Topics in Deep learning Hands-On Unit 2

Name – B Pravena Section – B

SRN – PES2UG19CS076

Google collab link -:

https://colab.research.google.com/drive/1pu61MguOq--DY8T7Xqd1UzHvu-3S_SMo?usp=sharing

Linear Kernel in SVM (Hard Margin Classifier**)-:**

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

bankdata = pd.read_csv("bill_authentication.csv")
print("Dimension of dataset -:", bankdata.shape)
print("\n")
print("Top 5 rows in dataset\n", bankdata.head())
print("\n")

X = bankdata.drop('Class', axis=1)
y = bankdata['Class']

from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.20)

from sklearn.svm import SVC
svclassifier = SVC(kernel='linear')
svclassifier.fit(X_train, y_train)
y_pred = svclassifier.predict(X_test)

from sklearn.metrics import classification_report, confusion_matrix
print(confusion_matrix(y_test,y_pred))
print(classification_report(y_test,y_pred))
```

Output -:

```
Dimension of dataset -: (1372, 5)


Top 5 rows in dataset
    Variance  Skewness  Curtosis  Entropy  Class
0   3.62160    8.6661   -2.8073 -0.44699      0
1   4.54590    8.1674   -2.4586 -1.46210      0
2   3.86600   -2.6383    1.9242  0.10645      0
3   3.45660    9.5228   -4.0112 -3.59440      0
4   0.32924   -4.4552    4.5718 -0.98880      0


[[148   3]
 [  0 124]]
              precision    recall  f1-score   support

           0       1.00      0.98      0.99       151
           1       0.98      1.00      0.99       124

    accuracy                           0.99       275
   macro avg       0.99      0.99      0.99       275
weighted avg       0.99      0.99      0.99       275
```

Kernel (Soft Margin) -:

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

url = "https://archive.ics.uci.edu/ml/machine-learning-databases/iris/iris.data"
colnames = ['sepal-length', 'sepal-width', 'petal-length', 'petal-width', 'Class'] # Assign colum names to the dataset
irisdata = pd.read_csv(url, names=colnames) # Read dataset to pandas dataframe

X = irisdata.drop('Class', axis=1)
y = irisdata['Class']

from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.20)

from sklearn.svm import SVC
svclassifier = SVC(kernel='poly', degree=8)
#svclassifier = SVC(kernel='linear')
#svclassifier = SVC(kernel='rbf')
#svclassifier = SVC(kernel='sigmoid')

svclassifier.fit(X_train, y_train)
y_pred = svclassifier.predict(X_test)

from sklearn.metrics import classification_report, confusion_matrix
print("Confusion Matrix\n", confusion_matrix(y_test, y_pred))
print("\n")
print("Classification Report\n", classification_report(y_test, y_pred))
```

Polynomial Kernel Output -:

```
Confusion Matrix
 [[12  0  0]
 [ 0  8  0]
 [ 0  0 10]]


Classification Report
                precision    recall  f1-score   support

    Iris-setosa       1.00      1.00      1.00        12
Iris-versicolor       1.00      1.00      1.00         8
 Iris-virginica       1.00      1.00      1.00        10

       accuracy                           1.00        30
      macro avg       1.00      1.00      1.00        30
   weighted avg       1.00      1.00      1.00        30
```

Linear Kernel Output -:

```
Confusion Matrix
 [[12  0  0]
 [ 0  7  0]
 [ 0  2  9]]


Classification Report
                 precision    recall  f1-score   support

    Iris-setosa       1.00      1.00      1.00        12
Iris-versicolor       0.78      1.00      0.88         7
 Iris-virginica       1.00      0.82      0.90        11

       accuracy                           0.93        30
      macro avg       0.93      0.94      0.92        30
   weighted avg       0.95      0.93      0.93        30
```

RBF Kernel Output-:

```
Confusion Matrix
 [[10  0  0]
 [ 0 10  0]
 [ 0  0 10]]


Classification Report
                 precision    recall  f1-score   support

    Iris-setosa       1.00      1.00      1.00        10
Iris-versicolor       1.00      1.00      1.00        10
 Iris-virginica       1.00      1.00      1.00        10

       accuracy                           1.00        30
      macro avg       1.00      1.00      1.00        30
   weighted avg       1.00      1.00      1.00        30
```

Sigmoid Kernel Output-:

```
Confusion Matrix
 [[ 0  0  9]
 [ 0  0 13]
 [ 0  0  8]]


Classification Report
                 precision    recall  f1-score   support

    Iris-setosa       0.00      0.00      0.00         9
Iris-versicolor       0.00      0.00      0.00        13
 Iris-virginica       0.27      1.00      0.42         8

       accuracy                           0.27        30
      macro avg       0.09      0.33      0.14        30
   weighted avg       0.07      0.27      0.11        30
```

