WEEK 8 NPTEL: AI

This lecture introduces the concept of learning in artificial intelligence, defining it as an agent improving its performance on future tasks after observing the world. It covers various aspects of learning, including:

- **What is Learning:** Improving performance on future tasks based on observations, and learning a function from input-output pairs to predict new outputs. It mentions linear, nonlinear (neural networks), nonparametric, and support vector machine models, and the benefit of model ensembles.
- **Types of Learning:** Categorized by what to learn from, what data, and prior knowledge.
- **Components to be Learnt:** Direct mapping from conditions to actions, inferring world properties, information on world evolution and agent actions, utility and action-value information, and goals.
- **Representation and Prior Knowledge:** Discusses propositions, first-order logic, Bayesian networks, and factored representations. It distinguishes between inductive learning (from example input-output pairs) and deductive learning (from known rules to new rules).
- **Learning According to Feedback:**
    - **Unsupervised learning:** Learns patterns without explicit feedback (e.g., clustering).
    - **Supervised learning:** Learns a mapping from input to output given example input-output pairs.
    - **Reinforcement learning:** Learns from a series of rewards or punishments.
    - **Semi-supervised learning:** Uses a few labeled and many unlabeled examples.
- **Supervised Learning (detailed):**
    - Aims to discover a function h that approximates an unknown function y=f(x) from a training set $(x_1,y_1),...,(x□,y□)$.
    - **Classification:** Output y is one of a finite set of values (Boolean/binary if two values).
    - **Regression:** Output y is a real number.
    - Learning is a search for a hypothesis h that performs well on new examples.
    - Accuracy is measured using a test set distinct from the training set; good generalization means correctly predicting y for novel examples.
    - Can involve learning a conditional probability distribution P(Y|X) if f is stochastic.
    - The optimal hypothesis h* is chosen based on P(h|data), equivalent to P(data|h)P(h) by Bayes' rule.
- **Classification:** Assigning categories or classes to items, used to predict group membership. Can be supervised (training data available) or unsupervised (no training data).
- **Learning Techniques (for classification):** Decision trees, rule-based systems, nearest neighbor classifiers, Bayes classifier, Support Vector Machines (SVMs), Neural Networks.
- **Evaluation of Classifiers:**
    - **Accuracy:** Ability to correctly classify unknown patterns.
    - **Design time and classification time:** Time to build the classifier vs. time to classify a pattern.
    - **Space required:** Amount of storage needed, depending on whether an abstraction of the training set is used.
    - **Explanation ability:** Clarity of why a classifier chose a particular class.
    - **Noise tolerance:** Ability to handle outliers and wrongly classified patterns.
- **Validation Methods:** To estimate classifier performance.

    - **Re-substitution estimate:** Uses the training set itself (assumes it's representative).
    - **Holdout method:** Divides the training set into training and validation subsets (e.g., two-thirds for training, one-third for validation).
    - **Random sub-sampling:** Repeats the holdout method multiple times with different subsets and averages the accuracy.
    - **Cross-validation:** Each pattern is used for training multiple times and for testing exactly once (e.g., k-fold cross-validation, leave-one-out).
    - **Bootstrap procedure:** Randomly samples patterns with replacement for training, and uses unchosen patterns for testing.
- **Evaluation Metrics:** Includes classification accuracy (CA), precision (p), and recall (r), defined by true positives (TP), false positives (FP), and false negatives (FN).

Lecture 37
This file, "lecture37.pdf," covers several algorithms for classification, including:

- **Non-parametric models:** These models are not characterized by a bounded set of parameters and include instance-based learning methods like table lookup and nearest neighbor classification.
- **Nearest Neighbor (NN) Classification:**
    - Classifies a new pattern based on the class label of its closest neighbor.
    - Distance metrics like Minkowski (Euclidean and Manhattan), Hamming, and Mahalanobis distances are used.
    - Variations include k-Nearest Neighbor (kNN), Modified kNN (mkNN), and r-Near Neighbors.
- **Rule-Based Classifiers:**
    - Use a set of IF-THEN rules for classification.
    - Rules can be mutually exclusive (each record covered by at most one rule) and exhaustive (each record covered by at least one rule).
    - Rules can be ordered by priority (rule-based or class-based).
    - Advantages include expressiveness, ease of interpretation and generation, rapid classification, and performance comparable to decision trees.
- **Support Vector Machine (SVM):**
    - A popular supervised learning approach that constructs a maximum margin separator.
    - Uses the "kernel trick" to embed data into a higher-dimensional space, allowing for non-linear separation in the original space.
    - Combines advantages of non-parametric and parametric models.
    - The optimization problem aims to minimize the squared norm of the weight vector subject to correct classification.
- **Bayes Classifier:**
    - Uses prior probabilities of classes and the distribution of patterns in each class to predict a new pattern's class.
    - Aims for minimum error rate by maximizing the posterior probability P(C|Y).
- **Naïve Bayes Classifier:**
    - Assumes each feature is class conditionally independent.
    - The posterior probability for a pattern is proportional to the prior probability of the class multiplied by the product of the probabilities of each feature given the class.

WEEK 8 NPTEL: AI

Lecture 38
This document, "Lecture 38: Learning algorithms - 2," focuses on decision tree classifiers.

Here's a summary of the key points:

- **Decision Tree Definition**: A decision tree is a rooted tree where internal nodes represent conditions on features, and outgoing edges represent outcomes. Leaf nodes provide class labels. Each path from the root to a leaf forms a rule with the leaf node's class label as the consequent.
- **Example of a Decision Tree**: An example of finding a heavier coin among four (a, b, c, d) illustrates how decisions are made at internal nodes and outcomes are found at leaf nodes.
- **Decision Trees for Pattern Classification**: The document provides an example of classifying animals based on features like "Legs," "Horns," "Size," "Colour," "Beak," and "Sound," demonstrating how a decision tree can categorize different animals.
- **Observations about Decision Trees**: Key characteristics include class labels at leaf nodes, rules represented by root-to-leaf paths, decisions at internal nodes, the exclusion of irrelevant features, and the ability to handle both numerical and categorical features. Trees can be binary or non-binary, and the rules are easy to understand.
- **Construction of Decision Trees**: Trees are induced by examples, and at each node, the "best" attribute is chosen to split the data. The goal is for each branch to have either all positive or all negative examples.
- **Measures of Impurity**:
    - **Entropy or Information Impurity**: Quantifies the impurity at a node, with higher values indicating more mixed classes and 0 indicating a pure node.
    - **Variance Impurity (Gini Impurity)**: Another measure of impurity, particularly for more than two categories.
    - **Misclassification Impurity**: Measures the probability of misclassifying a pattern at a node.
- **Attribute Selection**: The attribute that provides the maximum "drop in impurity" (or "Gain") is chosen for splitting at a node. The document provides calculations for drop in impurity using entropy, variance, and misclassification impurity.
- **Types of Splits**:
    - **Axis Parallel Split**: Decisions involve only one feature (e.g., `f_i > c`).
    - **Oblique Split**: Decisions involve more than one feature (e.g., `a.af_1 + bf_2 + c > 0`).
    - **Multivariate Split**: Uses a nonlinear combination of features for decision making.
- **Information Gain**: This concept is explained using a "PlayTennis" example.
    - **Entropy (Ent(S))**: Measures the impurity of a training set.
    - **Information Gain (Gain(S, A))**: Quantifies the reduction in entropy when a dataset is split by an attribute A. The attribute with the maximum information gain is considered the most informative. The document walks through calculating information gain for attributes like "Outlook," "Humidity," "Wind," and "Temperature."
- **Continuing to Split**: The process of selecting attributes and splitting continues until leaf nodes represent a single class or a predefined stopping criterion is met.
- **Final Decision Tree Example**: A complete decision tree for the "PlayTennis" example is

presented, illustrating how the various attributes lead to a "Yes" or "No" outcome for playing tennis.

Lecture 39
This lecture document on Ensemble Learning covers the following key areas:

1. Broadening Applicability of Decision Trees:

- Addresses issues like missing data, multi-valued attributes, continuous and integer-valued input attributes, and continuous-valued output attributes (regression trees).
- Highlights the importance of handling continuous variables in real-world applications.
- Emphasizes the interpretability of decision trees, especially for financial decisions subject to anti-discrimination laws.
- Mentions common decision tree induction packages like ID3, C4.5, SLIQ, and SPRINT.

