Introduction to Intelligent Systems

Machine Learning

Machine learning is a subset of artificial intelligence in the field of computer science that often uses statistical techniques to give computers the ability to "learn" (i.e., progressively improve performance on a specific task) with data, without being explicitly programmed. In the past decade, machine learning has given us self-driving cars, practical speech recognition, effective web search, and a vastly improved understanding of the human genome.

Machine learning tasks

Machine learning tasks are typically classified into two broad categories, depending on whether there is a learning "signal" or "feedback" available to a learning system:

1. **Supervised learning**: The computer is presented with example inputs and their desired outputs, given by a "teacher", and the goal is to learn a general rule that maps inputs to outputs. As special cases, the input signal can be only partially available, or restricted to special feedback:

2. **Semi-supervised learning**: the computer is given only an incomplete training signal: a training set with some (often many) of the target outputs missing.

3. **Active learning**: the computer can only obtain training labels for a limited set of instances (based on a budget), and also has to optimize its choice of objects to acquire labels for. When used interactively, these can be presented to the user for labeling.

4. **Reinforcement learning**: training data (in form of rewards and punishments) is given only as feedback to the program's actions in a dynamic environment, such as driving a vehicle or playing a game against an opponent.

5. **Unsupervised learning**: No labels are given to the learning algorithm, leaving it on its own to find structure in its input. Unsupervised learning can be a goal in itself (discovering hidden patterns in data) or a means towards an end (feature learning).

Supervised Learning	Unsupervised Learning	Instance Based Learning
Decision tree algorithm Back propagation Algorithm Naïve Bayes Algorithm K nearest neighbour algorithm(lazy learning algorithm)	EM algorithm K means algorithm	Locally weighted Regression algorithm

Intelligent System Applications

In classification, inputs are divided into two or more classes, and the learner must produce a model that assigns unseen inputs to one or more (multi-label classification) of these classes. This is typically tackled in a supervised manner. Spam filtering is an example of classification, where the inputs are email (or other) messages and the classes are "spam" and "not spam".

In regression, also a supervised problem, the outputs are continuous rather than discrete. In clustering, a set of inputs is to be divided into groups. Unlike in classification, the groups are not known beforehand, making this typically an unsupervised task.

Density estimation finds the distribution of inputs in some space. Dimensionality reduction simplifies inputs by mapping them into a lower dimensional space. Topic modeling is a related problem, where a program is given a list of human language documents and is tasked with finding out which documents cover similar topics.

Machine learning Approaches

1. Decision tree learning

Decision tree learning uses a decision tree as a predictive model, which maps observations about an item to conclusions about the item's target value.

2. Association rule learning

Association rule learning is a method for discovering interesting relations between variables in large databases.

3. Artificial neural networks

An artificial neural network (ANN) learning algorithm, usually called "neural network" (NN), is a learning algorithm that is vaguely inspired by biological neural networks. Computations are

structured in terms of an interconnected group of artificial neurons, processing information using a connectionist approach to computation. Modern neural networks are non-linear statistical data modeling tools. They are usually used to model complex relationships between inputs and outputs, to find patterns in data, or to capture the statistical structure in an unknown joint probability distribution between observed variables.

4. Deep learning

Falling hardware prices and the development of GPUs for personal use in the last few years have contributed to the development of the concept of deep learning which consists of multiple hidden layers in an artificial neural network. This approach tries to model the way the human brain processes light and sound into vision and hearing. Some successful applications of deep learning are computer vision and speech Recognition.

5. Inductive logic programming

Inductive logic programming (ILP) is an approach to rule learning using logic Programming as a uniform representation for input examples, background knowledge, and hypotheses. Given an encoding of the known background knowledge and a set of examples represented as a logical database of facts, an ILP system will derive a hypothesized logic program that entails all positive and no negative examples. Inductive programming is a related field that considers any kind of programming languages for representing hypotheses (and not only logic programming), such as functional programs.

6. Support vector machines

Support vector machines (SVMs) are a set of related supervised learning methods used for classification and regression. Given a set of training examples, each marked as belonging to one of two categories, an SVM training algorithm builds a model that predicts whether a new example falls into one category or the other.