Tuning hyperparameters C and Gamma (Multiclass Classification) -:

```
from sklearn import svm, datasets
import sklearn.model_selection as model_selection
from sklearn.metrics import accuracy_score
from sklearn.metrics import f1_score

iris = datasets.load_iris()

X = iris.data[:, :2]
y = iris.target

X_train, X_test, y_train, y_test = model_selection.train_test_split(X, y, train_size=0.80, test_size=0.20, random_state=101)
rbf = svm.SVC(kernel='rbf', gamma=0.5, C=0.1).fit(X_train, y_train)
poly = svm.SVC(kernel='poly', degree=3, C=100).fit(X_train, y_train)

poly_pred = poly.predict(X_test)
rbf_pred = rbf.predict(X_test)

poly_accuracy = accuracy_score(y_test, poly_pred) # Accuracy of polynomial kernel
poly_f1 = f1_score(y_test, poly_pred, average='weighted')

print('Accuracy (Polynomial Kernel): ', "%.2f" % (poly_accuracy*100))
print('F1 (Polynomial Kernel): ', "%.2f" % (poly_f1*100))

rbf_accuracy = accuracy_score(y_test, rbf_pred) # Accuracy of RBF kernel
rbf_f1 = f1_score(y_test, rbf_pred, average='weighted')
print('Accuracy (RBF Kernel): ', "%.2f" % (rbf_accuracy*100))
print('F1 (RBF Kernel): ', "%.2f" % (rbf_f1*100))
```

```
Accuracy (Polynomial Kernel):  70.00
F1 (Polynomial Kernel):  69.67
Accuracy (RBF Kernel):  76.67
F1 (RBF Kernel):  76.36
```

RBF => For gamma = 0.8, C=0.15 => accuracy reduces to 73.3% and on changing further it reduces to 70%.

Polynomial => On changing C there is no change in accuracy

On changing degree to 2, accuracy increases to 73.33

```
Accuracy (Polynomial Kernel):  73.33
F1 (Polynomial Kernel):  72.45
Accuracy (RBF Kernel):  76.67
F1 (RBF Kernel):  76.36
```

Text Classification using SVM -:

```
[11] import nltk
     nltk.download('punkt')
     nltk.download('wordnet')
     nltk.download('averaged_perceptron_tagger')
     nltk.download('stopwords')
```

```
import pandas as pd
import numpy as np
from nltk.tokenize import word_tokenize
from nltk import pos_tag
from nltk.corpus import stopwords
from nltk.stem import WordNetLemmatizer
from sklearn.preprocessing import LabelEncoder
from collections import defaultdict
from nltk.corpus import wordnet as wn
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn import model_selection, naive_bayes, svm
from sklearn.metrics import accuracy_score
```

```
np.random.seed(500) #Set Random seed

Corpus = pd.read_csv(r"corpus.csv",encoding='latin-1') # Add the Data using pandas

# Step - 1: Data Pre-processing - This will help in getting better results through the classification algorithms
# Step - 1a : Remove blank rows if any.
Corpus['text'].dropna(inplace=True)


# Step - 1b : Change all the text to lower case. This is required as python interprets 'dog' and 'DOG' differently
Corpus['text'] = [entry.lower() for entry in Corpus['text']]


# Step - 1c : Tokenization : In this each entry in the corpus will be broken into set of words
Corpus['text']= [word_tokenize(entry) for entry in Corpus['text']]


# Step - 1d : Remove Stop words, Non-Numeric and perfom Word Stemming/Lemmenting.
# WordNetLemmatizer requires Pos tags to understand if the word is noun or verb or adjective etc. By default it is set to Noun
tag_map = defaultdict(lambda : wn.NOUN)
tag_map['J'] = wn.ADJ
tag_map['V'] = wn.VERB
tag_map['R'] = wn.ADV
for index,entry in enumerate(Corpus['text']):
  Final_words = [] # Declaring Empty List to store the words that follow the rules for this step
  word_Lemmatized = WordNetLemmatizer() # Initializing WordNetLemmatizer()
  for word, tag in pos_tag(entry):
    if word not in stopwords.words('english') and word.isalpha():
      #condition checks for Stop words and consider only alphabets
      word_Final = word_Lemmatized.lemmatize(word,tag_map[tag[0]])
      Final_words.append(word_Final)
  Corpus.loc[index,'text_final'] = str(Final_words) #The final processed set of words for each iteration will be stored in 'text_final'
#print(Corpus['text_final'].head())


#Step - 2: Split the model into Train and Test Data set
Train_X, Test_X, Train_Y, Test_Y = model_selection.train_test_split(Corpus['text_final'],Corpus['label'],test_size=0.3)
```

```
# Step - 3: Label encode the target variable - This is done to transform Categorical data of string type in the data set into numerical values
Encoder = LabelEncoder()
Train_Y = Encoder.fit_transform(Train_Y)
Test_Y = Encoder.fit_transform(Test_Y)


# Step - 4: Vectorize the words by using TF-IDF Vectorizer - This is done to find how important a word in document is in comaprison to the corpus
Tfidf_vect = TfidfVectorizer(max_features=5000)
Tfidf_vect.fit(Corpus['text_final'])
Train_X_Tfidf = Tfidf_vect.transform(Train_X)
Test_X_Tfidf = Tfidf_vect.transform(Test_X)


# Step - 5: Now we can run different algorithms to classify out data check for accuracy
# Classifier - Algorithm - SVM
# fit the training dataset on the classifier
SVM = svm.SVC(C=1.0, kernel='linear', degree=3, gamma='auto')
SVM.fit(Train_X_Tfidf,Train_Y)
# predict the labels on validation dataset
predictions_SVM = SVM.predict(Test_X_Tfidf)
# Use accuracy_score function to get the accuracy
print("SVM Accuracy Score -> ",accuracy_score(predictions_SVM, Test_Y)*100)
```

```
SVM Accuracy Score ->  84.7
```