2. Decision Trees - Pros and Cons:

- **Advantages:** Inexpensive to construct, fast at classifying, easy to interpret, good accuracy, handles both categorical and numerical attributes.
- **Disadvantages:** High training time, susceptible to overfitting/underfitting, can be large (requiring pruning), and primarily handles rectangular regions (oblique trees as a solution).

3. Ensemble Learning:

- Introduces the concept of combining multiple hypotheses (an "ensemble") to improve prediction accuracy.
- Explains that misclassification by an ensemble using majority voting is less likely than by a single hypothesis, especially if errors are somewhat independent.
- Describes ensemble learning as a way to enlarge the hypothesis space and learn more expressive classes of hypotheses without significant computational overhead.

4. Boosting and ADABOOST:

- Explains boosting as a widely used ensemble method that uses weighted training sets.
- The process involves generating hypotheses sequentially, increasing the weights of misclassified examples and decreasing the weights of correctly classified ones.
- The final ensemble hypothesis is a weighted-majority combination of all generated hypotheses.
- Highlights ADABOOST as a specific boosting algorithm with the property that it can perfectly classify training data for a "weak learning algorithm" (one slightly better than random guessing) given enough hypotheses.

5. Methods for Constructing Ensembles of Classifiers:

- **Sub-sampling the training examples:** Includes Bagging (bootstrap aggregation) and leaving out disjoint subsets.
- **Manipulating the input features:** Involves taking a subset of features, effective when

features are redundant.
- **Manipulating the output targets:** Known as error correcting output coding, where classes are partitioned, and re-labeled data is used to train classifiers.
- **Injecting randomness:** Introduces randomness into the learning algorithm, e.g., by randomly choosing the order of features in decision trees.

6. Methods for Combining Classifiers:

- The simplest approach is an unweighted vote.
- A more refined method involves combining class probability estimates from individual classifiers, where the predicted class is the one with the highest combined probability.

Lecture 40
This file, "lecture40.pdf," provides an introduction to Artificial Neural Networks (ANNs).

Here's a summary of the key points:

- **Definition and Inspiration:** ANNs are computational methods inspired by the brain and nervous systems of biological organisms. They are made of simple, highly interconnected processing elements that process information through their dynamic state response to external input.
- **Why ANNs?** Traditional computers excel at fast arithmetic and explicit programming, but struggle with noisy data, massive parallelism, fault tolerance, and adapting to circumstances. ANNs offer a solution when algorithmic solutions are hard to formulate or when many examples of desired behavior are available. They are based on the parallel architecture of biological brains.
- **Biological Neuron Analogy:** The document explains the structure and function of biological neurons, including dendrites (inputs), soma/cell body (processing), axon (output), and synapses (connections). It details how electrochemical signals are received, processed, and transmitted.
- **Artificial Neuron Structure:** An artificial neuron (or node) receives multiple inputs, multiplies each by a weight, sums the weighted inputs, applies an activation function to the sum, and produces an output. Key components are weights, thresholds, and an activation function.
- **Activation Functions:** Various activation functions are discussed, including the threshold (or sign) function, sigmoid function, and step function, which introduce non-linearity to the neuron's output.
- **ANN Operation:** ANNs map a particular input to a particular output. Their output depends on the input and the weights of the connections between neurons.
- **Training ANNs:** Training involves updating the weights in the network so that it produces the correct output for every input. If the output is wrong, the error is calculated and used to adjust the weights. This learning process, often involving multiple "epochs" (runs through all training examples), aims to minimize error on unseen test data.
- **Types of Neural Networks:** The lecture briefly mentions some types of neural networks, including Hopfield networks, Boltzmann machines, perceptrons, multilayer perceptrons, backpropagation, and Kohonen Feature Maps.
- **Comparison to Human Brain:** A comparison highlights the human brain's massively

parallel computing at a slower speed per unit (10^-3 seconds for neurons) compared to a computer's serial or limited parallel computing at much faster speeds (10^-8 seconds for CPU).