7. Clustering

Cluster analysis is the assignment of a set of observations into subsets (called clusters) so that observations within the same cluster are similar according to some pre designated criterion or criteria, while observations drawn from different clusters are dissimilar. Different clustering techniques make different assumptions on the structure of the data, often defined by some similarity

metric and evaluated for example by internal compactness (similarity between members of the same cluster) and separation between different clusters. Other methods are based on estimated density and graph connectivity. Clustering is a method of unsupervised learning, and a common technique for statistical data analysis.

8. Bayesian networks

A Bayesian network, belief network or directed acyclic graphical model is a probabilistic graphical model that represents a set of random variables and their conditional independencies via a directed acyclic graph (DAG). For example, a Bayesian network could represent the probabilistic relationships between diseases and symptoms. Given symptoms, the network can be used to compute the probabilities of the presence of various diseases. Efficient algorithms exist that perform inference and learning.

9. Reinforcement learning

Reinforcement learning is concerned with how an agent ought to take actions in an environment so as to maximize some notion of long-term reward. Reinforcement learning algorithms attempt to find a policy that maps states of the world to the actions the agent ought to take in those states. Reinforcement learning differs from the supervised learning problem in that correct input/output pairs are never presented, nor sub-optimal actions explicitly corrected.

10. Similarity and metric learning

In this problem, the learning machine is given pairs of examples that are considered similar and pairs of less similar objects. It then needs to learn a similarity function (or a distance metric function) that can predict if new objects are similar. It is sometimes used in Recommendation systems.

11. Genetic algorithms

A genetic algorithm (GA) is a search heuristic that mimics the process of natural selection, and uses methods such as mutation and crossover to generate new genotype in the hope of finding good solutions to a given problem. In machine learning, genetic algorithms found some uses in the 1980s and 1990s. Conversely, machine learning techniques have been used to improve the performance of genetic and evolutionary algorithms.

12. Rule-based machine learning

Rule-based machine learning is a general term for any machine learning method that identifies, learns, or evolves "rules" to store, manipulate or apply, knowledge. The defining characteristic of a rule- based machine learner is the identification and utilization of a set of relational rules that collectively represent the knowledge captured by the system. This is in contrast to other machine learners that commonly identify a singular model that can be universally applied to any instance in order to make a prediction. Rule-based machine learning approaches include learning classifier systems, association rule learning, and artificial immune systems.

13. Feature selection approach

Feature selection is the process of selecting an optimal subset of relevant features for use in model construction. It is assumed the data contains some features that are either redundant or irrelevant, and can thus be removed to reduce calculation cost without incurring much loss of information. Common optimality criteria include accuracy, similarity and information measures.

I Polynomial Curve Fitting

1. **Polynomial Regression**

Polynomial regression is a special case of linear regression where we fit a polynomial equation on the data with a curvilinear relationship between the target variable and the independent variables.

In a curvilinear relationship, the value of the target variable changes in a non-uniform manner with respect to the predictor (s).

In Linear Regression, with a single predictor, we have the following equation:

$$y = w_0 + w_1 x$$

where, y is the target, x is the predictor, w_0 is the bias, and 1 is the weight in the regression equation. This linear equation can be used to represent a linear relationship. But, in polynomial regression, we have a polynomial equation of degree n represented as:

$$y = w_0 + w_1 x + w_2 x^2 + w_3 x^3 + \ldots\ldots\ldots + w_n x^n$$

Here, w_0 is the bias, $w_1, w_2, w_3, \ldots w_n$ are the weights in the equation of the polynomial regression, and n is the degree of the polynomial. The number of higher-order terms increases with the increasing value of n, and hence the equation becomes more complicated.

