**Machine Learning (MTCS 031)**

**UNIT-IV**

**SYLLABUS:**

***COMPUTATIONAL LEARNING THEORY*** – Sample Complexity for Finite Hypothesis spaces, Sample Complexity for Infinite Hypothesis spaces, The Mistake Bound Model of Learning;

***INSTANCE-BASED LEARNING*** – k-Nearest Neighbor Learning, Locally Weighted Regression, Radial basis function networks, Case-based learning.

**PYQ:**

**7 MARKS:**

* Discuss various Mistake Bound Models of learning.
* Compare Classification with Regression with an example.
* Distinguish between supervised learning and unsupervised learning. Illustrate with an example.
* Write short notes on:

1. K-Nearest Neighbor Learning.

2. Radial basis function networks.

* Discuss the major drawbacks of K-nearest Neighbour learning Algorithm and how it can be corrected?
* Explain Locally Weighted Linear Regression.
* Define the following terms

a. Sample error

b. True error

c. Random Variable

d. Expected value

e. Variance.

**2 MARKS:**

* Explain the term linear regression.
* Explain the problem of crowding in Genetic Algorithm?

**Mistake Bound Models**

* It is a theoritical framework in ML that analyze how well an algo can learn from a sequence of example.
* This model emphasizes the no. of mistakes an algo makes before converging to the correct hypothesis.
* ***Mistake :-*** A mistake occurs when the algo`s prediction does not match the true label of the instance.
* ***Goal :***- To minimize the total no. of mistakes over all instances.
* ***Mistake bound :***- It is the theoritical upper limit on the no. of mistakes an algo will make . It is a function of the complexity of the hypothesis space and the algo`s strategy.

**Working:-**

* **Initial state :-** The algo starts without any prior knowledge of the traget function.
* **Prediction & feedback loop :-** for each incoming instance ,the algo make a prediction and then receive the true label.
* **Update rule :-** After each mistake ,the algo updates its hypothesis to reduce future mistakes.
* **Termination :-**  The process continues untill the algo consistently makes correct predictions.

**Major Drawbacks of K-Nearest Neighbors (K-NN) Algorithm and Corrections**

1. **Computational Complexity**
   * **Drawback**: High cost in calculating distances for large datasets.
   * **Correction**: Use KD-trees, Ball Trees, or Approximate Nearest Neighbors (e.g., Locality-Sensitive Hashing).
2. **High Memory Usage**
   * **Drawback**: Requires storing the entire training dataset in memory.
   * **Correction**: Apply dimensionality reduction (PCA) or data compression techniques.
3. **Poor Performance with High-Dimensional Data**
   * **Drawback**: Distance metrics become less meaningful in high-dimensional spaces.
   * **Correction**: Perform feature selection or dimensionality reduction.
4. **Sensitive to Noise and Outliers**
   * **Drawback**: Noisy data and outliers affect distance calculations.
   * **Correction**: Use outlier detection methods and robust distance metrics.
5. **Choice of K (Number of Neighbors)**
   * **Drawback**: Performance depends on selecting the right K; too small or too large can be problematic.
   * **Correction**: Use cross-validation to find the optimal K or implement adaptive methods.
6. **Scalability Issues**
   * **Drawback**: Performance degrades with very large datasets.
   * **Correction**: Use approximate algorithms or incremental learning techniques.

### ****Classification vs. Regression****

#### ****1. Purpose****

* **Classification**: Predicts categorical labels or classes. The output is a discrete label.
* **Regression**: Predicts continuous values. The output is a continuous quantity.

#### ****2. Output****

* **Classification**: The result is one of several predefined categories (e.g., "spam" or "not spam").
* **Regression**: The result is a real number (e.g., temperature in degrees Celsius).

#### ****3. Example****

* **Classification Example**: Email Spam Detection
  + ***Task***: Determine whether an email is "spam" or "not spam."
  + ***Output***: Categorical labels ("spam" or "not spam").
  + ***Model***: A classification model, such as a decision tree or logistic regression, would classify emails into these categories based on features like email content, sender, and subject line.
* **Regression Example**: Predicting House Prices
  + ***Task***: Predict the price of a house based on its features like size, number of bedrooms, and location.
  + ***Output***: A continuous value representing the house price.
  + ***Model***: A regression model, such as linear regression or polynomial regression, would predict the price of the house based on these features.

