরিসর হলো: (−∞,∞)
- এই ফাংশন ইনপুটগুলির ওজনযুক্ত যোগফল (weighted total) গণনা করে এবং সরাসরি সেই ফলাফলকে আউটপুট হিসেবে প্রদান করে। অর্থাৎ, f(x)=x. এটি ইনপুট ও আউটপুটের মধ্যে সরল রৈখিক সম্পর্ক তৈরি করে।



সুবিধা ও অসুবিধা **(Pros and Cons)**

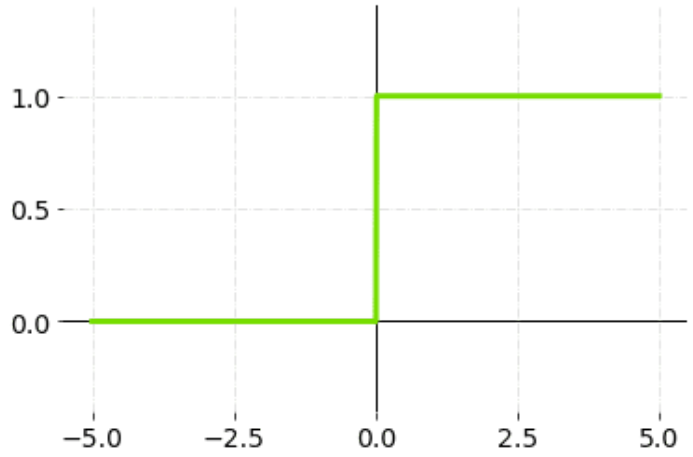
সুবিধা**:**
রৈখিক অ্যাক্টিভেশন ফাংশন (Linear Activation Function) দ্বিমূলক নয়, কারণ এটি কেবলমাত্র একটি মানের পরিসরের মধ্যে সক্রিয়তা (activation range) প্রদান করে।
আমরা একাধিক নিউরনকে একত্রে সংযুক্ত করতে পারি, এবং যদি একাধিক অ্যাক্টিভেশন ঘটে, তবে তাদের মধ্যে সর্বোচ্চ মান (**max**) বা **softmax** হিসাব করা সম্ভব।

অসুবিধা**:**
এই অ্যাক্টিভেশন ফাংশনের ডেরিভেটিভ **(derivative)** একটি ধ্রুবক (constant)।
অর্থাৎ, গ্রেডিয়েন্ট (gradient) ইনপুট xxx-এর সঙ্গে সম্পর্কিত নয়।
ফলে, এটি শেখার প্রক্রিয়ায় পরিবর্তনের প্রতি সাড়া দিতে সীমিত সক্ষমতা প্রদর্শন করে।

Binary Step Activation Function (দ্বিমূলক ধাপ অ্যাক্টিভেশন ফাংশন**)**

এই ফাংশনে একটি নির্দিষ্ট থ্রেশহোল্ড মান **(threshold value)** নির্ধারণ করে দেয় যে একটি নিউরন সক্রিয় হবে না নিষ্ক্রিয় থাকবে।

- অ্যাক্টিভেশন ফাংশন ইনপুট মানটির সঙ্গে থ্রেশহোল্ড মানের তুলনা করে।
- যদি ইনপুট মান থ্রেশহোল্ড মানের চেয়ে বেশি হয়, তবে নিউরন সক্রিয় **(activated)** হয়।
- যদি ইনপুট মান থ্রেশহোল্ড মানের চেয়ে কম হয়, তবে নিউরন নিষ্ক্রিয় **(deactivated)** থাকে — অর্থাৎ তার আউটপুট পরবর্তী স্তর বা হিডেন লেয়ার **(hidden layer)**-এ পাঠানো হয় না।



$$f(x) = \begin{cases} 0 & for\ x < 0 \\ 1 & for\ x \geqslant 0 \end{cases}$$

সুবিধা ও অসুবিধা **(Pros and Cons)**

অসুবিধা**:**

- এটি একাধিক মানের আউটপুট **(multi-value outputs)** প্রদান করতে পারে না — অর্থাৎ এটি বহু-শ্রেণির শ্রেণিবিন্যাস **(multi-class classification)** সমস্যার জন্য ব্যবহার করা যায় না।

- স্টেপ ফাংশন **(step function)**-এর গ্রেডিয়েন্ট **(gradient)** সর্বত্র শূন্য **(zero)** থাকে, যার ফলে ব্যাকপ্রোপাগেশন **(backpropagation)** প্রক্রিয়াটি কঠিন হয়ে পড়ে, কারণ শেখার জন্য প্রয়োজনীয় ত্রুটির (error) পরিবর্তন গণনা করা যায় না।

অরৈখিক অ্যাক্টিভেশন ফাংশন **(Non-linear Activation Functions)**

অরৈখিক অ্যাক্টিভেশন ফাংশনগুলোই কৃত্রিম নিউরাল নেটওয়ার্কে সবচেয়ে বেশি ব্যবহৃত হয়।
এগুলি নেটওয়ার্ককে বিভিন্ন ধরণের ডেটার সঙ্গে মানিয়ে নিতে সাহায্য করে এবং আউটপুটগুলির মধ্যে পার্থক্য করতে সক্ষম করে।

অরৈখিক অ্যাক্টিভেশন ফাংশন ব্যবহারের ফলে নিউরনের একাধিক স্তর (multiple layers) একসঙ্গে স্ট্যাক **(stack)** করা সম্ভব হয়, কারণ আউটপুটটি এখন ইনপুটের একটি অরৈখিক সমন্বয় **(non-linear combination)** হিসেবে কাজ করে, যা বহু স্তরের মধ্য দিয়ে প্রবাহিত হয়।

ফলস্বরূপ, নিউরাল নেটওয়ার্কে যেকোনো জটিল আউটপুটকে একটি কার্যকর **(functional)** গাণিতিক সমীকরণ দ্বারা প্রকাশ করা যায়।

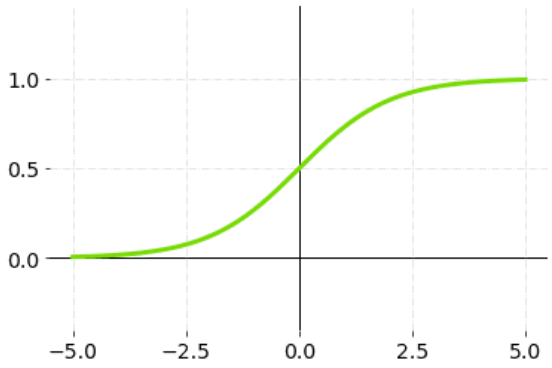
এই অরৈখিক অ্যাক্টিভেশন ফাংশনগুলোকে সাধারণত তাদের পরিসর **(range)** এবং বক্রতা **(curve)**-এর উপর ভিত্তি করে ভাগ করা হয়।
নিচে কৃত্রিম নিউরাল নেটওয়ার্কে ব্যবহৃত প্রধান কিছু অরৈখিক অ্যাক্টিভেশন ফাংশন তুলে ধরা হলো।

Sigmoid Function (সিগময়েড ফাংশন)

Sigmoid ফাংশন একটি ইনপুট সংখ্যাকে গ্রহণ করে এবং 0 ও 1-এর মধ্যে একটি মান প্রদান করে।
এটি ব্যবহার করা সহজ এবং অ্যাক্টিভেশন ফাংশনের সব কাঙ্ক্ষিত বৈশিষ্ট্য বহন করে — যেমন:

- অরৈখিকতা **(Non-linearity)**
- ধারাবাহিক অন্তরকতা **(Continuous differentiability)**
- মনোটোনিসিটি **(Monotonicity)**
- নির্দিষ্ট আউটপুট পরিসর **(Fixed output range)**

এই ফাংশনটি মূলত দ্বিমূলক শ্রেণিবিন্যাস **(Binary Classification)** সমস্যায় ব্যবহৃত হয়।
Sigmoid ফাংশন একটি নির্দিষ্ট শ্রেণি উপস্থিত থাকার সম্ভাবনা **(probability)** নির্ধারণ করতে সাহায্য করে।



গাণিতিকভাবে এটি প্রকাশ করা যায়ঃ

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

এখানে,

- x = ইনপুট মান
- f(x)= 0 থেকে 1-এর মধ্যে আউটপুট মান

সুবিধা ও অসুবিধা **(Pros and Cons)**

সুবিধা**:**

- এটি স্বভাবতই অরৈখিক **(non-linear)**, অর্থাৎ এই ফাংশনের সংমিশ্রণও অরৈখিক থাকে, যা এটিকে বাইনারি স্টেপ ফাংশন থেকে আলাদা করে।
- এটি একটি ধারাবাহিক গ্রেডিয়েন্ট **(smooth gradient)** প্রদান করে, যা শেখার প্রক্রিয়াকে (training) আরও কার্যকর করে তোলে।
- এটি শ্রেণিবিন্যাস **(classification)** ধরণের সমস্যার জন্য বিশেষভাবে উপযোগী।
- এর আউটপুট সর্বদা **(0, 1)** পরিসরের মধ্যে থাকে, যেখানে লিনিয়ার অ্যাক্টিভেশন ফাংশনের আউটপুট পরিসর **(-∞, ∞)**।
  ফলে আমাদের অ্যাক্টিভেশনগুলির জন্য একটি নির্দিষ্ট সীমা নির্ধারণ করা যায়।

অসুবিধা**:**

- সিগময়েড ফাংশনের একটি প্রধান সমস্যা হলো **“Vanishing Gradient Problem”**। অর্থাৎ, ইনপুট খুব বড় বা ছোট হলে গ্রেডিয়েন্ট প্রায় শূন্যের কাছাকাছি চলে যায়, ফলে নেটওয়ার্ক শেখা বন্ধ করে দেয় বা খুব ধীরগতিতে শেখে।
- সিগময়েড আউটপুট শূন্য-কেন্দ্রিক **(zero-centered)** নয়, যার ফলে গ্রেডিয়েন্ট আপডেট **(gradient updates)** একাধিক ভিন্ন দিকে সরে যেতে পারে, যা অপ্টিমাইজেশনকে কঠিন করে তোলে।
- এই কারণে অনেক সময় নেটওয়ার্ক শিখতে অস্বীকার করে অথবা অত্যন্ত ধীরে শেখে।

তুলনামূলক বৈশিষ্ট্য **(Feature Comparison)**

বৈশিষ্ট্য **(Feature)**	থ্রেশহোল্ড ফাংশন **(Threshold Function)**	সিগময়েড ফাংশন **(Sigmoid Function)**
আউটপুট পরিসর **(Output Range)**	0 বা 1	0 থেকে 1

ধারাবাহিকতা **(Continuity)**	বিচ্ছিন্ন (Discrete)	ধারাবাহিক (Continuous)
অন্তরকযোগ্য **(Differentiable)**	না (No)	হ্যাঁ (Yes)
উপযুক্ততা **(Suitability)**	বাইনারি ক্লাসিফিকেশন	বাইনারি ক্লাসিফিকেশন, সম্ভাবনামূলক আউটপুট
সাধারণ ব্যবহার **(Common Usage)**	পারসেপ্ট্রন, এক-স্তর নেটওয়ার্ক	বহুস্তর নিউরাল নেটওয়ার্ক, লজিস্টিক রিগ্রেশন

বাস্তব ব্যবহার **(Practical Usage)**

নিউরাল নেটওয়ার্কে সিগময়েড ফাংশন সাধারণত ব্যাকপ্রোপাগেশন **(backpropagation)**-এর সঙ্গে প্রশিক্ষণের জন্য ব্যবহৃত হয়, কারণ এটি ত্রুটি সংশোধনের **(error correction)** জন্য গ্রেডিয়েন্ট প্রদান করে।
অন্যদিকে, থ্রেশহোল্ড ফাংশন ব্যবহার করা হয় সহজ মডেলে বা যেখানে আউটপুটকে কঠোর বাইনারি মানে (strict binary terms) প্রকাশ করতে হয়।

সিগময়েড এবং অন্যান্য অ্যাক্টিভেশন ফাংশনের ব্যবহার নিউরাল নেটওয়ার্ককে জটিল ও অরৈখিক ফাংশনগুলির আনুমানিক মান নির্ণয় করতে সক্ষম করে, ফলে এটি আরও উন্নত কাজ সম্পাদন করতে পারে।

Perceptron as a Linear Classifier, Perceptron Training Rule

পারসেপ্ট্রনের মৌলিক উপাদানসমূহ **(Basic Components of Perceptron)**

পারসেপ্ট্রন **(Perceptron)** হলো কৃত্রিম নিউরাল নেটওয়ার্কের একটি প্রাথমিক ধরন, যা মেশিন লার্নিং-এর **(Machine Learning)** একটি মৌলিক ধারণা।

এর প্রধান উপাদানগুলি হলো —

1. ইনপুট স্তর **(Input Layer):**
   এই স্তরে এক বা একাধিক ইনপুট নিউরন থাকে, যা বাইরের বিশ্ব থেকে বা অন্য স্তর থেকে ইনপুট সংকেত গ্রহণ করে।

2. ওজন **(Weights):**
   প্রতিটি ইনপুট নিউরনের সঙ্গে একটি ওজন যুক্ত থাকে, যা ইনপুট এবং আউটপুট নিউরনের সংযোগের শক্তি বা গুরুত্ব নির্দেশ করে।

3. বায়াস **(Bias):**
   ইনপুট স্তরের সঙ্গে একটি বায়াস যোগ করা হয়, যা পারসেপ্ট্রনকে জটিল প্যাটার্ন বা ডেটা মডেল করতে আরও নমনীয়তা প্রদান করে।

4. অ্যাক্টিভেশন ফাংশন **(Activation Function):**
   ইনপুটগুলির ওজনযুক্ত যোগফল এবং বায়াসের ওপর ভিত্তি করে আউটপুট নির্ধারণ করে।
   সাধারণ অ্যাক্টিভেশন ফাংশনের মধ্যে রয়েছে — স্টেপ ফাংশন**,** সিগময়েড ফাংশন**,** এবং **ReLU** ফাংশন।

5. আউটপুট **(Output):**
   পারসেপ্ট্রনের আউটপুট সাধারণত একটি দ্বিমূলক মান **(0** বা **1)** হয়, যা ইনপুট ডেটা কোন শ্রেণিতে পড়ছে তা নির্দেশ করে।

6. প্রশিক্ষণ অ্যালগরিদম **(Training Algorithm):**
   পারসেপ্ট্রন সাধারণত সুপারভাইজড লার্নিং অ্যালগরিদম (যেমন perceptron learning algorithm বা backpropagation) দ্বারা প্রশিক্ষিত হয়।
   প্রশিক্ষণের সময়, ওজন এবং বায়াস এমনভাবে সমন্বয় করা হয় যাতে ভবিষ্যদ্বাণীকৃত আউটপুট এবং প্রকৃত আউটপুটের মধ্যে ত্রুটি (error) ন্যূনতম হয়।

সামগ্রিকভাবে, পারসেপ্ট্রন একটি সরল কিন্তু শক্তিশালী অ্যালগরিদম, যা বাইনারি শ্রেণিবিন্যাসের কাজ করতে পারে এবং আধুনিক ডিপ লার্নিং **(Deep Learning)**-এর ভিত্তি গঠনে গুরুত্বপূর্ণ ভূমিকা রেখেছে।