Polynomial Regression vs. Linear Regression

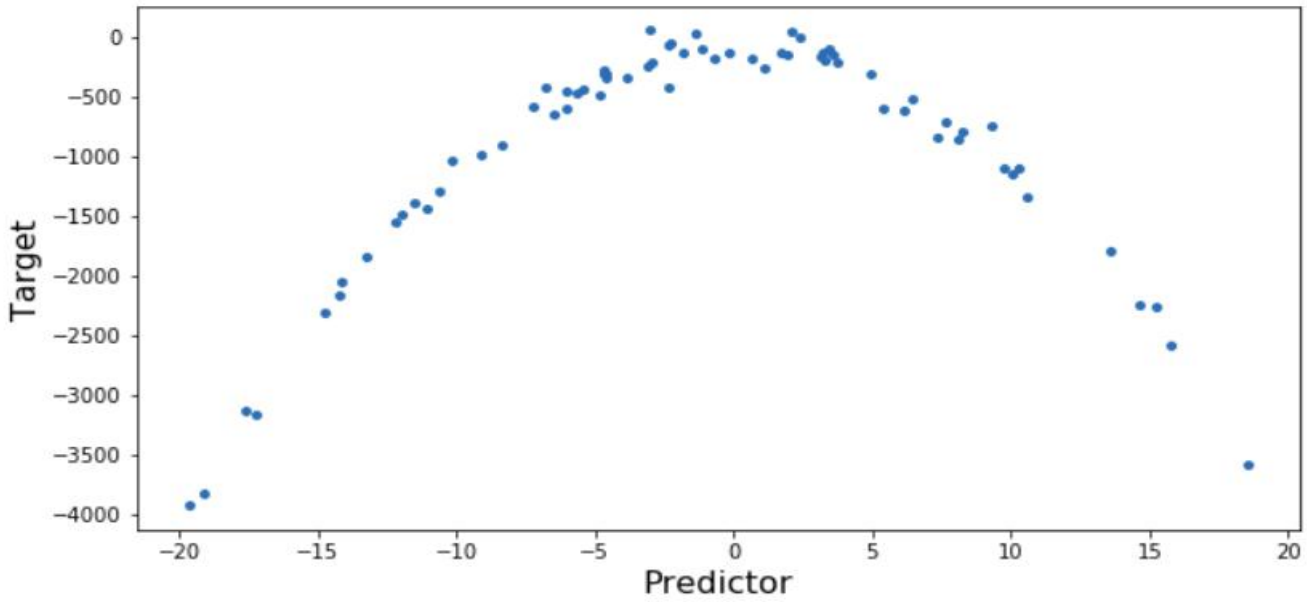
Now that we have a basic understanding of what Polynomial Regression is, let's open up our Python IDE and implement polynomial regression.

We are going to take a slightly different approach here. We will implement both the polynomial regression as well as linear regression algorithms on a simple dataset where we have a curvilinear relationship between the target and predictor. Finally, we will compare the results to understand the difference between the two.

First, import the required libraries and plot the relationship between the target variable and the independent variable:

```
# importing libraries
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
# for calculating mean_squared error
from sklearn.metrics import mean_squared_error
```

```
# creating a dataset with curvilinear relationship
x=10*np.random.normal(0,1,70)
y=10*(-x**2)+np.random.normal(-100,100,70)
# plotting dataset
plt.figure(figsize=(10,5))
plt.scatter(x,y,s=15)
plt.xlabel('Predictor',fontsize=16)
plt.ylabel('Target',fontsize=16)
plt.show()
```



Let's start with Linear Regression first:

```
# Importing Linear Regression

from sklearn.linear_model import LinearRegression

# Training Model

lm=LinearRegression()
lm.fit(x.reshape(-1,1),y.reshape(-1,1))
```