#### ****4. Evaluation Metrics****

* **Classification**:
  + **Accuracy**: Percentage of correctly classified instances.
  + **Precision, Recall, F1-Score**: Metrics for evaluating the performance of classification models, especially in imbalanced datasets.
  + **Confusion Matrix**: A table used to describe the performance of a classification model.
* **Regression**:
  + ***Mean Absolute Error (MAE)****:* Average of absolute errors between predicted and actual values.
  + ***Mean Squared Error (MSE)****:* Average of squared errors between predicted and actual values.
  + ***R-squared****:* A statistical measure representing the proportion of the variance for a dependent variable that's explained by the independent variables.

#### ****5. Algorithms****

* **Classification Algorithms**:
  + Decision Trees
  + Logistic Regression
  + Support Vector Machines (SVM)
  + k-Nearest Neighbors (K-NN)
* **Regression Algorithms**:
  + Linear Regression
  + Polynomial Regression
  + Support Vector Regression (SVR)
  + Ridge and Lasso Regression

#### ****6. Nature of Data****

* **Classification**: Data is often labeled into distinct categories.
* **Regression**: Data involves continuous variables and numeric predictions.

### ****Summary****

* **Classification**: Used for tasks where the output is a category or class. Example: Email spam detection.
* **Regression**: Used for tasks where the output is a continuous value. Example: Predicting house prices.

**Radial Basis Function (RBF) Networks**

Radial Basis Function (RBF) networks are a type of artificial neural network that uses radial basis functions as the activation functions for the hidden layer neurons. They are commonly used for function approximation, classification, and regression tasks.

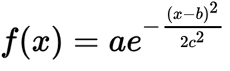
**Architecture**

An RBF network typically consists of three layers:

* *Input Layer*: This layer receives the input data and passes it to the hidden layer.
* *Hidden Layer*: This layer consists of a set of neurons that use radial basis functions as the activation functions. Each neuron in the hidden layer has a center and a width, which are used to compute the output of the neuron.
* *Output Layer*: This layer takes the output from the hidden layer and produces the final output of the network.

**Radial Basis Functions (RBFs):**

* These are functions whose output depends solely on the distance from the origin.
* Commonly used RBFs include Gaussian functions.
* The output of an RBF neuron is typically high when the input is close to its center and decreases as the distance increases.



* where x is the input vector, c is the center of the RBF, and σ is the width of the function. Other RBFs include the Multiquadric and Inverse Multiquadric functions.
* **Centers**: These are the points in the input space around which the RBFs are centered. They can be determined using clustering algorithms like K-means.
* **Weights**: The coefficients applied to the RBF outputs to produce the final prediction. These weights are learned during the training phase.

**Training Process**:

1. **Center Selection**: Determine the centers of the RBFs, usually by clustering the training data (e.g., using K-means clustering).
2. **Parameter Estimation**: Set the width of the RBFs, which can be fixed or learned from the data.
3. **Weight Training**: Train the network by adjusting the weights in the output layer to minimize the error between the predicted and actual outputs. This is often done using linear least squares or other optimization methods.

**Applications**:

* **Function Approximation**: Modeling complex functions in various fields, such as finance or engineering.
* **Classification**: Used in pattern recognition tasks, such as handwriting recognition or medical diagnosis.
* **Regression**: Predicting continuous values, such as stock prices or environmental data.

**Locally Weighted Linear Regression (LWLR)**

Locally Weighted Linear Regression (LWLR) is a non-parametric regression algorithm that combines the simplicity of linear regression with the flexibility of non-parametric methods. It is a type of lazy learning algorithm, which means that it does not require a separate training phase and can be used to make predictions directly.