পারসেপ্ট্রনের ধরন **(Types of Perceptron):**

1. সিঙ্গল লেয়ার পারসেপ্ট্রন **(Single-Layer Perceptron):**
   এটি শুধুমাত্র রৈখিকভাবে পৃথকযোগ্য **(linearly separable)** প্যাটার্ন শিখতে পারে।

2. মাল্টিলেয়ার পারসেপ্ট্রন **(Multilayer Perceptron):**
   এতে দুটি বা তার বেশি স্তর থাকে এবং এর প্রসেসিং ক্ষমতা **(processing power)** অনেক বেশি।

পারসেপ্ট্রন অ্যালগরিদম ইনপুট সিগন্যালগুলির জন্য ওজন **(weights)** শেখে, যাতে একটি রৈখিক সিদ্ধান্ত সীমানা **(linear decision boundary)** নির্ধারণ করা যায়।

নোট**:** সুপারভাইজড লার্নিং **(Supervised Learning)** হলো এমন একটি মেশিন লার্নিং প্রক্রিয়া যেখানে লেবেলযুক্ত (labeled) ট্রেনিং ডেটা ব্যবহার করে মডেল শেখানো হয়, যাতে ভবিষ্যতের বা অজানা ডেটার আউটপুট ভবিষ্যদ্বাণী করা যায়।

Perceptron in Machine Learning (মেশিন লার্নিং**-**এ পারসেপ্ট্রন**)**

পারসেপ্ট্রন হলো কৃত্রিম বুদ্ধিমত্তা (AI) ও মেশিন লার্নিং-এর (ML) সবচেয়ে প্রচলিত শব্দগুলির একটি।
এটি ডিপ লার্নিং ও লজিক গেটসের **(logic gates)** বাস্তবায়নের প্রথম ধাপ হিসেবে ব্যবহৃত হয়।

১৯৫৮ সালে ফ্র্যাঙ্ক রোজেনব্ল্যাট **(Frank Rosenblatt)** পারসেপ্ট্রন আবিষ্কার করেন।
এর মূল উদ্দেশ্য ছিল এমন একটি মেশিন তৈরি করা, যা মানুষের মস্তিষ্কের মতো শেখা ও প্যাটার্ন চিনতে **(pattern recognition)** সক্ষম হবে।

প্রাথমিকভাবে পারসেপ্ট্রন ব্যবহার করা হয়েছিল হাতে লেখা অক্ষর সনাক্তকরণ **(handwritten character recognition)**-এর মতো সহজ সমস্যার সমাধানে।
তবে এটি অরৈখিকভাবে পৃথকযোগ্য **(non-linearly separable)** ডেটা পরিচালনা করতে পারত না, যা এর একটি বড় সীমাবদ্ধতা ছিল।
ফলে ১৯৬০–৭০ দশকে এর গবেষণায় স্থবিরতা দেখা দেয়।

কিন্তু ১৯৮০-এর দশকে ব্যাকপ্রোপাগেশন **(Backpropagation)** অ্যালগরিদমের বিকাশের ফলে পারসেপ্ট্রন ও নিউরাল নেটওয়ার্ক নিয়ে গবেষণার পুনর্জাগরণ ঘটে।
আজকের দিনে পারসেপ্ট্রনকে সবচেয়ে সরল কৃত্রিম নিউরাল নেটওয়ার্ক **(ANN)** হিসেবে গণ্য করা হয় এবং এটি চিত্র সনাক্তকরণ **(image recognition),** ভাষা প্রক্রিয়াকরণ **(NLP),** ও বক্তৃতা সনাক্তকরণ **(speech recognition)**-এ ব্যবহৃত হয়।

Perceptron Model in Machine Learning (পারসেপ্ট্রন মডেল কীভাবে কাজ করে)

পারসেপ্ট্রন একটি সুপারভাইজড লার্নিং অ্যালগরিদম, যা বাইনারি ক্লাসিফিকেশন টাস্ক সম্পাদন করে।
এটি চারটি প্রধান উপাদান নিয়ে গঠিত —

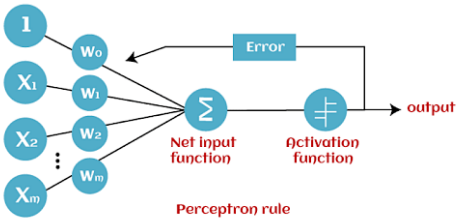
1. ইনপুট মান **(Input Values)**
2. ওজন ও বায়াস **(Weights and Bias)**
3. নেট সাম **(Net Sum)**
4. অ্যাক্টিভেশন ফাংশন **(Activation Function)**

কাজের প্রক্রিয়া**:**
পারসেপ্ট্রন মডেল ইনপুট মানগুলিকে তাদের সংশ্লিষ্ট ওজনের সঙ্গে গুণ করে এবং সবগুলিকে যোগ করে একটি ওজনযুক্ত যোগফল (weighted sum) তৈরি করে।
এরপর এই যোগফলটি একটি অ্যাক্টিভেশন ফাংশন ‘f’-এর মাধ্যমে প্রেরণ করা হয়, যা আউটপুট তৈরি করে। এই ফাংশনটি সাধারণত একটি স্টেপ ফাংশন হিসেবে কাজ করে।

গাণিতিকভাবে প্রকাশ করা যায়ঃ $y=f(\sum wixi+b)$, এখানে:

- xi = ইনপুট মান
- wi= ওজন
- b = বায়াস
- f = অ্যাক্টিভেশন ফাংশন
- y = আউটপুট

এইভাবে, পারসেপ্ট্রন একটি রৈখিক ক্লাসিফায়ার **(linear classifier)** হিসেবে কাজ করে, যা ইনপুট ডেটার ভিত্তিতে সিদ্ধান্ত নেয়।



এই ধাপ ফাংশন বা অ্যাক্টিভেশন ফাংশনটি আউটপুটকে (0,1) বা (-1,1)-এর মধ্যে মানচিত্র করতে অত্যন্ত গুরুত্বপূর্ণ। লক্ষ্য রাখতে হবে যে ইনপুটের ওজন একটি নোডের শক্তি নির্দেশ করে। একইভাবে, ইনপুট মান অ্যাক্টিভেশন ফাংশনের বক্ররেখাকে উপরে বা নিচে সরানোর ক্ষমতা প্রদান করে।

ধাপ ১: সমস্ত ইনপুট মানকে তাদের সংশ্লিষ্ট ওজনের সাথে গুণ করে যোগ করুন, যাতে ওজনযুক্ত যোগফল (weighted sum) নির্ণয় করা যায়। এর গাণিতিক প্রকাশটি নিচে দেওয়া হলো:
∑wi*xi* = *x1*w1 + x2*w2* + *x3*w3 + …… + x4*w4

এই ওজনযুক্ত যোগফলে একটি পদ যোগ করা হয়, যাকে বায়াস **(bias)** বলা হয়, যা মডেলের কর্মক্ষমতা উন্নত করতে সাহায্য করে।

ধাপ ২: উপরের ওজনযুক্ত যোগফলের উপর একটি অ্যাক্টিভেশন ফাংশন প্রয়োগ করা হয়, যা আমাদের একটি আউটপুট দেয় — হয় বাইনারি ফর্মে, নয়তো একটি ধারাবাহিক মান (continuous value) আকারে।

আমরা ইতিমধ্যে প্রারম্ভিক অংশে পারসেপট্রন (Perceptron) মডেলের ধরণগুলি নিয়ে আলোচনা করেছি। এখানে আমরা সেটির একটি গভীর বিশ্লেষণ করবো:

সিঙ্গেল লেয়ার পারসেপট্রন মডেল **(Single Layer Perceptron Model):**

এটি কৃত্রিম স্নায়ুবিশিষ্ট নেটওয়ার্কের (ANN) একটি সহজতম ধরন, যা একটি ফিড-ফরোয়ার্ড নেটওয়ার্ক নিয়ে গঠিত এবং এতে একটি থ্রেশহোল্ড ট্রান্সফার ফাংশন অন্তর্ভুক্ত থাকে।
এর মূল উদ্দেশ্য হলো রৈখিকভাবে বিভাজ্য **(linearly separable)** বস্তুগুলিকে বাইনারি আউটপুটের মাধ্যমে বিশ্লেষণ করা।
একটি সিঙ্গেল লেয়ার পারসেপট্রন শুধুমাত্র রৈখিকভাবে বিভাজ্য প্যাটার্ন শিখতে পারে।

মাল্টি-লেয়ার পারসেপট্রন মডেল **(Multi-Layer Perceptron Model):**

এটি মূলত সিঙ্গেল লেয়ার পারসেপট্রনের মতো, তবে এতে অতিরিক্ত হিডেন লেয়ার **(hidden layers)** থাকে।

ফরোয়ার্ড স্টেজ**:** ইনপুট লেয়ার থেকে শুরু করে আউটপুট লেয়ার পর্যন্ত অ্যাক্টিভেশন ফাংশন কাজ করে।

ব্যাকওয়ার্ড স্টেজ**:** এই পর্যায়ে ওজন এবং বায়াস মান পরিবর্তন করা হয় মডেলের প্রয়োজন অনুসারে। এটি আউটপুট স্তর থেকে প্রাপ্ত আসল ফলাফল এবং প্রত্যাশিত ফলাফলের মধ্যে থাকা ত্রুটি **(error)** কমায়।

একটি মাল্টি-লেয়ার পারসেপট্রনের প্রক্রিয়াকরণ ক্ষমতা বেশি এবং এটি রৈখিক ও অরৈখিক **(non-linear)** উভয় ধরনের প্যাটার্ন প্রক্রিয়া করতে পারে। এছাড়াও এটি AND, OR, XOR, XNOR, এবং NOR-এর মতো লজিক গেট বাস্তবায়ন করতে পারে।

সুবিধাসমূহ **(Advantages):**

- মাল্টি-লেয়ার পারসেপট্রন জটিল অরৈখিক সমস্যাগুলি সমাধান করতে পারে।
- ছোট এবং বড় উভয় ধরনের ইনপুট ডেটার সাথে ভালো কাজ করে।
- প্রশিক্ষণের পরে দ্রুত ভবিষ্যদ্বাণী করতে সহায়তা করে।
- ছোট ও বড় ডেটায় একই ধরনের সঠিকতা (accuracy) বজায় রাখতে সাহায্য করে।

অসুবিধাসমূহ **(Disadvantages):**

- মাল্টি-লেয়ার পারসেপট্রন মডেলে গণনাগুলি সময়সাপেক্ষ ও জটিল।
- নির্ভরশীল চলক (dependent variable) কতটা স্বাধীন চলকের (independent variable) উপর প্রভাব ফেলে, তা নির্ধারণ করা কঠিন।
- মডেলের কার্যকারিতা প্রশিক্ষণের গুণমানের উপর নির্ভরশীল।

পারসেপট্রন মডেলের বৈশিষ্ট্য **(Characteristics of the Perceptron Model):**

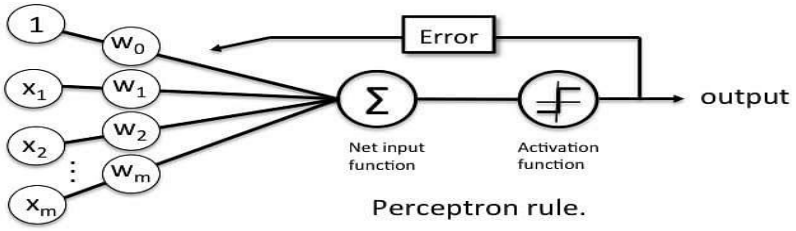
- এটি একটি মেশিন লার্নিং অ্যালগরিদম, যা সুপারভাইজড লার্নিং ব্যবহার করে বাইনারি ক্লাসিফায়ার শেখায়।
- পারসেপট্রনে ওজন সহগ (weight coefficient) স্বয়ংক্রিয়ভাবে শেখা হয়।
- প্রাথমিকভাবে ওজনগুলি ইনপুট বৈশিষ্ট্যের সাথে গুণ করা হয়, তারপর নির্ধারণ করা হয় নিউরন সক্রিয় হবে কিনা।
- অ্যাক্টিভেশন ফাংশনটি একটি স্টেপ রুল **(step rule)** প্রয়োগ করে, যা যাচাই করে ফাংশনের মান শূন্যের বেশি কি না।
- একটি রৈখিক সিদ্ধান্ত সীমা **(linear decision boundary)** আঁকা হয়, যা দুটি রৈখিকভাবে বিভাজ্য শ্রেণী (+1 এবং -1)-কে আলাদা করে।
- যদি সমস্ত ইনপুট মানের যোগফল থ্রেশহোল্ড মানের চেয়ে বেশি হয়, তাহলে আউটপুট সংকেত তৈরি হবে; অন্যথায় কোনো আউটপুট থাকবে না।

পারসেপট্রন মডেলের সীমাবদ্ধতা **(Limitation of Perceptron Model):**

- কঠিন প্রান্ত স্থানান্তর ফাংশনের (hard-edge transfer function) কারণে পারসেপট্রনের আউটপুট কেবলমাত্র বাইনারি **(0** বা **1)** হতে পারে।
- এটি শুধুমাত্র রৈখিকভাবে বিভাজ্য ইনপুট ভেক্টর শ্রেণিবদ্ধ করতে ব্যবহার করা যায়। ইনপুট ভেক্টর যদি অরৈখিক হয়, তাহলে সঠিকভাবে শ্রেণিবদ্ধ করা কঠিন।

পারসেপট্রন লার্নিং রুল **(Perceptron Learning Rule):**

পারসেপট্রন লার্নিং রুল বলে যে অ্যালগরিদমটি স্বয়ংক্রিয়ভাবে সর্বোত্তম ওজন সহগ **(optimal weight coefficients)** শিখে নেবে। ইনপুট বৈশিষ্ট্যগুলি এই শেখা ওজনের সাথে গুণ করা হয়, যাতে নির্ধারণ করা যায় নিউরন সক্রিয় হবে কিনা।



Perceptron rule.

