Programs

Note: To be installed:

pip install heuristicsearch

pip install decision-tree-ID3-Algorithm==0.0.4

1. **A***

```
from heuristicsearch.a_star_search import AStar
graph_nodes = {
'A': [('B', 6), ('F', 3)],
'B': [('C', 3), ('D', 2)],
'C': [('D', 1), ('E', 5)],
'D': [('C', 1), ('E', 8)],
'E': [('I', 5), ('J', 5)],
'F': [('G', 1),('H', 7)] ,
'G': [('I', 3)],
'H': [('I', 2)],
'I': [('E', 5), ('J', 3)]
}
heuristics = {'A': 10,'B': 8,'C': 5,'D': 7,'E': 3,'F': 6,'G': 5,'H': 3,'I': 1,'J': 0}
graph= AStar(graph_nodes,heuristics)
graph.apply_a_star(start='A', stop='J')
```

A* Search Algorithm:

1. Initialize the open list
2. Initialize the closed list

   put the starting node on the open

   list (you can leave its f at zero)
3. while the open list is not empty

   a) find the node with the least f on

   the open list, call it "q"

   b) pop q off the open list

   c) generate q's 8 successors and set their

   parents to q

   d) for each successor

   i) if successor is the goal, stop search

   ii) else, compute both g and h for successor

   successor.g = q.g + distance between

successor and q

successor.h = distance from goal to successor (This can be done using many ways, we will discuss three heuristics- Manhattan, Diagonal and Euclidean Heuristics)

successor.f = successor.g + successor.h

iii) if a node with the same position as successor is in the OPEN list which has a lower f than successor, skip this successor

iV) if a node with the same position as successor is in the CLOSED list which has a lower f than successor, skip this successor

otherwise, add the node to the open list

end (for loop)

e) push q on the closed list

end (while loop)

2. **AO***

```
from heuristicsearch.ao_star import AOStar
print("Graph-1")
heuristic = {'S': 1, 'A': 7, 'B': 12, 'C': 13, 'D': 5, 'E': 6, 'F': 5, 'G': 7, 'H': 2,}
adjacency_matrix = {'S': [[('A', 1), ('B', 1)], [('C', 1)]],
'A': [[('D', 1)], [('E', 1)]],
'C': [[('F', 1), ('G', 1)]],
'D': [('H', 1)]
}
graph=AOStar(adjacency_matrix,heuristic,'S')
graph.applyAOStar()
```

Algorithm:

Step 1: Place the starting node into OPEN.

Step 2: Compute the most promising solution tree say T0.

Step 3: Select a node n that is both on OPEN and a member of T0. Remove it from OPEN and place it in CLOSE

Step 4: If n is the terminal goal node then leveled n as solved and leveled all the ancestors of n as solved. If the starting node is marked as solved then success and exit.

Step 5: If n is not a solvable node, then mark n as unsolvable. If starting node is marked as unsolvable, then return failure and exit.

Step 6: Expand n. Find all its successors and find their h (n) value, push them into OPEN.

Step 7: Return to Step 2.

Step 8: Exit

3. **CEA:**

```
import csv
a=[]
with open("enjoysport.csv","r") as csvfile:
    fdata=csv.reader(csvfile)
    for row in fdata:
        a.append(row)
        print(row)
num_att=len(a[0])-1
S=['0']*num_att
G=['?']*num_att
print(S)
print(G)
temp=[]
for i in range(0,num_att):
    S[i]=a[0][i]
print(".....")
for i in range(0,len(a)):
    if a[i][num_att]=="Yes":
        for j in range(0,num_att):
            if S[j]!=a[i][j]:
                S[j]='?'
```

```
    for j in range(0,num_att):
        for k in range(0,len(temp)):
            if temp[k][j]!=S[j] and temp[k][j]!='?':
                del temp[k]
if a[i][num_att]=='No':
    for j in range(0,num_att):
        if a[i][j]!=S[j] and S[j]!='?':
            G[j]=S[j]
            temp.append(G)
            G=['?']*num_att
print(S)
if len(temp)==0:
    print(G)
else:
    print(temp)
print(".....")
```

4. **ID3**

Algorithm:

ID3(Examples, TargetAttribute, Attributes) Input:

Examples are the training examples.

