|  |  |  |  |
| --- | --- | --- | --- |
| A picture containing drawing, stop, room  Description automatically generated | Machine Learning  Practical # 9 | | |
|  |  |  |  |
|  |  |  |  |
| **Name** | Ninad Karlekar | **Roll Number** | 22306A1012 |
| **Subject/Course:** | Machine Learning | **Class** | M.Sc. IT – Sem III |
| **Topic** | Learning | **Batch** | Batch 1 |
|  |  |  |  |
| **Topic: Backpropagation algorithm and Text pre-processing, Text clustering, classification** | | | |
| **a) AIM: Build an Artificial Neural Network by implementing the Backpropagation algorithm and test the same using appropriate data sets.**  **DESCRIPTION:**  **Code and output:**  import numpy as np  X=np.array(([2,9],[1,5],[3,6]),dtype=float)  Y=np.array(([92],[86],[89]),dtype=float)  X=X/np.amax(X,axis=0)  Y=Y/100;    class NN(object):  def \_\_init\_\_(self):  self.inputsize=2  self.outputsize=1  self.hiddensize=3  self.W1=np.random.randn(self.inputsize,self.hiddensize)  self.W2=np.random.randn(self.hiddensize,self.outputsize)  def forward(self,X):  self.z=np.dot(X,self.W1)  self.z2=self.sigmoidal(self.z)  self.z3=np.dot(self.z2,self.W2)  op=self.sigmoidal(self.z3)  return op;  def sigmoidal(self,s):  return 1/(1+np.exp(-s))  def sigmoidalprime(self,s):  return s\* (1-s)  def backward(self,X,Y,o):  self.o\_error=Y-o  self.o\_delta=self.o\_error \* self.sigmoidalprime(o)  self.z2\_error=self.o\_delta.dot(self.W2.T)  self.z2\_delta=self.z2\_error \* self.sigmoidalprime(self.z2)  self.W1 = self.W1 + X.T.dot(self.z2\_delta)  self.W2= self.W2+ self.z2.T.dot(self.o\_delta)  def train(self,X,Y):  o=self.forward(X)  self.backward(X,Y,o)  obj=NN()  for i in range(2000):  print("input"+str(X))  print("Actual output"+str(Y))  print("Predicted output"+str(obj.forward(X)))  print("loss"+str(np.mean(np.square(Y-obj.forward(X)))))  obj.train(X,Y)    **Learnings:**  This code creates a basic computer program that tries to learn patterns from a small set of information. It uses a special math function to make predictions and adjusts itself to get better over 2000 tries. However, it needs some extra details, like how fast it should learn and a small fix to improve its performance. In a nutshell, it's like a beginner's attempt at building a smart program that needs a bit of fine-tuning.  **b) AIM: Perform Text pre-processing, Text clustering, classification with Prediction, Test Score and Confusion Matrix**  **DESCRIPTION:**  **Code and output:**  import numpy as np  import matplotlib.pyplot as plt  import pandas as pd    dataset = pd.read\_csv('Restaurant\_Reviews.tsv', delimiter = '\t', quoting = 3)    import re  import nltk  nltk.download('stopwords')  from nltk.corpus import stopwords  from nltk.stem.porter import PorterStemmer  corpus = []  for i in range(0,1000):  review = re.sub('[^a-zA-Z]','',dataset['Review'][i])  review = review.lower()  review = review.split()  ps = PorterStemmer()  review = [ps.stem(word) for word in review if not word in set(stopwords.words('english'))]  review = ''.join(review)  corpus.append(review)  #Creating the bag of words model  from sklearn.feature\_extraction.text import CountVectorizer  cv = CountVectorizer(max\_features=1500)  X = cv.fit\_transform(corpus).toarray()  Y = dataset.iloc[:,1].values    #Splitting the dataset into the training set and test set  from sklearn.model\_selection import train\_test\_split  X\_train, X\_test, Y\_train, Y\_test = train\_test\_split(X,Y, test\_size = 0.25, random\_state=100)      #Fitting naive bayes to the training set.  from sklearn.naive\_bayes import GaussianNB  classifier = GaussianNB()  classifier.fit(X\_train, Y\_train)  # Predicting the test set results.  Y\_pred = classifier.predict(X\_test)    #Model Accuracy  from sklearn import metrics  from sklearn.metrics import confusion\_matrix  print("Accuracy:",metrics.accuracy\_score(Y\_test, Y\_pred))  #Making the confusion matrix  from sklearn.metrics import confusion\_matrix  cm = confusion\_matrix(Y\_test, Y\_pred)  print(cm)    **Learnings:** | | | |
|  | | | |
|  | | | |