পারসেপট্রন (Perceptron) একাধিক ইনপুট সংকেত গ্রহণ করে, এবং যদি ইনপুট সংকেতগুলির যোগফল একটি নির্দিষ্ট থ্রেশহোল্ড **(threshold)**-এর চেয়ে বেশি হয়, তবে এটি একটি আউটপুট সংকেত প্রদান করে; অন্যথায় এটি কোনো আউটপুট দেয় না।

সুপারভাইজড লার্নিং **(supervised learning)** এবং ক্লাসিফিকেশন **(classification)**-এর প্রেক্ষাপটে, এই প্রক্রিয়াটি একটি নমুনার শ্রেণি (class) পূর্বাভাস দিতে ব্যবহার করা যায়।

পারসেপট্রন ফাংশন **(Perceptron Function)**

পারসেপট্রন হলো এমন একটি ফাংশন, যা তার ইনপুট "x"-কে শেখা ওজন সহগ **(learned weight coefficient)** দিয়ে গুণ করে একটি আউটপুট মান "f(x)" তৈরি করে।

উপরের সমীকরণে:

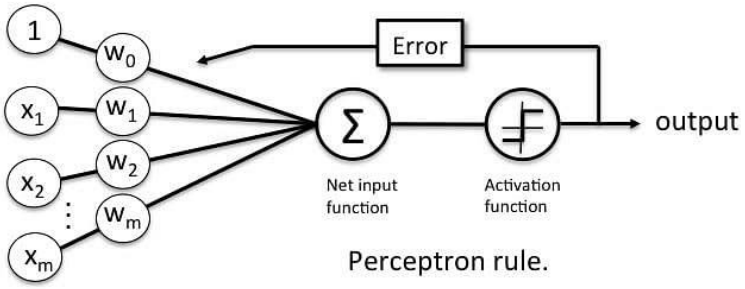
- **"w"** = বাস্তব মানবিশিষ্ট ওজনের ভেক্টর (vector of real-valued weights)
- **"b"** = বায়াস (bias) — একটি উপাদান যা ইনপুট মানের উপর নির্ভর না করে সিদ্ধান্ত সীমা (decision boundary)-কে উৎস (origin) থেকে দূরে সরিয়ে দেয়
- **"x"** = ইনপুট মানগুলির ভেক্টর
- **"m"** = পারসেপট্রনে প্রদত্ত ইনপুটের সংখ্যা

আউটপুটটি **"1"** বা **"0"** আকারে উপস্থাপিত হতে পারে।
কিছু ক্ষেত্রে, ব্যবহৃত অ্যাক্টিভেশন ফাংশন অনুযায়ী এটি **"1"** বা **"-1"** আকারেও প্রকাশ করা যায়।

পারসেপট্রনের ইনপুটসমূহ **(Inputs of a Perceptron)**

একটি পারসেপট্রন এক বা একাধিক ইনপুট গ্রহণ করে, এগুলোকে নির্দিষ্ট ওজন মান **(weight values)** দিয়ে নিয়ন্ত্রণ বা মডারেট করে, তারপর একটি ট্রান্সফরমেশন ফাংশন **(transformation function)** প্রয়োগ করে চূড়ান্ত আউটপুট প্রদান করে।

নিচের চিত্রে (image)-এ একটি বুলিয়ান আউটপুট **(Boolean output)**-সহ পারসেপট্রন দেখানো হয়েছে।



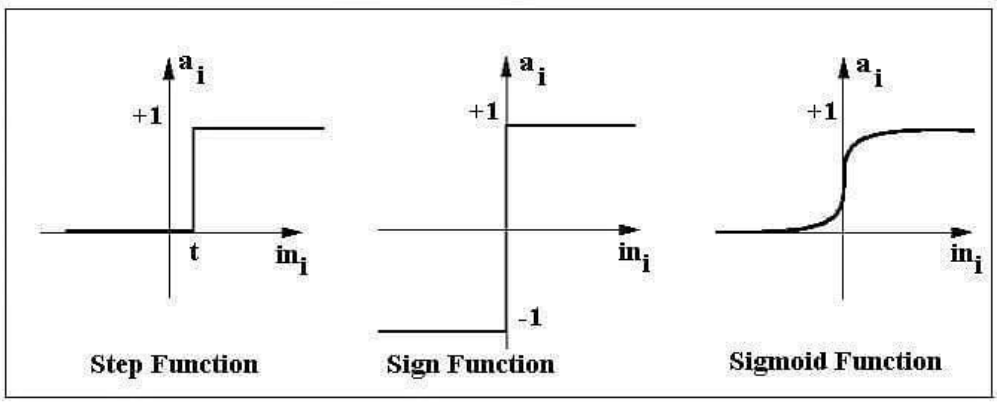
Perceptron rule.

একটি বুলিয়ান আউটপুট **(Boolean output)** এমন কিছু ইনপুটের উপর ভিত্তি করে নির্ধারিত হয়, যেমন — বেতনভুক্ত (salaried), বিবাহিত (married), বয়স (age), পূর্ববর্তী ক্রেডিট প্রোফাইল (past credit profile) ইত্যাদি। এটির মাত্র দুটি মান থাকে: হ্যাঁ বা না **(Yes/No)** অথবা সত্য বা মিথ্যা **(True/False)**। সমষ্টি ফাংশন **"∑"** ইনপুট "x"-এর প্রতিটি মানকে তার সংশ্লিষ্ট ওজন **"w"** দিয়ে গুণ করে, তারপর সেগুলোর যোগফল হিসাব করে নিচের মতো:

$$w_0 + w_1x_1 + w_2x_2 + \cdots + w_nx_n$$

পারসেপট্রনের অ্যাক্টিভেশন ফাংশন **(Activation Functions of Perceptron)**

অ্যাক্টিভেশন ফাংশন একটি স্টেপ রুল **(step rule)** প্রয়োগ করে, যা সংখ্যাগত আউটপুটকে **+1** বা **-1** এ রূপান্তরিত করে। এর মাধ্যমে যাচাই করা হয় যে, ওজনযুক্ত ফাংশনের **(weighting function)** আউটপুট শূন্যের চেয়ে বেশি কি না।



উদাহরণস্বরূপ:
যদি ∑ wixi > 0 হয় ⇒ তাহলে চূড়ান্ত আউটপুট "o" = 1 (ব্যাংক ঋণ মঞ্জুর করুন)
অন্যথায়, চূড়ান্ত আউটপুট "o" = -1 (ব্যাংক ঋণ অস্বীকার করুন)

স্টেপ ফাংশন **(Step Function)** নিউরনের আউটপুট একটি নির্দিষ্ট মানের উপরে গেলে সক্রিয় হয়; অন্যথায় এটি শূন্য আউটপুট প্রদান করে।
সাইন ফাংশন **(Sign Function)** নিউরনের আউটপুট শূন্যের চেয়ে বড় কি না তার উপর ভিত্তি করে **+1** বা **-1** প্রদান করে।
সিগময়েড ফাংশন **(Sigmoid Function)** একটি **S-**আকৃতির বক্ররেখা **(S-curve)** এবং এটি 0 এবং 1 এর মধ্যে একটি মান প্রদান করে।

পারসেপট্রনের আউটপুট **(Output of Perceptron)**

বুলিয়ান আউটপুটসহ পারসেপট্রন **(Perceptron with a Boolean output):**
ইনপুটসমূহ: $x_1 \dots x_n$
আউটপুট: $o(x_1 \dots x_n)$

$$o(x_1, \dots, x_n) = \begin{cases} 1 \text{ if } w_0 + w_1 x_1 + w_2 x_2 + \cdots + w_n x_n > 0 \\ -1 \text{ otherwise} \end{cases}$$

ওজন **(Weights):**
wi ⇒ ইনপুট **xi**-এর পারসেপট্রন আউটপুটে অবদান নির্দেশ করে।
w0 ⇒ বায়াস (bias) বা থ্রেশহোল্ড (threshold)।

যদি $\sum \mathbf{w.x} > 0$ হয়, তাহলে আউটপুট হবে **+1**, অন্যথায় **-1**।
নিউরন তখনই সক্রিয় (triggered) হয় যখন ওজনযুক্ত ইনপুট **(weighted input)** একটি নির্দিষ্ট থ্রেশহোল্ড মানে পৌঁছে যায়।

$$o(\vec{x}) = sgn(\vec{w} \cdot \vec{x})$$

$$sgn(y) = \begin{cases} 1 \text{ if } y > 0 \\ -1 \text{ otherwise} \end{cases}$$

আউটপুট **+1** নির্দেশ করে যে নিউরনটি সক্রিয় (triggered) হয়েছে।
আউটপুট **-1** নির্দেশ করে যে নিউরনটি সক্রিয় হয়নি।

"sgn" হলো সাইন ফাংশন **(sign function)**, যার আউটপুট **+1** বা **-1** হতে পারে।

পারসেপট্রনের ত্রুটি **(Error in Perceptron)**

পারসেপট্রন লার্নিং রুল **(Perceptron Learning Rule)**-এ, পূর্বাভাসিত আউটপুটটি (predicted output) পরিচিত বা প্রত্যাশিত আউটপুটের (known output) সাথে তুলনা করা হয়।
যদি এটি মেলে না, তবে ত্রুটি **(error)** বিপরীত দিকে প্রেরিত হয় (**backpropagation**) যাতে ওজনের মানগুলি সমন্বয় (weight adjustment) করা যায়।

পারসেপট্রন: সিদ্ধান্ত ফাংশন **(Perceptron: Decision Function)**

পারসেপট্রনের একটি সিদ্ধান্ত ফাংশন **φ(z)** সংজ্ঞায়িত করা হয়, যা **x** এবং **w** ভেক্টরগুলির একটি রৈখিক সংমিশ্রণ **(linear combination)** গ্রহণ করে।

$$\boldsymbol{w} = \begin{bmatrix} w_1 \\ \vdots \\ w_m \end{bmatrix}, \quad \boldsymbol{x} = \begin{bmatrix} x_1 \\ \vdots \\ x_m \end{bmatrix}$$

decision function **z**-এর মান দেওয়া হয় নিম্নরূপঃ

$$z = w_1 x_1 + \ldots + w_m x_m$$

যদি **z** এর মান থ্রেশহোল্ড **θ**-এর চেয়ে বেশি হয়, তবে সিদ্ধান্ত ফাংশনের **(decision function)** মান হবে **+1**; অন্যথায়, এর মান হবে **-1**।

$$\phi(z) = \begin{cases} 1 & if\ z \geq \theta \\ -1 & otherwise \end{cases}$$

এটাই হলো পারসেপট্রন অ্যালগরিদম **(Perceptron Algorithm)**।

বায়াস ইউনিট **(Bias Unit)**

সহজভাবে বোঝানোর জন্য, থ্রেশহোল্ড **θ**-কে সমীকরণের বাম পাশে আনা যায় এবং সেটিকে $\mathbf{w_0 x_0}$ আকারে প্রকাশ করা যায়,
যেখানে $\mathbf{w_0 = -\theta}$ এবং $\mathbf{x_0 = 1}$।

$$z = w_0 x_0 + w_1 x_1 + \ldots + w_m x_m = \boldsymbol{w}^T \boldsymbol{x}$$

$\mathbf{w_0}$-এর মানকে বায়াস ইউনিট **(bias unit)** বলা হয়। এরপর **decision function** হয়:

$$\phi(z) = \begin{cases} 1 & if\ z \geq 0 \\ -1 & otherwise \end{cases}$$

পারসেপট্রন এক নজরে **(Perceptron at a Glance)**

পারসেপট্রনের নিম্নলিখিত বৈশিষ্ট্য রয়েছে:

- পারসেপট্রন হলো একটি অ্যালগরিদম **(algorithm)** যা সুপারভাইজড লার্নিং **(Supervised Learning)**-এর মাধ্যমে এক স্তরের **(single layer)** বাইনারি লিনিয়ার ক্লাসিফায়ার **(binary linear classifiers)** শেখার জন্য ব্যবহৃত হয়।

- সর্বোত্তম ওজন সহগ **(optimal weight coefficients)** স্বয়ংক্রিয়ভাবে শেখা হয়।

- ওজনগুলো ইনপুট ফিচারের সাথে গুণ করা হয় এবং সিদ্ধান্ত নেওয়া হয় যে নিউরনটি সক্রিয় হবে কি হবে না।

- অ্যাক্টিভেশন ফাংশন **(activation function)** একটি স্টেপ রুল **(step rule)** প্রয়োগ করে যাচাই করে যে ওজনযুক্ত ফাংশনের আউটপুট শূন্যের চেয়ে বড় কি না।

- একটি রৈখিক সিদ্ধান্ত সীমা **(linear decision boundary)** অঙ্কিত হয়, যা +1 এবং -1 দুটি রৈখিকভাবে আলাদা শ্রেণি **(linearly separable classes)** এর মধ্যে পার্থক্য করতে সাহায্য করে।

- যদি ইনপুট সংকেতগুলির যোগফল একটি নির্দিষ্ট থ্রেশহোল্ড অতিক্রম করে, তবে এটি একটি সংকেত আউটপুট দেয়; অন্যথায় কোনো আউটপুট দেয় না।

- অ্যাক্টিভেশন ফাংশনের ধরনগুলোর মধ্যে রয়েছে সাইন **(sign),** স্টেপ **(step),** এবং সিগময়েড **(sigmoid)** ফাংশন।

পারসেপট্রন দিয়ে লজিক গেট **(Implement Logic Gates with Perceptron)**

- পারসেপট্রন **-** ক্লাসিফায়ার হাইপারপ্লেন **(Classifier Hyperplane)**

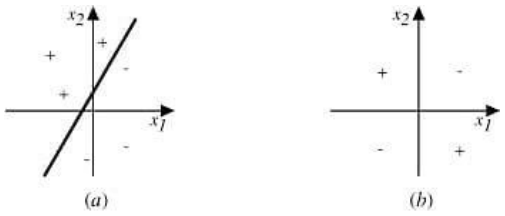
- পারসেপট্রন লার্নিং রুল তখনই সংযুক্ত হয় (converges), যখন দুটি শ্রেণি রৈখিক হাইপারপ্লেন **(linear hyperplane)** দ্বারা আলাদা করা যায়।

- তবে, যদি শ্রেণিগুলো পূর্ণভাবে রৈখিকভাবে আলাদা করা না যায়, তাহলে এটি ত্রুটি **(errors)** সৃষ্টি করতে পারে।

যেমনটি পূর্বের আলোচনায় বলা হয়েছে, পারসেপট্রনে বাইনারি আউটপুটের ক্লাসিফায়ার সীমা **(classifier boundary)** নিম্নলিখিত সমীকরণের মাধ্যমে উপস্থাপিত হয়:

$$\vec{w} \cdot \vec{x} = 0$$

উপরের চিত্রে দুটি ইনপুট বিশিষ্ট পারসেপট্রন **(two-input Perceptron)** দ্বারা উপস্থাপিত সিদ্ধান্ত পৃষ্ঠ **(decision surface)** দেখানো হয়েছে।



পর্যবেক্ষণ **(Observation):**

উপরের **Fig (a)**-তে উদাহরণগুলো স্পষ্টভাবে ধনাত্মক **(positive)** এবং ঋণাত্মক **(negative)** মানে ভাগ করা যায়; তাই এগুলো রৈখিকভাবে পৃথকযোগ্য **(linearly separable)**। এতে **AND, OR, NOR, NAND**-এর মতো লজিক গেট অন্তর্ভুক্ত থাকতে পারে।

Fig (b)-তে উদাহরণগুলো রৈখিকভাবে পৃথকযোগ্য নয় **(not linearly separable)**, যেমন একটি **XOR** গেট।

- চিত্র (a) হলো একটি প্রশিক্ষণ উদাহরণ সেট এবং পারসেপট্রনের সিদ্ধান্ত পৃষ্ঠ (decision surface), যা সেগুলো সঠিকভাবে শ্রেণীবদ্ধ করে।