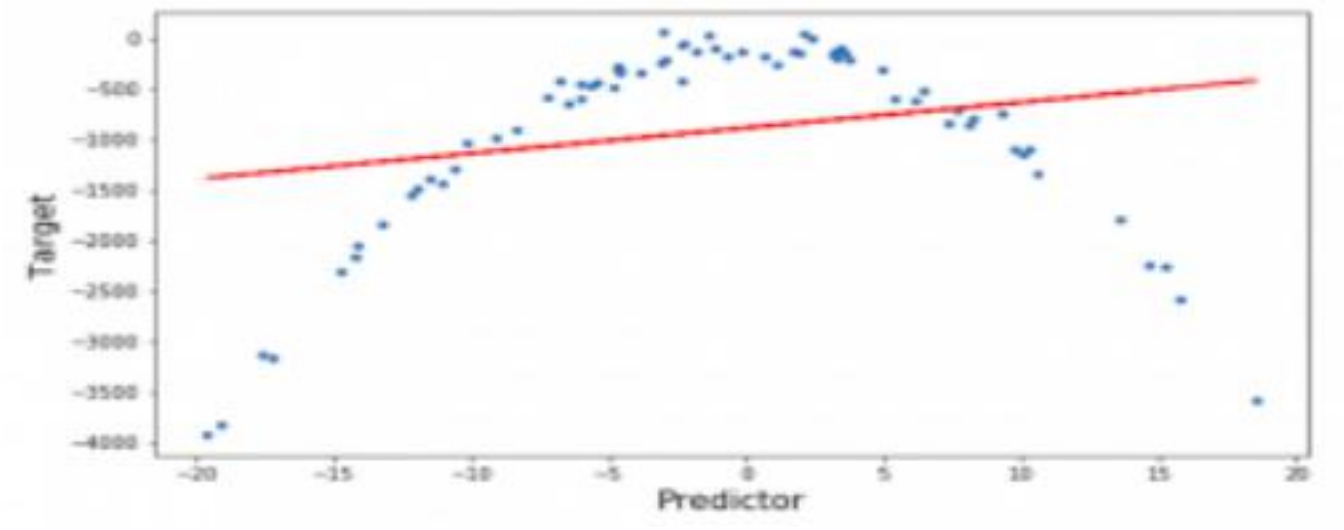
Let's see how linear regression performs on this dataset:

```
y_pred=lm.predict(x.reshape(-1,1))

# plotting prediction

plt.figure(figsize=(10,5))
plt.scatter(x,y,s=15)
plt.plot(x,y_pred,color='r')
plt.xlabel('Predictor',fontsize=16)
plt.ylabel('Target',fontsize=16)
```

```
plt.show()
```



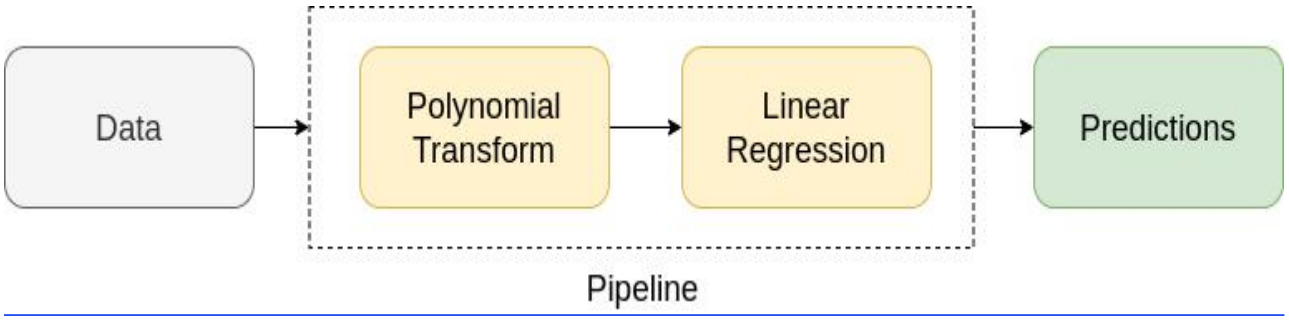

```
print('RMSE for Linear Regression=>',np.sqrt(mean_squared_error(y,y_pred)))
```

```
RMSE for Linear Regression=> 952.4968170543966
```

Here, you can see that the linear regression model is not able to fit the data properly and the RMSE (Root Mean Squared Error) is also very high.

Now, let's try polynomial regression.