Targetattribute is the attribute whose value is to be predicted by the tree.

Attributes is a list of other attributes that may be tested by the learned decision tree.

Output: Returns a decision tree that correctly classifies the given example.

1. Calculate the Information Gain of each feature.
2. Considering that all rows don't belong to the same class, split the dataset S into subsets using the feature for which the Information Gain is maximum.
3. Make a decision tree node using the feature with the 4. maximum Information gain.
4. If all rows belong to the same class, make the current node as a leaf node with the class as its label.

5. Repeat for the remaining features until we run out of all features, or the decision tree has all leaf nodes.

Program:

```
from ID3Algorithm import ID3
import csv
def load_csv(filename):
    lines=csv.reader(open(filename,"r"))
    dataset = list(lines)
    headers = dataset.pop(0)
    return dataset,headers


dataset_train, headers_train = load_csv('PlayTennis.csv')
dataset_test, headers_test = load_csv('PlayTennisTest.csv')


id3 = ID3(dataset_train,headers_train,dataset_test,headers_test)
id3.build_tree()
id3.classify()
```

5. Backpropagation

```
import numpy as np
X = np.array(([2, 9], [1, 5], [3, 6]))
y = np.array(([92], [86], [89]))
y = y/100 # max test score is 100

def sigmoid(x):
    return 1/(1 + np.exp(-x))

def derivatives_sigmoid(x):
    return x * (1 - x)

epoch=10000
lr=0.1
inputlayer_neurons = 2
hiddenlayer_neurons = 3
output_neurons = 1
```

```
wh=np.random.uniform(size=(inputlayer_neurons,hiddenlayer_neurons))
bias_hidden=np.random.uniform(size=(1,hiddenlayer_neurons))
weight_hidden=np.random.uniform(size=(hiddenlayer_neurons,output_neurons))
bias_output=np.random.uniform(size=(1,output_neurons))

for i in range(epoch):
    hinp1=np.dot(X,wh)
    hinp= hinp1 + bias_hidden
    hlayer_activation = sigmoid(hinp)


    outinp1=np.dot(hlayer_activation,weight_hidden)
    outinp= outinp1+ bias_output
    output = sigmoid(outinp)


    EO = y-output

    outgrad = derivatives_sigmoid(output)

    d_output = EO * outgrad

    EH = d_output.dot(weight_hidden.T)

    hiddengrad = derivatives_sigmoid(hlayer_activation)
    d_hiddenlayer = EH * hiddengrad

    weight_hidden += hlayer_activation.T.dot(d_output) *lr
    bias_hidden += np.sum(d_hiddenlayer, axis=0,keepdims=True) *lr

    wh += X.T.dot(d_hiddenlayer) *lr
    bias_output += np.sum(d_output, axis=0,keepdims=True) *lr

print("Input: \n" + str(X))
print("Actual Output: \n" + str(y))
print("Predicted Output: \n" ,output)
```

6. **Naive Bayes:**

```
import pandas as pd

PlayTennis=pd.read_csv("tennis.csv")
```

```
print("\n Given Data set is: \n",PlayTennis)

from sklearn.preprocessing import LabelEncoder
Le=LabelEncoder()

PlayTennis['outlook']=Le.fit_transform(PlayTennis['outlook'])
PlayTennis['temp']=Le.fit_transform(PlayTennis['temp'])
PlayTennis['humidity']=Le.fit_transform(PlayTennis['humidity'])
PlayTennis['windy']=Le.fit_transform(PlayTennis['windy'])
PlayTennis['play']=Le.fit_transform(PlayTennis['play'])

print("\n Encoded Data set:n",PlayTennis)

x=PlayTennis.drop(['play'],axis=1)
y=PlayTennis['play']

from sklearn.model_selection import train_test_split
from sklearn.naive_bayes import GaussianNB
from sklearn.metrics import accuracy_score

x_train,x_test,y_train,y_test = train_test_split(x,y,test_size=0.20)

print("\n x_train\n",x_train)
print("\n y_train\n",y_train)
print("\n x_test\n",x_test)
print("\n y_test\n",y_test)

classifier= GaussianNB()
classifier.fit(x_train,y_train)

accuracy=accuracy_score(classifier.predict(x_test),y_test)

print("\n Accuracy: ",accuracy)
```