- চিত্র (b) হলো এমন প্রশিক্ষণ উদাহরণ সেট, যা রৈখিকভাবে পৃথকযোগ্য নয়, অর্থাৎ কোনো সরলরেখা দিয়ে সঠিকভাবে শ্রেণীবদ্ধ করা যায় না।
  $\mathbf{X}_1$ এবং $\mathbf{X}_2$ হলো পারসেপট্রনের ইনপুট।

লজিক গেট কি**? (What is Logic Gate?)**

লজিক গেট হলো একটি ডিজিটাল সিস্টেমের **(digital system)** মৌলিক উপাদান, বিশেষত নিউরাল নেটওয়ার্কে। সংক্ষেপে, এগুলো ইলেকট্রনিক সার্কিট, যা যোগ (addition), নির্বাচন (choice), নেগেশন (negation), এবং সংমিশ্রণ (combination) করে জটিল সার্কিট তৈরি করতে সাহায্য করে।

লজিক গেট ব্যবহার করে, নিউরাল নেটওয়ার্ক নিজে থেকে শিখতে পারে, আপনাকে ম্যানুয়ালি লজিক কোড করতে হবে না। বেশিরভাগ লজিক গেটের দুটি ইনপুট এবং একটি আউটপুট থাকে।

প্রতিটি টার্মিনাল (terminal)-এর দুটি বাইনারি অবস্থার একটি থাকে: **low (0)** বা **high (1)**, যা ভোল্টেজ স্তরের মাধ্যমে প্রকাশ করা হয়। সার্কিটে ডেটা প্রক্রিয়াকরণের উপর ভিত্তি করে টার্মিনালের লজিক অবস্থা পরিবর্তিত হয়।

এই লজিকের ভিত্তিতে, লজিক গেটগুলো সাতটি ধরনের হতে পারে:

1. AND
2. NAND

3. OR
4. NOR
5. NOT
6. XOR
7. XNOR

পারসেপট্রন দিয়ে মৌলিক লজিক গেট বাস্তবায়ন **(Implementing Basic Logic Gates With Perceptron)**

নিম্নলিখিত লজিক গেটগুলো পারসেপট্রনের মাধ্যমে বাস্তবায়ন করা যায়:

1. AND

যদি দুটি ইনপুট **TRUE (+1)** হয়, পারসেপট্রনের আউটপুট ধনাত্মক হয়, যা **TRUE** নির্দেশ করে।
এটি **AND** গেটের প্রত্যাশিত আচরণ।

উদাহরণ:

- $x_1 = 1$ (TRUE), $x_2 = 1$ (TRUE)
- $w_0 = -0.8$, $w_1 = 0.5$, $w_2 = 0.5$

$o(x1,x2) = -0.8 + 0.5*1 + 0.5*1 = 0.2 > 0 o(x_1, x_2) = -0.8 + 0.5*1 + 0.5*1 = 0.2 > 0 o(x1,x2) = -0.8 + 0.5*1 + 0.5*1 = 0.2 > 0$

2. OR

যদি দুটি ইনপুটের যেকোনো একটি **TRUE (+1)** হয়, পারসেপট্রনের আউটপুট ধনাত্মক হয়, যা **TRUE** নির্দেশ করে।

উদাহরণ:

- $x_1 = 1$ (TRUE), $x_2 = 0$ (FALSE)
- $w_0 = -0.3$, $w_1 = 0.5$, $w_2 = 0.5$

$o(x1,x2) = -0.3 + 0.5*1 + 0.5*0 = 0.2 > 0 o(x_1, x_2) = -0.3 + 0.5*1 + 0.5*0 = 0.2 > 0 o(x1,x2) = -0.3 + 0.5*1 + 0.5*0 = 0.2 > 0$

3. XOR

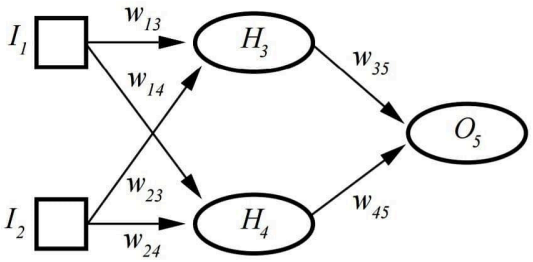
XOR গেট **(Exclusive OR gate)**-এর দুটি ইনপুট এবং একটি আউটপুট থাকে। গেটটি **TRUE** আউটপুট দেয় শুধুমাত্র তখনই, যদি একটি ইনপুট **TRUE** হয় এবং অন্যটি **FALSE**।

XOR ট্রুথ টেবিল **(XOR Truth Table):**

ইনপুট **A**	ইনপুট **B**	আউটপুট **Y**
0	0	0
0	1	1
1	0	1
1	1	0

XOR গেট এবং নিউরাল নেটওয়ার্ক **(XOR Gate with Neural Networks)**

AND এবং OR গেটের বিপরীতে, **XOR** গেট-এর জন্য একটি মধ্যবর্তী হিডেন লেয়ার **(intermediate hidden layer)** প্রয়োজন। এটি প্রাথমিক রূপান্তর **(preliminary transformation)** সম্পাদন করে, যাতে XOR-এর লজিক অর্জন করা যায়।



একটি **XOR** গেট এমনভাবে ওজন (weights) নির্ধারণ করে যাতে XOR শর্তগুলো পূরণ হয়। এটি এক স্তরের পারসেপট্রন **(single layer Perceptron)** দিয়ে বাস্তবায়ন করা সম্ভব নয় এবং মাল্টি-লেয়ার পারসেপট্রন **(Multi-layer Perceptron** বা **MLP)** প্রয়োজন।

- **H** হলো হিডেন লেয়ার **(hidden layer)**, যা XOR বাস্তবায়ন করতে সাহায্য করে।

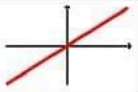
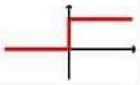
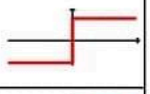
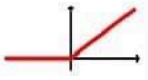
- I_1, I_2, H_3, H_4, O_5 হলো **0 (FALSE)** বা **1 (TRUE)**।
- t_3 = H_3-এর থ্রেশহোল্ড; t_4 = H_4-এর থ্রেশহোল্ড; t_5 = O_5-এর থ্রেশহোল্ড।

$H3=sigmoid(I1*w13+I2*w23-t3), H4=sigmoid(I1*w14+I2*w24-t4)H_3 = \text{sigmoid}(I_1*w_{13} + I_2*w_{23} - t_3), \quad H_4 = \text{sigmoid}(I_1*w_{14} + I_2*w_{24} - t_4)H3=sigmoid(I1*w13+I2*w23-t3), H4=sigmoid(I1*w14+I2*w24-t4)$
$O5=sigmoid(H3*w35+H4*w45-t5)O_5 = \text{sigmoid}(H_3*w_{35} + H_4*w_{45} - t_5)O5=sigmoid(H3*w35+H4*w45-t5)$

পরবর্তী অংশে আসুন সিগময়েড অ্যাক্টিভেশন ফাংশন **(Sigmoid Activation Function)** সম্পর্কে আরও শিখি।

অ্যাক্টিভেশন ফাংশনের সংক্ষিপ্ত পর্যালোচনা **(Activation Functions at a Glance)**

পারসেপট্রনের সাথে ব্যবহারযোগ্য বিভিন্ন অ্যাক্টিভেশন ফাংশন নিচে দেখানো হলো:

Activation Function	Equation	Example	1D Graph
Linear	$\phi(z) = z$	Adaline, linear regression	
Unit Step (Heaviside Function)	$\phi(z) = \begin{cases} 0 & z < 0 \\ 0.5 & z = 0 \\ 1 & z > 0 \end{cases}$	Perceptron variant	
Sign (signum)	$\phi(z) = \begin{cases} -1 & z < 0 \\ 0 & z = 0 \\ 1 & z > 0 \end{cases}$	Perceptron variant	
Piece-wise Linear	$\phi(z) = \begin{cases} 0 & z \leq -½ \\ z + ½ & -½ \leq z \leq ½ \\ 1 & z \geq ½ \end{cases}$	Support vector machine	
Logistic (sigmoid)	$\phi(z) = \frac{1}{1 + e^{-z}}$	Logistic regression, Multilayer NN	
Hyperbolic Tangent (tanh)	$\phi(z) = \frac{e^{z} - e^{-z}}{e^{z} + e^{-z}}$	Multilayer NN, RNNs	
ReLU	$\phi(z) = \begin{cases} 0 & z < 0 \\ z & z > 0 \end{cases}$	Multilayer NN, CNNs	

অ্যাক্টিভেশন ফাংশন ব্যবহারের সিদ্ধান্ত সম্পূর্ণরূপে ডেটা সায়েন্টিস্টের উপর নির্ভর করে, সমস্যা বিবরণ এবং কাঙ্ক্ষিত ফলাফলের ধরন অনুযায়ী। যদি লার্নিং প্রক্রিয়া ধীরগতির হয় বা ভ্যানিশিং বা এক্সপ্লোডিং গ্রেডিয়েন্টস সমস্যায় পড়ে, ডেটা সায়েন্টিস্ট অ্যাক্টিভেশন ফাংশন পরিবর্তন করে দেখতে পারেন যে সমস্যাগুলো সমাধান করা যায় কিনা।

পারসেপট্রনের ভবিষ্যত

মেশিন লার্নিং-এর জনপ্রিয়তা ও ব্যবহার বৃদ্ধির সাথে, পারসেপট্রনের ভবিষ্যত উজ্জ্বল মনে হচ্ছে। এটি ডেটা বিশ্লেষণ করতে সাহায্য করে, অন্তর্নিহিত প্যাটার্নগুলো তৈরি করে এবং দ্রুত প্রয়োগ করে। কোডিং ক্রমবর্ধমানভাবে বিকশিত হচ্ছে, এবং পারসেপট্রন প্রযুক্তি আধুনিক কম্পিউটারগুলোতে বিশ্লেষণাত্মক ক্ষমতা বৃদ্ধিতে অব্যাহতভাবে সহায়তা করবে।

সারাংশ

- একটি কৃত্রিম নিউরন **(Artificial Neuron)** হলো একটি গাণিতিক ফাংশন যা জীববৈজ্ঞানিক নিউরনের মডেল হিসেবে ধারণা করা হয়েছে, অর্থাৎ এটি একটি নিউরাল নেটওয়ার্ক।

- পারসেপট্রন হলো একটি নিউরাল নেটওয়ার্ক ইউনিট, যা ইনপুট ডেটা থেকে ফিচার বা ব্যবসায়িক ইন্টেলিজেন্স শনাক্ত করতে নির্দিষ্ট গণনা সম্পন্ন করে। এটি একটি ফাংশন যা ইনপুট “x”-কে শিখিত ওজন সহ মানচিত্রায়িত করে এবং একটি আউটপুট মান “f(x)” উৎপন্ন করে।

- পারসেপট্রন লার্নিং রুল বলে যে অ্যালগরিদম স্বয়ংক্রিয়ভাবে সর্বোত্তম ওজন নির্ধারণ শিখবে।

- সিঙ্গল লেয়ার পারসেপট্রন শুধুমাত্র লিনিয়ারলি সেপারেবল প্যাটার্ন শিখতে পারে।

- মাল্টিলেয়ার পারসেপট্রন বা ফিডফরওয়ার্ড নিউরাল নেটওয়ার্ক (দুটি বা ততোধিক লেয়ারসহ) বড় প্রসেসিং ক্ষমতা রাখে এবং নন-লিনিয়ার প্যাটার্নও প্রক্রিয়াজাত করতে পারে।

- পারসেপট্রন লজিক গেট যেমন AND, OR, XOR বাস্তবায়ন করতে পারে।

ডেল্টা রুল দিয়ে থ্রেশহোল্ডেড পারসেপট্রন ট্রেনিং

ডেল্টা রুল **(Delta Rule)** বা লিস্ট মিন স্কোয়ারস **(LMS)** রুল ব্যবহার করে অথ্রেশহোল্ডেড পারসেপট্রন ট্রেন করা হয়।

- সাধারণ পারসেপট্রন ট্রেনিং রুল থ্রেশহোল্ডেড আউটপুটের উপর নির্ভরশীল।

- ডেল্টা রুল কন্টিনিউয়াস আউটপুট (সাধারণত 0 এবং 1-এর মধ্যে, যদি সিগময়েড অ্যাক্টিভেশন ব্যবহার করা হয়) সহ পারসেপট্রনের জন্য প্রযোজ্য।

- এটি প্রকৃত আউটপুট এবং কাঙ্ক্ষিত আউটপুটের পার্থক্য কমায়, এবং মাল্টিলেয়ার পারসেপট্রন ট্রেনিং-এর ভিত্তি গঠন করে।

গ্রেডিয়েন্ট ডিসেন্ট কেন?

যেমন আমরা পূর্বে আলোচনা করেছি, পারসেপট্রন ট্রেনিং রুল লিনিয়ারলি সেপারেবল ডেটার জন্য কার্যকর।

- সীমাবদ্ধতা: যদি একাধিক লোকাল মিনিমা থাকে, রুলটি লোকাল মিনিমায় কনভার্জ হতে পারে, গ্লোবাল মিনিমায় নয়।

- সমাধান: আমরা গ্রেডিয়েন্ট ডিসেন্ট ব্যবহার করে পারসেপট্রন ট্রেন করব।

গ্রেডিয়েন্ট ডিসেন্ট হলো একটি গাণিতিক অ্যালগরিদম, যা পুনরাবৃত্তভাবে ফাংশনের নেতিবাচক গ্রেডিয়েন্টের দিকে চলে যায় এবং ফাংশনের সর্বনিম্ন মান খুঁজে বের করে।

এটি কিভাবে কাজ করে?