**How LWLR Works**

**Step 1: Compute Weights**

* Assign weights to each training data point based on its proximity to the query point
* Use a kernel function to compute the weights, such as:
  + Gaussian kernel
  + Triangular kernel
  + Epanechnikov kernel
* The kernel function assigns:
  + Higher weights to data points closer to the query point
  + Lower weights to data points farther away from the query point

**Step 2: Compute Weighted Linear Regression**

* Use the computed weights to perform a weighted linear regression
* Similar to ordinary linear regression, but with weights
* Compute the regression coefficients using the weighted data points

**Step 3: Make Prediction**

* Use the weighted linear regression model to make a prediction at the query point
* The prediction is based on the weighted regression coefficients and the query point's features

Here is a summary of the LWLR process in pointwise form:

* Compute weights for each training data point using a kernel function
* Perform weighted linear regression using the computed weights
* Make a prediction at the query point using the weighted linear regression model.

**Key Components of LWLR**

1. **Kernel Function**: The kernel function is used to compute the weights for the training data points. Common kernel functions include the Gaussian kernel, triangular kernel, and Epanechnikov kernel.
2. **Bandwidth**: The bandwidth is a parameter that controls the width of the kernel function. A smaller bandwidth results in a more localized model, while a larger bandwidth results in a more global model.
3. **Weighted Linear Regression**: The weighted linear regression algorithm is used to make a prediction at the query point. The algorithm uses the weights computed by the kernel function to compute the regression coefficients.

**Advantages of LWLR**

1. **Flexibility**: LWLR can handle non-linear relationships between the input and output variables.
2. **Locality**: LWLR can capture local patterns in the data, which can be useful when the relationships between the input and output variables vary across different regions of the input space.
3. **Interpretability**: LWLR provides interpretable results, as the weights assigned to the training data points can be used to understand the importance of each feature.

**Disadvantages of LWLR**

1. **Computational Cost**: LWLR can be computationally expensive, especially when dealing with large datasets.
2. **Overfitting**: LWLR can suffer from overfitting, especially when the bandwidth is too small.
3. **Choice of Kernel**: The choice of kernel function and bandwidth can significantly affect the performance of LWLR.

**Applications of LWLR**

1. **Function Approximation**: LWLR can be used to approximate complex functions in a variety of fields, including engineering, physics, and economics.
2. **Time Series Analysis**: LWLR can be used to analyze time series data and make predictions about future values.
3. **Image Processing**: LWLR can be used in image processing applications, such as image denoising and image segmentation.

**Sample Error**

* **Definition**: Sample error, also known as the empirical error or training error, is the difference between the predicted values and the actual values in the training dataset. It measures how well the model performs on the data it was trained on.

**True Error**

* **Definition**: True error, also known as the generalization error or out-of-sample error, is the error rate of a model when applied to new, unseen data from the same distribution. It measures the model’s ability to generalize beyond the training data.

**Random Variable**

* **Definition:** A variable whose value is subject to random variation. It represents a numerical outcome of a random phenomenon.

**Expected Value**

* **Definition**: The expected value (or mean) of a random variable is a measure of the central tendency of the probability distribution. It is the long-run average value of repetitions of the experiment it represents.

**Variance**

* **Definition:** A measure of how spread out the values of a random variable are. It quantifies the deviation from the expected value.

**Linear Regression**

Linear regression is a statistical method used to model the relationship between a dependent variable and one or more independent variables. The goal is to find the best-fitting line (or hyperplane in higher dimensions) that minimizes the difference between the predicted and actual values.

**Problem of Crowding in Genetic Algorithms**

**Definition**: Crowding in Genetic Algorithms occurs when the population becomes similar or converges in a small region of the solution space, reducing diversity and leading to suboptimal solutions.

**Problems Caused**:

1. **Premature Convergence**: The algorithm may get stuck in a local optimum due to high similarity among individuals.
2. **Loss of Diversity**: Reduced variability limits exploration, causing stagnation and poor performance.

**Mitigation Strategies**:

1. **Diversity Preservation**: Techniques like fitness sharing and crowding distance help maintain diversity.
2. **Adaptive Methods**: Adjust mutation rates and fitness scaling to balance exploration and exploitation.
3. **Niching**: Use methods like speciation to encourage diverse subpopulations.
4. **Immigration**: Periodically introduce new solutions to maintain diversity.