7. **K means and EM**

```
import matplotlib.pyplot as plt
from sklearn import datasets
```

```
from sklearn.cluster import KMeans
import pandas as pd
import numpy as np

iris = datasets.load_iris()
X = pd.DataFrame(iris.data)
X.columns =  ['Sepal_Length','Sepal_Width','Petal_Length','Petal_Width']
y = pd.DataFrame(iris.target)
y.columns = ['Targets']

model = KMeans(n_clusters=3)
model.fit(X)

plt.figure(figsize=(14,14))
colormap = np.array(['red', 'lime', 'black'])

plt.subplot(2, 2, 1)
plt.scatter(X.Petal_Length, X.Petal_Width, c=colormap[y.Targets], s=40)
plt.title('Real Clusters')
plt.xlabel('Petal Length')
plt.ylabel('Petal Width')
# Plot the Models Classifications
plt.subplot(2, 2, 2)
plt.scatter(X.Petal_Length, X.Petal_Width, c=colormap[model.labels_], s=40)
plt.title('K-Means Clustering')
plt.xlabel('Petal Length')
plt.ylabel('Petal Width')

from sklearn import preprocessing
scaler = preprocessing.StandardScaler()
scaler.fit(X)
xsa = scaler.transform(X)
```

```
xs = pd.DataFrame(xsa, columns = X.columns)

from sklearn.mixture import GaussianMixture
gmm = GaussianMixture(n_components=3)
gmm.fit(xs)
gmm_y = gmm.predict(xs)
plt.subplot(2, 2, 3)
plt.scatter(X.Petal_Length, X.Petal_Width, c=colormap[gmm_y], s=40)
plt.title('GMM Clustering')
plt.xlabel('Petal Length')
plt.ylabel('Petal Width')
print('Observation: The GMM using EM algorithm based clustering matched
the true labels more closely than the Kmeans.')
```

8. KNN

```
from sklearn.model_selection import train_test_split
from sklearn.neighbors import KNeighborsClassifier
from sklearn import datasets

iris=datasets.load_iris()
print("Iris Data set loaded...")

x_train, x_test, y_train, y_test =
train_test_split(iris.data,iris.target,test_size=0.1)
print("Dataset is split into training and testing...")
print("Size of trainng data and its label",x_train.shape,y_train.shape)
print("Size of trainng data and its label",x_test.shape, y_test.shape)

for i in range(len(iris.target_names)):
    print("Label", i , "-",str(iris.target_names[i]))
    # Create object of KNN classifier
classifier = KNeighborsClassifier(n_neighbors=1)
```

```
# Perform Training
classifier.fit(x_train, y_train) # Perform testing
y_pred=classifier.predict(x_test)

# Display the results
print("Results of Classification using K-nn with K=1 ")
for r in range(0,len(x_test)):
    print(" Sample:", str(x_test[r]), " Actual-label:", str(y_test[r]), "
Predicted-label:", str(y_pred[r]))
print("Classification Accuracy :" , classifier.score(x_test,y_test));

from sklearn.metrics import classification_report, confusion_matrix
print('Confusion Matrix')
print(confusion_matrix(y_test,y_pred))
print('Accuracy Metrics')
print(classification_report(y_test,y_pred))
```

9. Locally weighted regression

```
import matplotlib.pyplot as plt
import pandas as pd
import numpy as np

def kernel(point,xmat, k):
    m,n = np.shape(xmat)
    weights = np.mat(np.eye((m))) # eye - identity matrix
    for j in range(m):
        diff = point - X[j]
        weights[j,j] = np.exp(diff*diff.T/(-2.0*k**2))
    return weights
def localWeight(point,xmat,ymat,k):
    wei = kernel(point,xmat,k)
    W = (X.T*(wei*X)).I*(X.T*(wei*ymat.T))
    return W
```

```
def localWeightRegression(xmat,ymat,k):
    m,n = np.shape(xmat)
    ypred = np.zeros(m)
    for i in range(m):
        ypred[i] = xmat[i]*localWeight(xmat[i],xmat,ymat,k)
    return ypred

def graphPlot(X,ypred):
    sortindex = X[:,1].argsort(0) #argsort - index of the smallest
    xsort = X[sortindex][:,0]
    fig = plt.figure()
    ax = fig.add_subplot(1,1,1)
    ax.scatter(bill,tip, color='green')
    ax.plot(xsort[:,1],ypred[sortindex], color = 'red', linewidth=8)
    plt.xlabel('Total bill')
    plt.ylabel('Tip')
    plt.show();

# load data points
data = pd.read_csv('data10_tips.csv')
bill = np.array(data.total_bill) # We use only Bill amount and Tips data
tip = np.array(data.tip)

mbill = np.mat(bill) # .mat will convert nd array is converted in 2D array
mtip = np.mat(tip)
m= np.shape(mbill)[1]
one = np.mat(np.ones(m))
X = np.hstack((one.T,mbill.T)) # 244 rows, 2 cols

ypred = localWeightRegression(X,mtip,10) # increase k to get smooth curves
graphPlot(X,ypred)
```