গ্রেডিয়েন্ট ডিসেন্ট বা ডেল্টা রুলের মূল ধারণা হলো:

- আমরা সম্ভাব্য সব ওজন ভেক্টরের হাইপোথিসিস স্পেসে অনুসন্ধান করি, যাতে সর্বোত্তম ওজন নির্ধারণ করা যায়।

- গ্রেডিয়েন্ট ডিসেন্ট হলো ব্যাকপ্রোপাগেশন অ্যালগরিদমের ভিত্তি, যা আমরা পরবর্তী আলোচনায় দেখব।

ডেল্টা রুল বোঝার উপায়:

- এটি হলো থ্রেশহোল্ডেড পারসেপট্রনকে গ্রেডিয়েন্ট ডিসেন্ট ব্যবহার করে ট্রেন করা।

- ওজন এবং সংশ্লিষ্ট ইনপুটের লিনিয়ার কম্বিনেশন অ্যাক্টিভেশন ফাংশনের ইনপুট হিসেবে কাজ করে।
  $O(x)=w \cdot x$

ট্রেনিং ত্রুটি গণনা:

- অ্যাক্টিভেশন ফাংশনে যাওয়ার আগে, মিসক্লাসিফিকেশনের ক্ষেত্রে ওজনের ত্রুটি কিভাবে গণনা করবেন তা জানা জরুরি।

- গ্রেডিয়েন্ট ডিসেন্টে সাধারণত আউটপুট এবং প্রাপ্ত মানের অর্ধেক বর্গ পার্থক্য ব্যবহার করা হয় হাইপোথিসিসের ট্রেনিং ত্রুটির জন্য।

$$E(\vec{w}) = \frac{1}{2} \sum_{d \epsilon D} (t_d - o_d)^2$$

গ্রেডিয়েন্ট ডিসেন্টের জন্য ট্রেনিং ত্রুটি

- **D** হলো ট্রেনিং স্যাম্পলস এর একটি সেট।

- **t** হলো ট্রেনিং উদাহরণের লক্ষ্য আউটপুট (target output) 'd' এর জন্য।

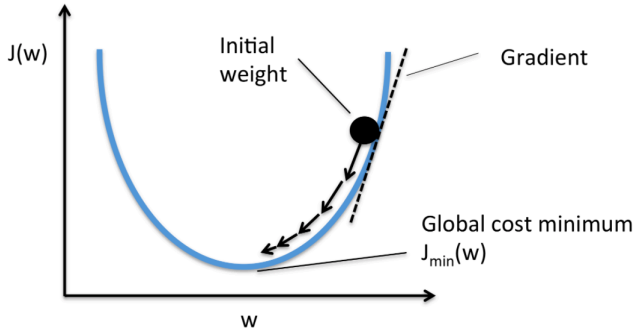
- **o** হলো ট্রেনিং উদাহরণের লিনিয়ার ইউনিটের আউটপুট 'd' এর জন্য।

এই উদাহরণে আমরা অ্যাক্টিভেশন ফাংশন হিসেবে একই ইউনিট স্টেপ ফাংশন ব্যবহার করব।

ওজন ভেক্টর এবং তাদের ত্রুটি মান ভিজ্যুয়ালাইজ করা:

(এখানে, "ভিজ্যুয়ালাইজেশন" বলতে, ওজন ভেক্টরের মান এবং সংশ্লিষ্ট ত্রুটির পরিবর্তন গ্রাফিক্যালভাবে উপস্থাপন করার ধারণা বোঝানো হয়েছে।)

গ্রেডিয়েন্ট ডিসেন্টে ট্রেনিং ত্রুটি গণনা করার জন্য, আমরা সাধারণত আউটপুট এবং লক্ষ্য আউটপুটের মধ্যে পার্থক্য ব্যবহার করি।



গ্রেডিয়েন্ট ডিসেন্ট

- **w** সমস্ত সম্ভব ওজন ভেক্টর (weight vectors) কে প্রতিনিধিত্ব করে।
- **J(w)** প্রতিনিধিত্ব করে ওজন ভেক্টরের সম্পর্কিত ত্রুটি মান (error values)।
- ত্রুটি সপাটের (error surface) প্যারাবোলিক পথ, যার একটি গ্লোবাল মিনিমা (global minima) থাকে, যেখানে ওজন ভেক্টরগুলি প্রশিক্ষণ স্যাম্পলগুলির জন্য সবচেয়ে উপযুক্ত।

গ্রেডিয়েন্ট ডিসেন্টে ওয়েট আপডেশন রুল

আমরা আমাদের চূড়ান্ত সমীকরণে পৌঁছেছি যা ডেল্টা রুল ব্যবহার করে ওয়েট আপডেট করার জন্য। একটি মূল পার্থক্য হলো, perceptron রুলে আমরা সমস্ত স্যাম্পল ট্রেনিং করার পর ওয়েট পরিবর্তন করি, কিন্তু ডেল্টা রুলে আমরা প্রতিটি মিসক্লাসিফিকেশন পরবর্তী ওয়েট আপডেট করি, যার ফলে গ্লোবাল মিনিমা পাওয়ার সম্ভাবনা বৃদ্ধি পায়।

অ্যালগরিদম:

1. ওয়েট ভেক্টর ইনিশিয়ালাইজ করুন।

2. ডেটাসেটের প্রতিটি ট্রেনিং স্যাম্পলের জন্য, অ্যাক্টিভেশন ফাংশন প্রয়োগ করুন এবং যদি কোনো ত্রুটি ঘটে, তাহলে রুল অনুসারে ওয়েট আপডেট করুন।

3. একাধিক ইপোকসের জন্য এটি পুনরাবৃত্তি করুন, যাতে এটি আরও সঠিক হয়।

অ-রৈখিকতার প্রয়োজনীয়তা কেন?

নিউরাল নেটওয়ার্কে অ-রৈখিকতা (non-linearity) গুরুত্বপূর্ণ কারণ এটি ডেটার জটিল নিদর্শনগুলি ক্যাপচার করতে সাহায্য করে, যা সাধারণত একটি সহজ রৈখিক সিদ্ধান্ত সীমানার দ্বারা আলাদা করা যায় না। এখানে প্রধান কারণগুলি রয়েছে কেন অ-রৈখিকতা প্রয়োজন:

1. অ-রৈখিকভাবে বিচ্ছিন্ন ডেটা **(Non-Linearly Separable Data):**
   অনেক বাস্তব জীবনের সমস্যা যেমন ইমেজ রিকগনিশন, ভাষা প্রক্রিয়াকরণ এবং জটিল শ্রেণীবিভাজন কাজগুলিতে ডেটা অ-রৈখিকভাবে বিচ্ছিন্ন হয়। একটি নিখুঁত রৈখিক মডেল এই সমস্যাগুলি সমাধান করতে সক্ষম হবে না, কারণ এটি কেবলমাত্র রৈখিক সিদ্ধান্ত সীমানা তৈরি করতে সীমাবদ্ধ থাকবে।

2. জটিল নিদর্শন শেখার সক্ষমতা**:**
   সিগময়েড, ReLU, এবং টানহ (tanh) এর মতো অ-রৈখিক অ্যাক্টিভেশন ফাংশনগুলি নেটওয়ার্কে অ-রৈখিকতা প্রবাহিত করে, যা ডেটাতে জটিল সম্পর্ক মডেল করতে সহায়ক। এই ফাংশনগুলি নেটওয়ার্ককে ইনপুট এবং আউটপুটের মধ্যে জটিল ম্যাপিং শিখতে সহায়তা করে, যার ফলে প্রয়োজনে বাঁকানো বা এমনকি বহু-মাত্রিক সিদ্ধান্ত সীমানা তৈরি হয়।

3. লেয়ারগুলি কার্যকরভাবে স্ট্যাকিং**:**
   একটি বহু-স্তরের নিউরাল নেটওয়ার্কে, লেয়ারগুলির মধ্যে অ-রৈখিক অ্যাক্টিভেশন ফাংশন প্রয়োগ করা প্রতিটি লেয়ারকে ইনপুট ডেটাকে বিভিন্নভাবে রূপান্তরিত করতে দেয়। অ-রৈখিকতা ছাড়া, একাধিক লেয়ার কেবল একটি সোজা রৈখিক রূপান্তর হয়ে যাবে, যা গভীর নেটওয়ার্কগুলিকে অকার্যকর করে ফেলবে। অ-রৈখিক ফাংশনগুলি প্রতিটি লেয়ারকে আগের লেয়ারের উপর ভিত্তি করে নতুনভাবে তৈরি করতে সহায়ক, যার ফলে ডেটার আরও বিমূর্ত এবং উচ্চ-স্তরের উপস্থাপনা তৈরি হয়।

নেটওয়ার্ক স্ট্রাকচারগুলি**:** ফিডফরওয়ার্ড নেটওয়ার্ক এবং রিকারেন্ট নেটওয়ার্ক

ফিডফরওয়ার্ড নেটওয়ার্ক

- স্ট্রাকচার**:** ফিডফরওয়ার্ড নিউরাল নেটওয়ার্কে তথ্য একমাত্রিকভাবে প্রবাহিত হয়—ইনপুট থেকে আউটপুট—কোনও ফিডব্যাক লুপ বা রিকারেন্ট সংযোগ ছাড়াই।

- লেয়ারস**:** এই নেটওয়ার্কগুলি সাধারণত একটি ইনপুট লেয়ার, এক বা একাধিক হিডেন লেয়ার এবং একটি আউটপুট লেয়ার নিয়ে গঠিত।

- অ্যাক্টিভেশন ফাংশন: প্রতিটি নিউরন তার ইনপুটে একটি অ্যাক্টিভেশন ফাংশন প্রয়োগ করে, তারপরে এটি পরবর্তী লেয়ারে পাঠায়।

- ব্যবহার: ফিডফরওয়ার্ড নেটওয়ার্কগুলি ইমেজ শ্রেণীবিভাগ, অবজেক্ট রিকগনিশন এবং এমন অন্যান্য অ্যাপ্লিকেশনগুলির জন্য ব্যাপকভাবে ব্যবহৃত হয় যেখানে ইনপুট-আউটপুট সম্পর্ক স্থির থাকে এবং কোনও সিকোয়েন্স বা সময়কালের নির্ভরতা থাকে না।

সুবিধা:

- সহজ এবং প্রশিক্ষণ করা সহজ।

- এমন কাজের জন্য কার্যকর যা ইনপুট থেকে আউটপুটে সরল ম্যাপিং প্রয়োজন।

সীমাবদ্ধতা:

- সিকোয়েন্সিয়াল বা সময়সীমাবদ্ধ ডেটার জন্য উপযুক্ত নয়, যেখানে পূর্ববর্তী ইনপুট থেকে প্রাসঙ্গিকতা বর্তমান সিদ্ধান্তকে প্রভাবিত করতে পারে (যেমন ভাষা প্রক্রিয়াকরণে)।

রিকারেন্ট নিউরাল নেটওয়ার্ক **(RNNs)**

- স্ট্রাকচার: রিকারেন্ট নিউরাল নেটওয়ার্কে এমন সংযোগ থাকে যা সিকোয়েন্সের প্রতিটি স্টেপে তথ্য সংরক্ষণ করতে পারে। তারা ফিডব্যাক লুপ অন্তর্ভুক্ত করে, যেখানে একটি নিউরনের আউটপুট তার ভবিষ্যত ইনপুটকে প্রভাবিত করতে পারে, এটি নেটওয়ার্ককে সময়ের সাথে তথ্য মনে রাখতে সক্ষম করে।

- হিডেন স্টেট: RNN গুলি একটি হিডেন স্টেট বজায় রাখে যা প্রতিটি টাইম স্টেপে আপডেট হয়, যার ফলে এটি পূর্ববর্তী ইনপুটগুলির তথ্য মনে রাখতে সক্ষম হয়, যা সিকোয়েন্সিয়াল ডেটার জন্য অত্যন্ত গুরুত্বপূর্ণ।

- ব্যবহার: RNN গুলি সাধারণত সিকোয়েন্সিয়াল ডেটায়, যেমন স্পিচ রিকগনিশন, ভাষা মডেলিং, টাইম সিরিজ ভবিষ্যদ্বাণী, এবং যেকোনো ডেটাতে যেখানে সিকোয়েন্সের নির্ভরতা থাকে, সেখানে ব্যবহৃত হয়।

সুবিধা:

- সিকোয়েন্সিয়াল এবং সময়-নির্ভর ডেটা পরিচালনার জন্য উপযুক্ত।

- একাধিক সময়সীমার মধ্যে তথ্য মনে রাখতে সক্ষম, যা মডেলকে প্রসঙ্গ প্রদান করে।

সীমাবদ্ধতা:

- RNN প্রশিক্ষণ করা চ্যালেঞ্জিং হতে পারে, বিশেষ করে দীর্ঘ সিকোয়েন্সগুলিতে তথ্য মনে রাখার চেষ্টা করার সময় ভ্যানিশিং গ্রেডিয়েন্ট সমস্যা দেখা দিতে পারে।

- RNN গুলি কম্পিউটেশনালভাবে ব্যয়বহুল হতে পারে, কারণ প্রতিটি টাইম স্টেপ প্রক্রিয়া করতে iterative প্রক্রিয়া প্রয়োজন।

RNN সমস্যাগুলি সমাধান করতে, লং শর্ট-টার্ম মেমরি (LSTM) এবং গেটেড রিকারেন্ট ইউনিট (GRU) এর মতো ভেরিয়েন্টগুলি তৈরি করা হয়েছে, যা RNN গুলিকে দীর্ঘ নির্ভরতা আরও কার্যকরভাবে পরিচালনা করতে সক্ষম।

জেনারেলাইজেশন, ওভারফিটিং, এবং স্টপিং ক্রাইটেরিয়া

মেশিন লার্নিং এবং নিউরাল নেটওয়ার্কে, একটি মডেল যা নতুন ডেটার উপর ভালভাবে জেনারেলাইজ করতে পারে, তা অর্জন করা অত্যন্ত গুরুত্বপূর্ণ। এই ধারণাগুলি জেনারেলাইজেশন, ওভারফিটিং, এবং স্টপিং ক্রাইটেরিয়া সম্পর্কে আলোচনা করা হলো।

1. জেনারেলাইজেশন

জেনারেলাইজেশন হলো এমন একটি নিউরাল নেটওয়ার্কের ক্ষমতা যা নতুন, অদেখা ডেটার (টেস্ট ডেটা) উপর ভালো পারফর্ম করে, যেটি বিশেষভাবে ট্রেনিং ডেটার উপর ট্রেনিং করা হয়েছে। একটি ভালো জেনারেলাইজিং মডেল ডেটাতে লুকানো নিদর্শনগুলি শিখে, ট্রেনিং উদাহরণগুলি মনে রাখে না।

2. ওভারফিটিং

ওভারফিটিং ঘটে যখন একটি মডেল ট্রেনিং ডেটাকে খুব ভালোভাবে শেখে, নNoise এবং অপ্রাসঙ্গিক বিস্তারিত (irrelevant details) সহ, যার ফলে ট্রেনিং ডেটার উপর উচ্চ সঠিকতা হলেও নতুন ডেটার উপর খারাপ পারফরম্যান্স হতে পারে।

ওভারফিটিংয়ের কারণ**:**

- জটিল মডেল**:** যখন মডেলটি খুব বেশি প্যারামিটার (যেমন অনেক লেয়ার বা নিউরন) নিয়ে থাকে, এটি ট্রেনিং ডেটা "মনে" রাখতে পারে।

- অপর্যাপ্ত ট্রেনিং ডেটা**:** কম ট্রেনিং ডেটা থাকলে, মডেলটি নির্দিষ্ট উদাহরণগুলি শিখতে পারে, সাধারণ নিদর্শন নয়।
- অতিরিক্ত প্রশিক্ষণ এপোক্সের কারণে ওভারফিটিং**:** অতিরিক্ত প্রশিক্ষণ এপোক্স (High Training Epochs): যখন মডেলটি অনেক এপোক্স ধরে প্রশিক্ষিত হয়, তখন এটি ট্রেনিং ডেটার বিশেষ বৈশিষ্ট্যগুলোর সাথে খাপ খাইয়ে নেয় এবং ওভারফিটিং ঘটতে পারে। এর ফলে মডেলটি ট্রেনিং ডেটার উপর ভাল কাজ করে, কিন্তু নতুন বা অদেখা ডেটার উপর খারাপ পারফরম্যান্স প্রদর্শন করে।