The implementation of polynomial regression is a two-step process. First, we transform our data into a polynomial using the Polynomial Features function from sklearn and then use linear regression to fit the parameters:



We can automate this process using pipelines. Pipelines can be created using Pipeline from sklearn. Let's create a pipeline for performing polynomial regression:

```
# importing libraries for polynomial transform
from sklearn.preprocessing import PolynomialFeatures
# for creating pipeline
from sklearn.pipeline import Pipeline
# creating pipeline and fitting it on data
```

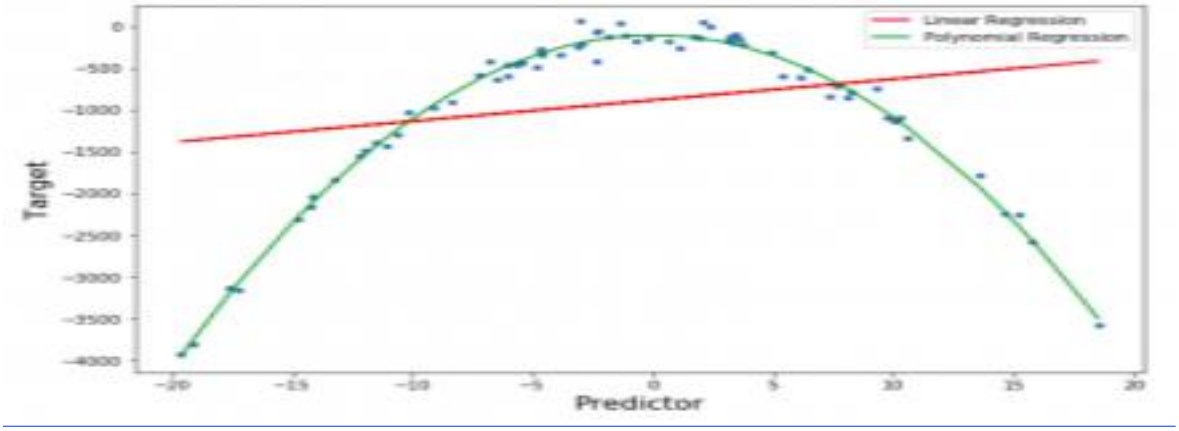
```
Input=[('polynomial',PolynomialFeatures(degree=2)),('modal',LinearRegression())]
pipe=Pipeline(Input)
pipe.fit(x.reshape(-1,1),y.reshape(-1,1))
```

Here, I have taken a 2-degree polynomial. We can choose the degree of polynomial based on the relationship between target and predictor. The 1-degree polynomial is a simple linear regression; therefore, the value of degree must be greater than 1.

With the increasing degree of the polynomial, the complexity of the model also increases. Therefore, the value of n must be chosen precisely. If this value is low, then the model won't be able to fit the data properly and if high, the model will overfit the data easily.

Read more about underfitting and overfitting in machine learning here. Let's take a look at our model's performance:

```
poly_pred=pipe.predict(x.reshape(-1,1))
#sorting predicted values with respect to predictor
sorted_zip = sorted(zip(x,poly_pred))
x_poly, poly_pred = zip(*sorted_zip)
#plotting prediction
splt.figure(figsize=(10,6))
plt.scatter(x,y,s=15)
plt.plot(x,y_pred,color='r',label='Linear Regression')
plt.plot(x_poly,poly_pred,color='g',label='Polynomial Regression')
plt.xlabel('Predictor',fontsize=16)
plt.ylabel('Target',fontsize=16)
plt.legend()
plt.show()
```




```
print('RMSE for Polynomial Regression=>',np.sqrt(mean_squared_error(y,poly_pred)))
```

```
RMSE for Polynomial Regression=> 97.70062852221804
```

We can clearly observe that Polynomial Regression is better at fitting the data than linear regression. Also, due to better-fitting, the RMSE of Polynomial Regression is way lower than that of Linear Regression.

But what if we have more than one predictor?

For 2 predictors, the equation of the polynomial regression becomes:

$$y = w_0 + w_1x_1 + w_2x_2 + w_3x_1x_2 + w_4x_1^2 + w_5x_2^2$$

Where, y is the target, x_1, x_2 are the predictors, w_0 is the bias, and, w_1, w_2, w_3, w_4 and w_5 are the weights in the regression equation

For n predictors, the equation includes all the possible combinations of different order polynomials. This is known as Multi-dimensional Polynomial Regression.

But, there is a major issue with multi-dimensional Polynomial Regression – multicollinearity. Multicollinearity is the interdependence between the predictors in a multiple dimensional regression problem. This restricts the model from fitting properly on the dataset.

1. Write a program to implement the concepts of regression learnt in class via polynomial curve fitting.