Viva Questions

1. What is machine learning?
A: Machine learning (ML) is a discipline of artificial intelligence (AI) that provides machines with the ability to automatically learn from data and past experiences while identifying patterns to make predictions with minimal human intervention.

OR

Machine learning is a branch of artificial intelligence (AI) and computer science which focuses on the use of data and algorithms to imitate the way that humans learn, gradually improving its accuracy.

2. Define supervised learning
A: Supervised learning, also known as supervised machine learning, is a subcategory of machine learning and artificial intelligence. It is defined by its use of labeled datasets to train algorithms that to classify data or predict outcomes accurately.

3. Define unsupervised learning
A: Unsupervised learning, also known as unsupervised machine learning, uses machine learning algorithms to analyze and cluster unlabeled datasets. These algorithms discover hidden patterns or data groupings without the need for human intervention.

4. Define semi supervised learning
A: Semi-supervised learning occurs when only part of the given input data has been labelled. Semi-supervised machine learning is a combination of supervised and unsupervised learning. It uses a small amount of labeled data and a large amount of unlabeled data, which provides the benefits of both unsupervised and supervised learning while avoiding the challenges of finding a large amount of labeled data.

5. Define reinforcement learning
A: Reinforcement Learning is a feedback-based Machine learning technique in which an agent learns to behave in an environment by performing the actions and seeing the results of actions. For each good action, the agent gets positive feedback, and for each bad action, the agent gets negative feedback or penalty.

6. What do you mean by hypotheses?
A: A hypothesis is a function that best describes the target in supervised machine learning. The hypothesis that an algorithm would come up depends upon the data and also depends upon the restrictions and bias that we have imposed on the data.

7. What is classification?

A: Classification is a systematic grouping of observations into categories, such as when biologists categorize plants, animals, and other lifeforms into different taxonomies. It is one of the primary uses of data science and machine learning.

8. What is clustering?
A: Grouping unlabeled examples is called clustering. As the examples are unlabeled, clustering relies on unsupervised machine learning.

9. Define precision, accuracy and recall
A: Accuracy tells you how many times the ML model was correct overall. Precision is how good the model is at predicting a specific category. Recall tells you how many times the model was able to detect a specific category.

10. Define entropy
A: A measure of the randomness in the information being processed. The higher the entropy, the harder it is to draw any conclusions from that information.

11. Define regression
A: Regression is a statistical method that attempts to determine the strength and character of the relationship between one dependent variable (usually denoted by Y) and a series of other variables (known as independent variables).

12. How Knn is different from k-means clustering
A:KNN is a supervised learning algorithm mainly used for classification problems, whereas K-Means (aka K-means clustering) is an unsupervised learning algorithm. K in K-Means refers to the number of clusters, whereas K in KNN is the number of nearest neighbors (based on the chosen distance metric).
13. What is concept learning?
A: A task of acquiring a potential hypothesis (solution) that best fits the given training examples.

14. Define specific boundary and general boundary
A: The general boundary , with respect to hypothesis space and training data , is the set of maximally general members of consistent with . The specific boundary , with respect to hypothesis space and training data , is the set of minimally general (i.e., maximally specific) members of consistent with .

15. Define target function
A: A method for solving a problem that an AI algorithm parses its training data to find. Once an algorithm finds its target function, that function can be used to predict results (predictive analysis). The function can then be used to find output data related to inputs for real problems where, unlike training sets, outputs are not included.