মডেলে ওভারফিটিং কীভাবে এড়ানো যাবে

ওভারফিটিং এবং আন্ডারফিটিং উভয়ই মেশিন লার্নিং মডেলের পারফরম্যান্স খারাপ করে তোলে। তবে মূল সমস্যা হলো ওভারফিটিং, তাই আমরা কিছু পদ্ধতি অনুসরণ করে আমাদের মডেলে ওভারফিটিং কমাতে পারি।

ওভারফিটিং কমানোর জন্য কিছু উপায়:

1. ক্রস-ভ্যালিডেশন **(Cross-Validation):**
   ক্রস-ভ্যালিডেশন পদ্ধতি ব্যবহার করলে আমরা ডেটাকে বিভিন্ন অংশে ভাগ করে মডেলটি প্রশিক্ষণ এবং টেস্ট করতে পারি। এর ফলে মডেলটি বিভিন্ন ডেটাসেটের উপর পরীক্ষা করা হয়, যা ওভারফিটিং কমাতে সাহায্য করে।

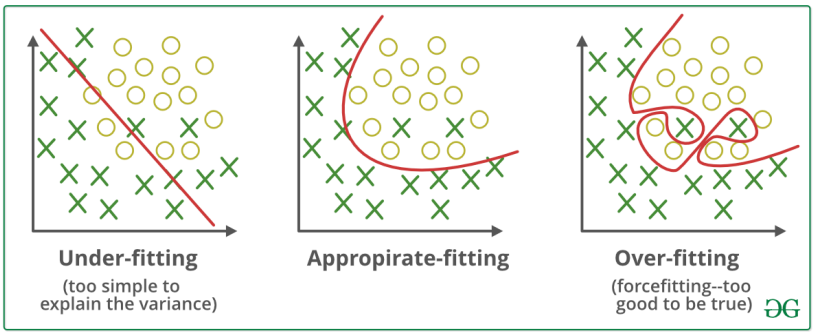
2. আরো ডেটা দিয়ে প্রশিক্ষণ **(Training with More Data):**
   প্রশিক্ষণ ডেটার পরিমাণ বাড়ালে মডেলটি আরও বেশি বৈশিষ্ট্য শিখতে পারে এবং ওভারফিটিং কম হয়। বেশি ডেটা মডেলকে সাধারণ নিদর্শনগুলি শিখতে সহায়তা করে, যেটি এটি অপ্রাসঙ্গিক বা ভুল তথ্য শিখতে বাধা দেয়।

3. ফিচার সরানো **(Removing Features):**
   কিছু বৈশিষ্ট্য বা ফিচার ডেটার মধ্যে অতিরিক্ত বা অপ্রয়োজনীয় তথ্য প্রদান করতে পারে, যা মডেলটির জটিলতা বাড়িয়ে দেয়। অপ্রয়োজনীয় বা কম গুরুত্বপূর্ণ ফিচারগুলি বাদ দিলে মডেলটি আরও সাধারণ এবং কার্যকরী হতে পারে।

4. প্রারম্ভিক প্রশিক্ষণ থামানো **(Early Stopping the Training):**
   যদি মডেলটি একাধিক এপোক্সের পর ট্রেনিং ডেটার উপর ভাল ফলাফল প্রদর্শন করতে থাকে, তবে প্রারম্ভিক প্রশিক্ষণ থামানো যেতে পারে। এর ফলে মডেলটি টেস্ট ডেটার উপর অপ্রয়োজনীয় ভাবে খাপ খাওয়ানোর সুযোগ পায় না এবং ওভারফিটিং রোধ করা হয়।

5. রেগুলারাইজেশন **(Regularization):**
   রেগুলারাইজেশন পদ্ধতিগুলি (যেমন L1, L2 রেগুলারাইজেশন) মডেলের প্রশিক্ষণের সময় অতিরিক্ত প্যারামিটার বা ওয়েট ভেক্টর ব্যবহার কমাতে সাহায্য করে। এর ফলে মডেলটি অত্যাধিক জটিলতা থেকে মুক্ত থাকে এবং সাধারণ নিদর্শন শিখে।

6. এনসেম্বলিং **(Ensembling):**
   এনসেম্বলিং পদ্ধতি বিভিন্ন মডেলের আউটপুটগুলিকে একত্রিত করে একটি একক মডেলের তুলনায় অধিক কার্যকর ফলাফল তৈরি করতে সহায়তা করে। এই পদ্ধতিটি সাধারণত মডেলের স্থিতিশীলতা বাড়ায় এবং ওভারফিটিং কমায়।



3. স্টপিং ক্রাইটেরিয়া **(Stopping Criterion)**

স্টপিং ক্রাইটেরিয়া নির্ধারণ করে কখন নিউরাল নেটওয়ার্কের প্রশিক্ষণ থামানো উচিত। সঠিক স্টপিং ক্রাইটেরিয়া না থাকলে, মডেলটি প্রশিক্ষণের ডেটার সাথে খাপ খাইয়ে নিতে পারে এবং অতিরিক্ত ফিটিং হতে পারে। সাধারণ স্টপিং ক্রাইটেরিয়া গুলি অন্তর্ভুক্ত:

- প্রারম্ভিক স্টপিং **(Early Stopping):** প্রশিক্ষণের সময় মডেলের পারফরম্যান্স একটি আলাদা ডেটা সেট (ভ্যালিডেশন সেট) তে মনিটর করা হয়। যখন ভ্যালিডেশন সেটে পারফরম্যান্স উন্নতি বন্ধ হয়ে যায়, তখন প্রশিক্ষণ থামিয়ে দেওয়া হয় যাতে ওভারফিটিং রোধ করা যায়।

- ফিক্সড নং অফ এপোক্স **(Fixed Number of Epochs):** পূর্বের জ্ঞান বা পরীক্ষার ভিত্তিতে একটি পূর্বনির্ধারিত সংখ্যক এপোক্স সেট করা হয়। তবে, এই পদ্ধতিটি কম অভিযোজিত এবং ওভারফিটিং প্রতিরোধ করতে খুব কার্যকর নাও হতে পারে।

- ভ্যালিডেশন-ভিত্তিক স্টপিং **(Validation-Based Stopping):** কিছু অ্যালগরিদম স্বয়ংক্রিয়ভাবে ভ্যালিডেশন ত্রুটি ট্র্যাক করে এবং এই ত্রুটি কমা বন্ধ বা বাড়লে প্রশিক্ষণ থামিয়ে দেয়।

4. ওভারফিটিং সমস্যা সমাধান **(Overcoming the Overfitting Problem)**

ওভারফিটিং সমস্যাটি সমাধান করার জন্য কিছু কৌশল প্রয়োগ করা যেতে পারে। এই পদ্ধতিগুলির মধ্যে ভ্যালিডেশন সেট ব্যবহার করা অন্যতম, কারণ এটি মডেলটির ট্রেনিং ডেটা ছাড়াও কতটা ভালো জেনারালাইজেশন করতে পারছে তা নিয়ে ফিডব্যাক দেয়।

ওভারফিটিং প্রতিরোধের কৌশলসমূহ**:**

- ভ্যালিডেশন সেট **(Validation Set):** ডেটাকে প্রশিক্ষণ, ভ্যালিডেশন, এবং টেস্ট সেটে ভাগ করুন। প্রশিক্ষণের সময়, মডেলটি প্রতিটি এপোক্সের পর ভ্যালিডেশন সেটে মূল্যায়িত হয়, যা ওভারফিটিং সনাক্ত করতে এবং প্রতিরোধ করতে সাহায্য করে।

    - প্রারম্ভিক স্টপিং **(Early Stopping):** যদি ভ্যালিডেশন সেটের পারফরম্যান্স স্থির হয়ে যায় বা খারাপ হতে শুরু করে, তাহলে মডেলটি সর্বোত্তম জেনারালাইজেশন অবস্থায় পৌঁছেছে বলে ধরে নিয়ে প্রশিক্ষণ বন্ধ করে দেওয়া হয়।

- রেগুলারাইজেশন কৌশল **(Regularization Techniques):**

    - **L1** এবং **L2** রেগুলারাইজেশন **(L1 and L2 Regularization):** লস ফাংশনে বড় ওয়েটগুলির জন্য একটি পেনাল্টি যোগ করা হয়, যা মডেলটিকে খুব জটিল হতে এবং ট্রেনিং ডেটার সাথে অতিরিক্ত ফিট হতে বাধা দেয়।

    - ড্রপআউট **(Dropout):** প্রতিটি প্রশিক্ষণ পদক্ষেপের সময় নিউরনের একটি উপসেট এলোমেলোভাবে বাদ দেওয়া হয় (তাদের অ্যাকটিভেশন শূন্যে সেট করা হয়), যা নেটওয়ার্কটিকে এমন বৈশিষ্ট্যগুলি শিখতে বাধ্য করে যা আরও ভালোভাবে জেনারালাইজ করতে পারে।

- ডেটা অগমেন্টেশন **(Data Augmentation):**
    - চিত্র ডেটার জন্য, ডেটা অগমেন্টেশন পদ্ধতি চিত্রগুলোকে ঘোরানো, উল্টানো, বা কিছু নয়েজ যোগ করার মাধ্যমে ডেটার পরিমাণ এবং বৈচিত্র্য বাড়াতে সহায়ক হতে পারে, যা মডেলটিকে আরও ভালো জেনারালাইজ করতে সাহায্য করে।

- মডেল জটিলতা কমানো **(Reduce Model Complexity):**
    - মডেলের আর্কিটেকচার সহজ করুন, যেমন লেয়ার বা নিউরনের সংখ্যা কমানো। ছোট মডেলগুলি কম ডেটার সাথে প্রশিক্ষিত হলে অতিরিক্ত ফিটিংয়ের সম্ভাবনা কম থাকে।

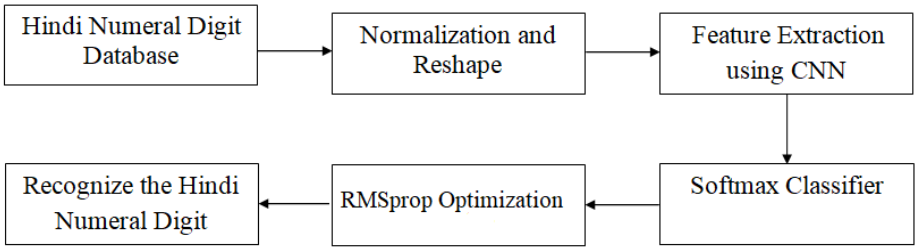
- ক্রস-ভ্যালিডেশন **(Cross-Validation):**
    - ক্রস-ভ্যালিডেশন (যেমন, K-fold ক্রস-ভ্যালিডেশন) ডেটাকে একাধিক অংশে ভাগ করে মডেলটি একাধিকবার প্রশিক্ষণ এবং পরীক্ষা করার পদ্ধতি। এটি মডেলের জেনারালাইজেশন পারফরম্যান্সের আরও শক্তিশালী পরিমাপ দেয়।

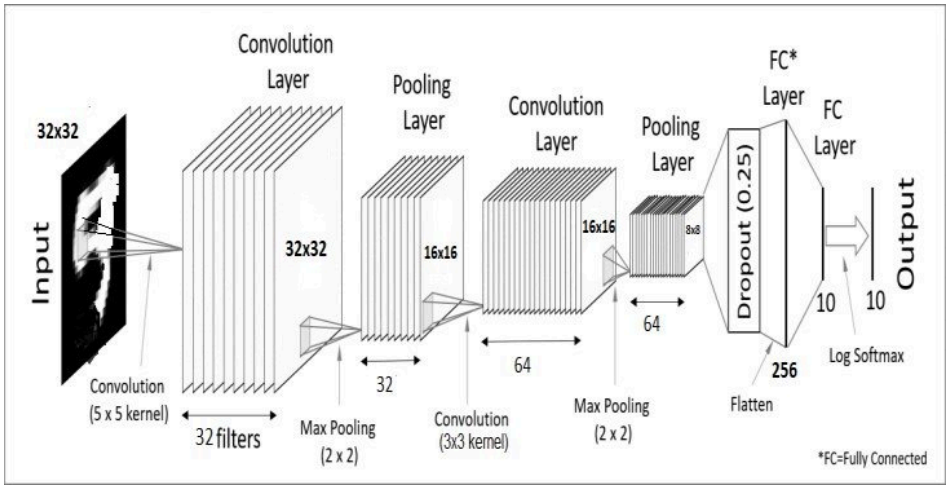
সারাংশ **(Summary)**

- জেনারালাইজেশন হল মডেলের নতুন, অদেখা ডেটার উপর ভালো পারফরম্যান্স করার ক্ষমতা।
- ওভারফিটিং ঘটে যখন মডেলটি প্রশিক্ষণ ডেটাকে পুরোপুরি মনে রেখে শিখে, যার ফলে মডেলটি সাধারণ নিদর্শনগুলি শিখতে না পেরে খারাপ জেনারালাইজেশন করে।
- স্টপিং ক্রাইটেরিয়া ওভারফিটিং প্রতিরোধ করতে সাহায্য করে, এবং প্রারম্ভিক স্টপিং হল সবচেয়ে সাধারণ পদ্ধতি।
- ভ্যালিডেশন ডেটা এবং রেগুলারাইজেশন পদ্ধতি যেমন ড্রপআউট, L1/L2 পেনাল্টি, এবং ডেটা অগমেন্টেশন ওভারফিটিং প্রতিরোধে গুরুত্বপূর্ণ ভূমিকা পালন করে এবং মডেলটিকে ভালোভাবে জেনারালাইজ করতে সাহায্য করে।
- এই কৌশলগুলি একত্রে কাজ করে মডেলের দৃঢ়তা বাড়িয়ে এবং নতুন ডেটার উপর সঠিক নিদর্শন শিখে জেনারালাইজেশন ক্ষমতা উন্নত করে।

হ্যান্ডরিটেন ডিজিট রিকগনিশন জন্য এএনএন আর্কিটেকচার **(ANN Architecture for Handwritten Digit Recognition)**

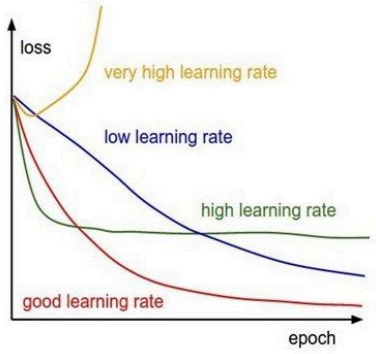
হ্যান্ডরিটেন ডিজিট রিকগনিশন এর জন্য একটি সাধারণ কৌশল হল একটি নেটওয়ার্ক ডিজাইন করা যার মধ্যে ইনপুট লেয়ার, এক বা একাধিক হিডেন লেয়ার এবং আউটপুট লেয়ার থাকে। একটি সরল উদাহরণ এখানে দেওয়া হল:

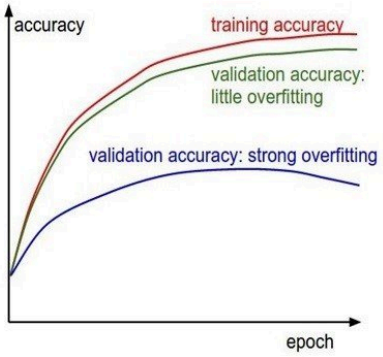




আমাদের প্রস্তাবিত সিস্টেমের CNN মডেলের সামগ্রিক কাঠামো বিভিন্ন লেয়ারের সাথে। কনভোলিউশনাল নিউরাল নেটওয়ার্ক (CNNs) ব্যবহার করে ডিজিট শনাক্তকরণের জন্য RMSprop ব্যবহার করা হয়েছে এবং এটি হিন্দি হ্যান্ডরিটেন নম্বর ডেটাসেটে প্রশিক্ষিত। যেখানে Keras API ব্যবহার করা হয়েছে এবং TensorFlow ব্যাকএন্ড হিসেবে ব্যবহৃত হয়েছে। মোট ৭টি লেয়ার ব্যবহার করা হয়েছে, যার মধ্যে প্রথম ও তৃতীয় লেয়ার কনভোলিউশনাল (Conv2D) লেয়ার, দ্বিতীয় ও চতুর্থ লেয়ার হল CNN এর গুরুত্বপূর্ণ লেয়ার—পুলিং (MaxPool2D) লেয়ার, তারপর Flatten লেয়ার এবং শেষ দুটি ফুলি কানেকটেড Dense লেয়ার, যা মূলত কৃত্রিম নিউরাল নেটওয়ার্ক (ANN) ক্লাসিফায়ার।

প্রথম লেয়ার হিসাবে কনভোলিউশনাল লেয়ার হিসেবে Conv2D ব্যবহার করা হয়েছে, যার মধ্যে প্রথম লেয়ারে ৩২টি লার্নেবল ফিল্টার এবং মডেলের শেষ লেয়ারে ৬৪টি ফিল্টার রয়েছে। প্রতিটি ফিল্টার একটি ছবির একটি অংশকে রূপান্তরিত করে, যা কের্নেল সাইজ দিয়ে কের্নেল ফিল্টার নির্ধারণ করে। কের্নেল ফিল্টার ম্যাট্রিক্সের সাইজ ৫x৫, যা পুরো ছবির ওপর প্রয়োগ করা হয়। ফিল্টারগুলোকে ছবির বৈশিষ্ট্য ম্যাপ হিসেবে দেখতে পাওয়া যায়। প্যাডিং কনসেপ্ট এখানে "same" হিসেবে ব্যবহার করা হয়েছে, যার মানে হল যে ইনপুটের সাথে আউটপুটের দৈর্ঘ্য সমান থাকবে, অর্থাৎ ইনপুটে প্যাডিং প্রয়োগ করা হয় যাতে আউটপুটের দৈর্ঘ্য মূল ইনপুটের মতো থাকে [8][11]। নিচের চিত্র ৪-এ প্যাডিং কিভাবে ঘটে তা দেখানো হয়েছে।





1. ইনপুট উপস্থাপন:

ইনপুট লেয়ার:

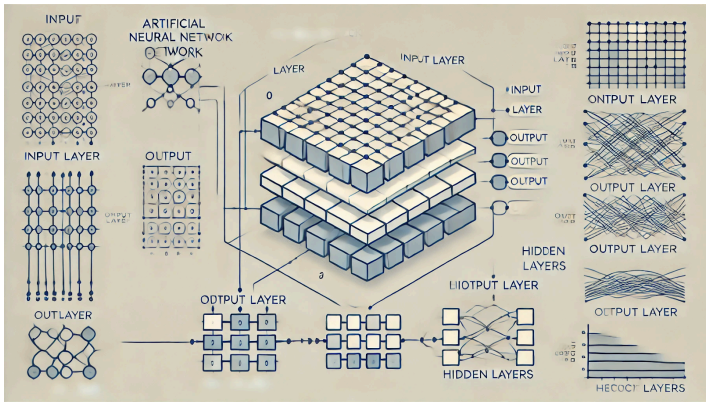
ইনপুট লেয়ার একটি ২৮x২৮ গ্রেস্কেল চিত্রের পিক্সেল মানগুলোকে উপস্থাপন করে (যেমন MNIST ডেটাসেটের ক্ষেত্রে)। প্রতিটি পিক্সেলের একটি গ্রেস্কেল মান থাকে যা ০ থেকে ২৫৫ এর মধ্যে থাকে, এবং এটিকে ০ থেকে ১ এর মধ্যে নর্মালাইজ করা যায়। ২৮x২৮ পিক্সেল হলে, ইনপুট লেয়ারে ৭৮৪টি নিউরন থাকে (প্রত্যেকটি পিক্সেলের জন্য একটি করে নিউরন)।

2. আউটপুট উপস্থাপন:

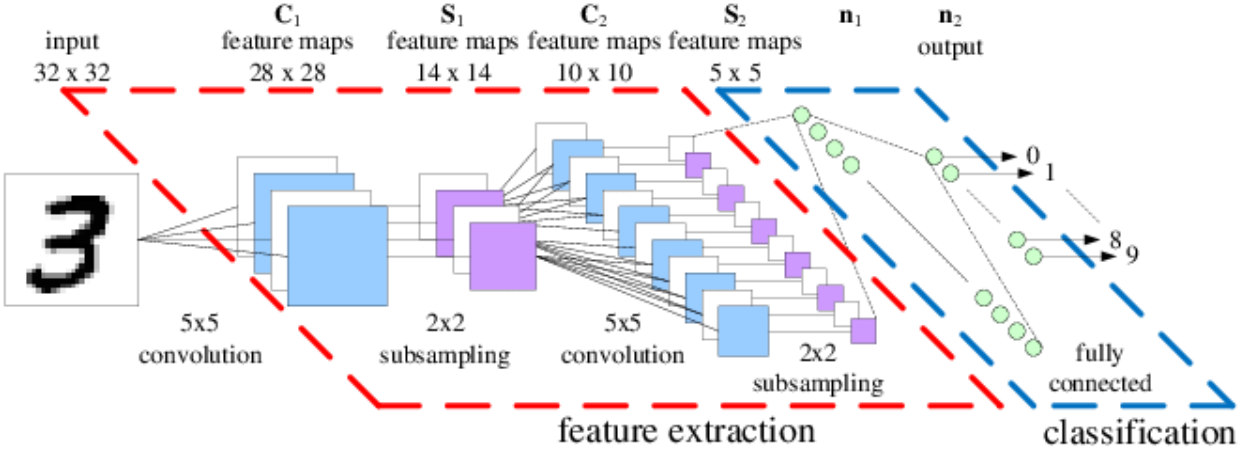
আউটপুট লেয়ার:

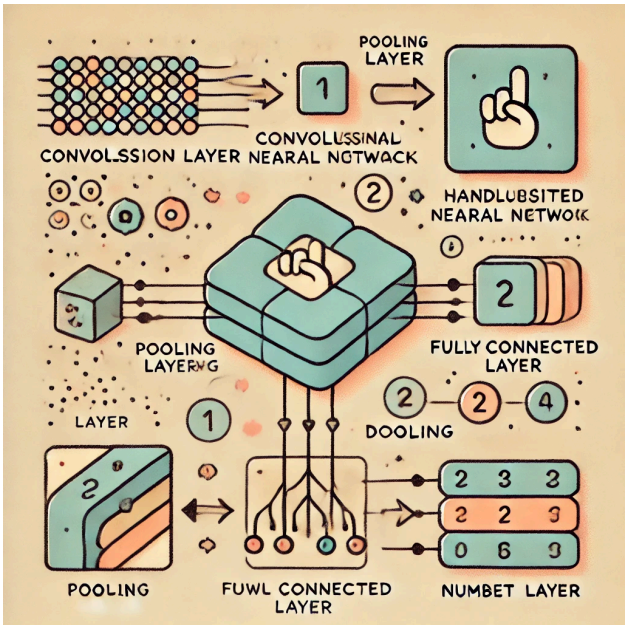
আউটপুট লেয়ারটি ০ থেকে ৯ পর্যন্ত ডিজিটগুলোকে উপস্থাপন করে। এই লেয়ারে ১০টি নিউরন থাকে, যেখানে প্রতিটি নিউরন একটি করে ডিজিটের সাথে সম্পর্কিত। আউটপুট লেয়ারে সাধারণত সফটম্যাক্স অ্যাকটিভেশন ফাংশন ব্যবহৃত হয়, যা প্রতিটি ক্লাস (০-৯) এর জন্য প্রোবাবিলিটি প্রদান করে, যেখানে সবচেয়ে উচ্চ প্রোবাবিলিটি সহ নিউরনটি প্রেডিক্টেড ডিজিট হিসেবে নির্দেশ করে।

3. নেটওয়ার্কের ব্লক ডায়াগ্রাম:
   এখানে হ্যান্ডরিটেন ডিজিট শনাক্তকরণের জন্য একটি সাধারণ ANN আর্কিটেকচারের ব্লক ডায়াগ্রাম দেখানো হল:



হস্তলিখিত অঙ্ক সনাক্তকরণ কাজের জন্য প্রয়োগিত **CNN** এর ব্লক ডায়াগ্রাম





1. স্বয়ংক্রিয় বৈশিষ্ট্য শিখনের প্রয়োজনীয়তা

ম্যানুয়াল বৈশিষ্ট্য নিষ্কাশন**:** প্রচলিত পদ্ধতিতে চিত্র বিশ্লেষণের জন্য বৈশিষ্ট্য নিষ্কাশন ম্যানুয়ালি করতে হত, যা শ্রমসাধ্য এবং সূক্ষ্ম প্যাটার্ন মিস হতে পারে।

স্বয়ংক্রিয় বৈশিষ্ট্য শিখন**:** CNNs স্বয়ংক্রিয়ভাবে গুরুত্বপূর্ণ বৈশিষ্ট্য (যেমন এজ, টেক্সচার, এবং শেপ) কাঁচা ডেটা থেকে শিখতে পারে। এর ফলে চিত্র শ্রেণীবিভাগের মতো জটিল কাজগুলোতে পারফরম্যান্স উন্নত হয়, পূর্বনির্ধারিত বৈশিষ্ট্য ছাড়াই।

2. Feed-Forward Neural Networks এবং **CNN-**এর মধ্যে পার্থক্য

Feed-Forward Neural Networks (FNNs): FNN-এ, প্রতিটি নিউরন পরবর্তী স্তরের সাথে পূর্ণরূপে সংযুক্ত থাকে। এই নেটওয়ার্কগুলি গঠনগত ডেটার জন্য ভাল কাজ করে কিন্তু চিত্রের মতো উচ্চ মাত্রার ডেটার জন্য সংগ্রাম করে, কারণ এতে প্রচুর পরিমাণে প্যারামিটার এবং গণনাগত খরচ হয়।

Convolutional Neural Networks (CNNs): CNNs কনভলিউশনাল এবং পুলিং লেয়ারগুলি প্রবর্তন করে, যা স্থানীয় অবস্থানে ওজন শেয়ার করে প্যারামিটারগুলির সংখ্যা কমায়। এর ফলে CNNs চিত্র এবং স্থানীয় ডেটার জন্য আরও দক্ষ এবং কার্যকর হয়, কারণ এটি প্যাটার্ন এবং স্থানীয় বৈশিষ্ট্যগুলো চিনতে সাহায্য করে।

3. CNN-এ কনভলিউশনাল লেয়ারের ভূমিকা

কনভলিউশনাল লেয়ার একটি বৈশিষ্ট্য নিষ্কাশক হিসেবে কাজ করে। ফিল্টার (ছোট ম্যাট্রিক্স) ইনপুট চিত্রের উপর দিয়ে স্লাইড করে, স্থানীয় প্যাটার্ন যেমন এজ, কর্নার বা টেক্সচার চিহ্নিত করে।

প্রতিটি ফিল্টার একটি বৈশিষ্ট্য ম্যাপ উৎপন্ন করে, যা চিহ্নিত বৈশিষ্ট্যগুলোকে গুরুত্ব দেয় এবং একাধিক ফিল্টার CNN-কে স্বয়ংক্রিয়ভাবে বিভিন্ন বৈশিষ্ট্য শিখতে সহায়তা করে।

4. 2D কনভলিউশনের একটি উদাহরণ

একটি 2D কনভলিউশনে:

- একটি ফিল্টার (যেমন ৩x৩ ম্যাট্রিক্স) চিত্র ম্যাট্রিক্সের উপর দিয়ে স্লাইড করে।

- প্রতিটি ফিল্টার অবস্থান চিত্রের উপাদানগুলির সাথে ফিল্টারের সংশ্লিষ্ট উপাদানগুলির গুণফল নিয়ে, তারপর ফলাফলগুলির যোগফল করে একটি নতুন মান উৎপন্ন করে।

- এর ফলে একটি বৈশিষ্ট্য ম্যাপ উৎপন্ন হয়, যা চিত্রে নির্দিষ্ট প্যাটার্নগুলিকে হাইলাইট করে।
  এই প্রক্রিয়া CNN-কে স্থানীয় সম্পর্ক বজায় রাখতে এবং চিত্রের বিভিন্ন অবস্থানে আকার বা টেক্সচার চিনতে সাহায্য করে।

5. পুলিং লেয়ারের কার্যাবলী

পুলিং লেয়ার বৈশিষ্ট্য ম্যাপগুলির স্থানীয় মাত্রা কমিয়ে দেয়, যার ফলে গণনাগত লোড কমে যায় এবং পরিবর্তনের প্রতি রোবস্টনেস (দ্রুত প্রতিক্রিয়া ক্ষমতা) উন্নত হয়।
সাধারণভাবে ব্যবহৃত পদ্ধতিগুলি হল:

- ম্যাক্স পুলিং**:** প্রতিটি অঞ্চলের সর্বোচ্চ মান নির্বাচন করে।

- এভারেজ পুলিং**:** প্রতিটি অঞ্চলের গড় মান নিয়ে আসে।
  এটি CNN-কে সবচেয়ে গুরুত্বপূর্ণ বৈশিষ্ট্যগুলির উপর মনোযোগ দিতে সহায়তা করে, শব্দ বা অপ্রয়োজনীয় তথ্য ছাড়াই এবং গুরুত্বপূর্ণ তথ্য বজায় রাখে।

এই ধারণাগুলি একত্রিতভাবে CNN-কে জটিল, উচ্চ মাত্রার ডেটা যেমন চিত্র প্রক্রিয়া করতে সক্ষম করে, যা চিত্র চিনহিতকরণ, অবজেক্ট ডিটেকশন, এবং আরও অনেক ক্ষেত্রে মৌলিক ভূমিকা পালন করে।

অনুশীলনী**:-**

[নম্বর**- ২]**

1. এআইএনএস (ANNs)-এর পেছনে জীববৈজ্ঞানিক অনুপ্রেরণা কী?

2. ম্যাককুলচ এবং পিটসের নিউরন মডেল বর্ণনা করুন।

3. একটি অ্যাকটিভেশন ফাংশন কী এবং এর গুরুত্ব কী?

4. পারসেপ্ট্রন কী, এবং এটি কীভাবে একটি লিনিয়ার ক্লাসিফায়ার হিসেবে ব্যবহৃত হয়?

5. পারসেপ্ট্রন ট্রেনিংয়ের জন্য ডেল্টা রুলটি সংক্ষেপে বর্ণনা করুন।

6. নিউরাল নেটওয়ার্কে অ-রৈখিকতা কেন অপরিহার্য?

7. ফিড-ফরওয়ার্ড নেটওয়ার্ক এবং রিকারেন্ট নেটওয়ার্কের মধ্যে পার্থক্য কী?

8. পারসেপ্ট্রন লার্নিং-এ XOR সমস্যা কী?

9. নিউরাল নেটওয়ার্কে হিডেন লেয়ারের প্রয়োজন কী?

10. CNN-এ পুলিংয়ের ভূমিকা কী?

11. CNN-এ কনভলিউশনাল লেয়ার কীভাবে কাজ করে?

12. ওভারফিটিং কী, এবং এটি নিউরাল নেটওয়ার্কে কীভাবে প্রভাব ফেলে?

13. CNN এবং প্রচলিত ফিড-ফরওয়ার্ড নেটওয়ার্কের মধ্যে পার্থক্য কী?

14. নিউরাল নেটওয়ার্ক প্রশিক্ষণের ক্ষেত্রে আর্লি স্টপিং (early stopping)-এর ধারণা কী?

15. ব্যাকপ্রপাগেশন কীভাবে কাজ করে?

16. সিগময়েড অ্যাকটিভেশন ফাংশন কী, এবং এর রেঞ্জ কী?

17. হস্তলিখিত ডিজিট সনাক্তকরণের জন্য একটি CNN-এর মৌলিক কাঠামো কী?

18. লিনিয়ার সেপারেটর কী, এবং পারসেপ্ট্রন লার্নিংয়ে এটি কীভাবে প্রযোজ্য?

19. CNN-এ স্বয়ংক্রিয় বৈশিষ্ট্য শিখনের ধারণা কী?

20. একটি একক স্তরের পারসেপ্ট্রন ব্যবহার করে অ-রৈখিক সমস্যাগুলি সমাধান করার প্রধান চ্যালেঞ্জ কী?

[নম্বর- ৩]

1. কৃত্রিম নিউরাল নেটওয়ার্ক (ANN)-এর জীববৈজ্ঞানিক প্রেরণা কী? McCulloch এবং Pitts (1943)-এর নিউরন মডেল সংক্ষেপে বর্ণনা করো।

2. জৈবিক নিউরন এবং কৃত্রিম নিউরনের মধ্যে সাদৃশ্য ও বৈসাদৃশ্য ব্যাখ্যা করো।

3. ANN-এ ‘weighted sum’ এবং ‘activation function’—এই দুটি ধারণা ব্যাখ্যা করো।

4. Threshold activation function এবং Sigmoid activation function এর মধ্যে পার্থক্য ব্যাখ্যা করো।

5. ANN-এ activation function কেন প্রয়োজন? এর অনুপস্থিতিতে কী সমস্যা হয়?

6. Sigmoid ফাংশন কীভাবে আউটপুটকে (0,1) রেঞ্জে রূপান্তর করে—বুঝিয়ে লিখো।

7. Perceptron কী? এটিকে কেন একটি linear classifier বলা হয়?

8. Perceptron training rule-এর ধারণা ব্যাখ্যা করো।

9. দুই ইনপুট বিশিষ্ট AND ও OR ফাংশন Threshold Perceptron দিয়ে কীভাবে প্রকাশ করা যায়—ব্যাখ্যা করো।

10. Input space-এ linear separator-এর সমীকরণ লিখে ব্যাখ্যা করো।

11. Perceptron-এর representational power কী? একটি উদাহরণ সহ ব্যাখ্যা করো।

12. XOR সমস্যা percepton দিয়ে সমাধান না হওয়ার কারণ ব্যাখ্যা করো।

13. Unthresholded perceptron-এর Delta rule-এর মূল ধারণা ব্যাখ্যা করো (গাণিতিক বিশদ নয়)।

14. Hidden layer কেন প্রয়োজন? একটি উপযুক্ত উদাহরণসহ ব্যাখ্যা করো।

15. ANN-এ non-linearity কেন অপরিহার্য? উদাহরণসহ ব্যাখ্যা করো।

16. Feed-forward এবং Recurrent neural network-এর মৌলিক ধারণা ও পার্থক্য লিখো।

17. Feed-forward network-এ Forward pass কীভাবে কাজ করে—সংক্ষেপে ব্যাখ্যা করো।

18. Backpropagation-এর মূল ধারণা কী? এটি কীভাবে ওজন (weights) আপডেট করতে সাহায্য করে?

19. Multilayer feed-forward network ট্রেনিং-এ Backpropagation কেন অত্যাবশ্যক—ব্যাখ্যা করো।

20. Overfitting কী? ANN-এ কেন ঘটে?

21. Validation dataset-এর ভূমিকা ব্যাখ্যা করো।

22. ANN-এ generalization ক্ষমতা বাড়ানোর দুইটি উপায় ব্যাখ্যা করো।

23. Handwritten digit recognition-এ ANN-এর ইনপুট representation কীভাবে তৈরি হয়?

24. হ্যান্ডরাইটেন ডিজিট রিকগনিশনে আউটপুট representation (0–9) কীভাবে গঠন করা হয়—ব্যাখ্যা করো।

25. একটি সাধারণ ANN ভিত্তিক digit recognition system-এর ব্লক ডায়াগ্রাম সংক্ষেপে ব্যাখ্যা করো।

26. Automatic feature learning কেন গুরুত্বপূর্ণ? Conventional ANN এবং CNN-এর পার্থক্য লেখো।

27. Convolution layer-এর ভূমিকা ব্যাখ্যা করো।

28. 2D convolution কী? সহজ উদাহরণসহ ব্যাখ্যা করো (গাণিতিক ধাপ ছাড়াই)।

29. Pooling layer-এর কাজ কী? Max pooling-এর একটি উদাহরণ দাও।

30. Handwritten digit recognition-এ CNN ব্যবহারের একটি ব্লক ডায়াগ্রাম নিয়ে সংক্ষেপে আলোচনা করো।

[নম্বর- ৫]

1. কৃত্রিম নিউরাল নেটওয়ার্কের জন্য জীববৈজ্ঞানিক অনুপ্রেরণা ব্যাখ্যা করুন এবং স্বয়ংক্রিয় বৈশিষ্ট্য শিখনের গুরুত্ব আলোচনা করুন।

2. ম্যাককুলচ এবং পিটসের নিউরন মডেল বর্ণনা করুন, যার মধ্যে অ্যাকটিভেশন ফাংশনের ধারণা এবং উদাহরণসহ ব্যাখ্যা করুন।

3. পারসেপ্ট্রন ট্রেনিং সংজ্ঞায়িত করুন এবং পারসেপ্ট্রন ট্রেনিং রুল কীভাবে প্রয়োগ হয় তা উদাহরণসহ ব্যাখ্যা করুন।

4. একটি অ-থ্রেশোল্ডেড পারসেপ্ট্রন ট্রেনিং-এর জন্য ডেল্টা রুল নির্দিষ্ট করুন এবং এর কার্যাবলী ব্যাখ্যা করুন যা ওজনের সমন্বয়ে ব্যবহৃত হয়।

5. নিউরাল নেটওয়ার্কে অ-রৈখিকতা কেন প্রয়োজন, এবং ফিড-ফরওয়ার্ড এবং রিকারেন্ট নেটওয়ার্ক কাঠামোর মধ্যে পার্থক্য কী?

6. ব্যাকপ্রপাগেশন অ্যালগরিদম বর্ণনা করুন এবং এটি কীভাবে মাল্টি-লেয়ার ফিড-ফরওয়ার্ড নিউরাল নেটওয়ার্ক ট্রেনিং-এ ব্যবহৃত হয় তা ব্যাখ্যা করুন।

7. নিউরাল নেটওয়ার্কে ওভারফিটিং কী, এবং এটি কাটিয়ে উঠতে কী কী কৌশল রয়েছে, যেমন ভ্যালিডেশন ডেটাসেট ব্যবহার করা?

8. উদাহরণসহ পারসেপ্ট্রনের প্রতিনিধিত্ব ক্ষমতা ব্যাখ্যা করুন, যেমন AND এবং OR লজিক্যাল ফাংশন বাস্তবায়ন করা।

9. হস্তলিখিত ডিজিট সনাক্তকরণের জন্য একটি CNN কাঠামো আঁকুন এবং ইনপুট ও আউটপুট উপস্থাপনাগুলি বর্ণনা করুন।

10. CNN এবং প্রচলিত ফিড-ফরওয়ার্ড নিউরাল নেটওয়ার্কের মধ্যে পার্থক্য ব্যাখ্যা করুন, বিশেষ করে CNN-এ কনভলিউশন এবং পুলিং লেয়ারের ভূমিকা কি?

SEMESTER IV - UNIT 6

কৃত্রিম বুদ্ধিমত্তায় নৈতিক বিষয়সমূহ

কৃত্রিম বুদ্ধিমত্তা (Artificial Intelligencel) মানুষের চিন্তা, শেখা ও সিদ্ধান্ত গ্রহণের ক্ষমতাকে অনুকরণ করে কাজ করে। যদিও এটি আমাদের জীবনকে সহজ ও কার্যকর করছে, তবুও এর ব্যবহারের সঙ্গে যুক্ত রয়েছে কিছু গুরুত্বপূর্ণ নৈতিক (Ethical) সমস্যা।

নৈতিকতার অর্থ:

নৈতিকতা (Ethics) বলতে এমন নীতি বা মানদণ্ডকে বোঝায় যা আমাদের সঠিক ও ভুল কাজের মধ্যে পার্থক্য করতে সাহায্য করে। AI ব্যবহারে নৈতিকতার মূল লক্ষ্য হলো — প্রযুক্তির ব্যবহার যেন মানবকল্যাণে হয়, ক্ষতির জন্য নয়।

কৃত্রিম বুদ্ধিমত্তায় গুরুত্বপূর্ণ নৈতিক বিষয়সমূহ:

- গোপনীয়তা **(Privacy)**

AI সিস্টেম বিপুল পরিমাণ ব্যক্তিগত তথ্য সংগ্রহ ও বিশ্লেষণ করে। এই তথ্য ভুলভাবে ব্যবহার বা ফাঁস হলে ব্যক্তির গোপনীয়তা নষ্ট হতে পারে। উদাহরণঃ ফেস রিকগনিশন সিস্টেমে ব্যক্তিগত ছবি অনুমতি ছাড়াই ব্যবহার।

- পক্ষপাতদুষ্টতা **(Bias)** ও বৈষম্য **(Discrimination)**

AI সিস্টেম যদি পক্ষপাতপূর্ণ তথ্য দিয়ে প্রশিক্ষিত হয়, তাহলে সেটি অন্যায় সিদ্ধান্ত নিতে পারে। উদাহরণঃ চাকরি বা ঋণ অনুমোদনে লিঙ্গ বা জাতিগত পক্ষপাত। সমাধানঃ নিরপেক্ষ ও বৈচিত্র্যময় ডেটা ব্যবহার করা।

- স্বচ্ছতা **(Transparency)**

AI-এর সিদ্ধান্ত গ্রহণ প্রক্রিয়া অনেক সময় বোঝা কঠিন হয়, যাকে বলে “Black Box Problem।” সমাধানঃ Explainable AI (XAI) ব্যবহার করা উচিত।

- দায়বদ্ধতা **(Accountability)**

AI যদি কোনো ভুল সিদ্ধান্ত নেয়, তবে দায়ী কে হবে — প্রোগ্রামার, ব্যবহারকারী, না মেশিন? সমাধানঃ স্পষ্ট নীতি ও আইন প্রণয়ন করা।

- নিরাপত্তা **(Security)**

AI সিস্টেম হ্যাক বা অপব্যবহার করা গেলে বড় ধরনের ক্ষতি হতে পারে। উদাহরণঃ স্বচালিত গাড়ির হ্যাকিং। সমাধানঃ সাইবার নিরাপত্তা জোরদার করা।

- চাকরির ক্ষতি **(Job Displacement)**

AI স্বয়ংক্রিয়ভাবে কাজ করতে পারে, ফলে অনেক মানুষের কর্মসংস্থান হুমকির মুখে পড়ে। সমাধানঃ নতুন দক্ষতা অর্জনের জন্য প্রশিক্ষণ ও পুনঃশিক্ষা (Reskilling) প্রয়োজন।

- মানবনিয়ন্ত্রণের অভাব **(Loss of Human Control)**

AI যদি অত্যধিক স্বাধীনতা পায়, তবে মানুষের নিয়ন্ত্রণ কমে যেতে পারে। উদাহরণঃ যুদ্ধক্ষেত্রে স্বয়ংক্রিয় রোবট বা ড্রোনের ব্যবহার। সমাধানঃ “Human-in-the-loop” ব্যবস্থা বজায় রাখা।

উপসংহার:

AI আমাদের সমাজে অপরিসীম সুবিধা এনে দিতে পারে, তবে তা নৈতিকভাবে ব্যবহার করাই সবচেয়ে গুরুত্বপূর্ণ। নৈতিক নীতিমালা মেনে চললে AI প্রযুক্তি মানবকল্যাণে সহায়ক হবে এবং একটি দায়িত্বশীল ও নিরাপদ ভবিষ্যৎ গড়ে তুলতে পারবে।

অনুশীলনী: [নম্বর- ৩]

1. কৃত্রিম বুদ্ধিমত্তায় নৈতিকতা (Ethics) বলতে কী বোঝায়? উদাহরণসহ ব্যাখ্যা করো।

2. কৃত্রিম বুদ্ধিমত্তায় গোপনীয়তা (Privacy) সমস্যা কীভাবে দেখা দেয়? একটি উদাহরণ দাও।

3. পক্ষপাতদুষ্টতা (Bias) কী? এটি AI সিস্টেমে কীভাবে প্রভাব ফেলে?

4. “Explainable AI” (XAI) বলতে কী বোঝায়? এর প্রয়োজনীয়তা সংক্ষেপে লেখো।

5. দায়বদ্ধতা (Accountability) সম্পর্কিত নৈতিক সমস্যা ব্যাখ্যা করো। AI-এর ভুল সিদ্ধান্তের ক্ষেত্রে দায়ী কে হবে?

6. কৃত্রিম বুদ্ধিমত্তার নিরাপত্তা (Security) সংক্রান্ত নৈতিক চ্যালেঞ্জগুলি উল্লেখ করো।

7. AI-এর কারণে চাকরির ক্ষতি (Job Displacement) কেন একটি নৈতিক সমস্যা হিসেবে বিবেচিত হয়? এর সমাধান কী হতে পারে?

8. মানবনিয়ন্ত্রণের অভাব (Loss of Human Control) কীভাবে সমাজের জন্য বিপজ্জনক হতে পারে? সংক্ষেপে ব্যাখ্যা করো।

9. কৃত্রিম বুদ্ধিমত্তা ব্যবহারে স্বচ্ছতা (Transparency) কেন গুরুত্বপূর্ণ?

10. AI প্রযুক্তি ব্যবহারে নৈতিক নীতিমালা অনুসরণের প্রয়োজনীয়তা ব্যাখ্যা করো।