16. Define decision tree

A: Decision Tree is a Supervised learning technique that can be used for both classification and Regression problems, but mostly it is preferred for solving Classification problems. It is a tree-structured classifier, where internal nodes represent the features of a dataset, branches represent the decision rules and each leaf node represents the outcome.

17. What is ANN

A:The term "Artificial Neural Network" is derived from Biological neural networks that develop the structure of a human brain. Similar to the human brain that has neurons interconnected to one another, artificial neural networks also have neurons that are interconnected to one another in various layers of the networks. These neurons are known as nodes.

18. Explain gradient descent approximation

A: Gradient Descent is an optimization algorithm for finding a local minimum of a differentiable function. Gradient descent in machine learning is simply used to find the values of a function's parameters (coefficients) that minimize a cost function as far as possible.

19. State Bayes theorem

A: Bayes theorem provides a principled way to calculate the posterior probability of each hypothesis given the training data. It acts as a basis for a straightforward learning algorithm that calculates the probability for each possible hypothesis, then outputs the most probable

20. Define Bayesian belief networks

A: Bayesian belief network is a useful way to represent probabilistic models and visualize them.

OR

A Bayesian belief network (Bayesian network) represents a joint probability distribution for a set of variables.In particular, each node is asserted to be conditionally independent of its non-descendants, given its immediate parents. Associated with each node is a conditional probability table that specifies the conditional distribution for the variable given its immediate parents in the graph.

21. Differentiate hard and soft clustering

A: Hard clustering is about grouping the data items such that each item is only assigned to one cluster. As an instance, we want the algorithm to read all of the tweets and determine if a tweet is a positive or a negative tweet.

Sometimes we don't need a binary answer. Soft clustering is about grouping the data items such that an item can exist in multiple clusters.

22. Define variance
A:Variance refers to the changes in the model when using different portions of the training data set. Simply stated, variance is the variability in the model prediction—how much the ML function can adjust depending on the given data set.

23. What is inductive machine learning?
A: Inductive Learning Algorithm (ILA) is an iterative and inductive machine learning algorithm which is used for generating a set of a classification rule, which produces rules of the form "IF-THEN", for a set of examples, producing rules at each iteration and appending to the set of rules.

24. Why K nearest neighbor algorithm is lazy learning algorithm
A: Because it does no training at all when you supply the training data. At training time, all it is doing is storing the complete data set but it does not do any calculations at this point.

25. Why naïve Bayes is naive
A: Naive Bayes is called naive because it assumes that each input variable is independent. This is a strong assumption and unrealistic for real data; however, the technique is very effective on a large range of complex problems.

26. Mention classification algorithms
A: Logistic Regression, Naive Bayes, K-Nearest Neighbors, Decision Tree, Support Vector Machines

27. Define pruning
A: In machine learning and data mining, pruning is a technique associated with decision trees. Pruning reduces the size of decision trees by removing parts of the tree that do not provide power to classify instances.

28. Differentiate Clustering and classification
A: The primary difference between classification and clustering is that classification is a supervised learning approach where a specific label is provided to the machine to classify new observations. Here the machine needs proper testing and training for the label verification. So, classification is a more complex process than clustering. On the other hand, clustering is an unsupervised learning approach where grouping is done on

similarities basis. Here the machine learns from the existing data and does not need any training.

29. Mention clustering algorithms- Tell them 3 or 4

A:

1. Affinity Propagation
2. Agglomerative Hierarchical Clustering
3. BIRCH (Balanced Iterative Reducing and Clustering using Hierarchies)
4. DBSCAN (Density-Based Spatial Clustering of Applications with Noise)
5. Gaussian Mixture Models (GMM)
6. K-Means, Mean Shift Clustering
7. Mini-Batch K-Means
8. OPTICS
9. Spectral Clustering

30. Define Bias

A: Bias is a phenomenon that skews the result of an algorithm in favor or against an idea. Bias is considered a systematic error that occurs in the machine learning model itself due to incorrect assumptions in the ML process.

31. What is learning rate? Why it is need.

A: This parameter determines how fast or slow we will move towards the optimal weights. If the learning rate is very large we will skip the optimal solution. If it is too small we will need too many iterations to converge to the best values. So using a good learning rate is crucial.