HELP MANUAL

DDCN-2019@CSED, MNNIT Allahabad

SVM Classification on Lists

SVM Classification on Lists

- **Required packages to implement SVM Classification algorithm on Lists**
  - import numpy as np
  - import matplotlib.pyplot as plt
  - from sklearn import svm

SVM Classification on Lists Contd…

- **Creating Lists and displaying them**
  - **x=[1, 5, 1.5, 8, 1, 9]**
  - **y=[2, 8, 1.8, 8, 0.6, 11]**
  - print (x)
  - print (y)
- **Plot and display scatter chart of x and y**
  - plt.scatter(x,y)
  - plt.show()
- **Creating an array X which stores pair (x, y)**
  - X = np.array([[1,2], [5,8], [1.5,1.8], [8,8], [1,0.6], [9,11]])
- **Create the target class Y as Y= [0,1,0,1,0,1]**

SVM Classification on Lists Contd…

- **Classifying list X and comparing with target Y**
  - clf = svm.SVC(kernel='linear', C = 1.0) **# Linear Kernel and SVC method**
  - clf.fit(X,Y)
  - print(clf.predict([0.58,0.76]))
  - print(clf.predict([10.58,10.76]))
  - w = clf.coef_[0]
  - print("weight",w)
  - a = -w[0] / w[1]
  - print ("Bias",a)

SVM Classification on Lists Contd…

- **Visualizing Results**
  - xx = np.linspace(0,12)
  - print (“xx”,xx)
  - yy = a * xx - clf.intercept_[0] / w[1]
  - print (“yy”,yy)
  - h0 = plt.plot(xx, yy, 'k-', label="SVM Linear Classifier Chart")
  - plt.scatter(X[:, 0], X[:, 1], c = y)
  - plt.legend()
  - plt.show()

SVM Classification on Dataset

SVM Classification on Dataset

- **Required packages to implement SVM Classification algorithm on datasets**
  - from sklearn import datasets
  - import numpy as np
  - import matplotlib.pyplot as plt
  - from sklearn import svm

SVM Classification on Dataset Contd…

- **Loading dataset**
  - iris_dataset = datasets.load_iris()
  - print("Iris data set Description :: ", iris_dataset['DESCR'])
  - print ("Iris feature data :: ", iris_dataset['data'])
  - print ("Iris target :: ", iris_dataset['target'])

SVM Classification on Dataset Contd…

- **Visualizing dataset for sepal attribute**
  - def visuvalize_sepal_data():
    - iris = datasets.load_iris()
    - X = iris.data[:, :2] **# we only take the first two features**
    - y = iris.target
    - plt.scatter(X[:, 0], X[:, 1], c=y, cmap=plt.cm.coolwarm)
    - plt.xlabel('Sepal length')
    - plt.ylabel('Sepal width')
    - plt.title('Sepal Width & Length')
    - plt.show()

SVM Classification on Dataset Contd…

- **Visualizing dataset for petal attribute**
  - def visuvalize_petal_data():
    - iris = datasets.load_iris()
    - X = iris.data[:, 2:] **# we only take the last two features**
    - y = iris.target
    - plt.scatter(X[:, 0], X[:, 1], c=y, cmap=plt.cm.coolwarm)
    - plt.xlabel('Petal length')
    - plt.ylabel('Petal width')
    - plt.title('Petal Width & Length')
    - plt.show()

SVM Classification on Dataset Contd…

- **Classification based on Sepal attribute**
  - iris = datasets.load_iris()
  - X = iris.data[:, :2] **# we only take the Sepal two features**
  - y = iris.target
  - C = 1.0 # SVM regularization parameter
  - Linear Kernel
    - svc = svm.SVC(kernel='linear', C=C).fit(X, y) **#SVC function**
    - lin_svc = svm.LinearSVC(C=C).fit(X, y) **# Linear SVC**
  - rbf/poly Kernel
    - rbf_svc = svm.SVC(kernel='rbf', gamma=0.7, C=C).fit(X, y)
    - poly_svc = svm.SVC(kernel='poly', degree=3, C=C).fit(X, y)

SVM Classification on Dataset Contd…

- **Creating Meshgrid**
  - h = .02 **#step size in the mesh**
  - **# create a mesh to plot in**
  - x_min, x_max = X[:, 0].min() - 1, X[:, 0].max() + 1
  - y_min, y_max = X[:, 1].min() - 1, X[:, 1].max() + 1
  - xx, yy = np.meshgrid(np.arange(x_min, x_max, h), np.arange(y_min, y_max, h))

SVM Classification on Dataset Contd…

- **Visualizing Results**
  - for i, clf in enumerate((svc, lin_svc, rbf_svc, poly_svc)):
  - **# Plot the decision boundary assigning a color to each**
  - **# point in the mesh [x_min, x_max]x[y_min, y_max]**
  - plt.subplot(2, 2, i + 1)
  - plt.subplots_adjust(wspace=0.4, hspace=0.4)
  - Z = clf.predict(np.c_[xx.ravel(), yy.ravel()])
  - **# Put the result into a color plot**
  - Z = Z.reshape(xx.shape)
  - plt.contourf(xx, yy, Z, cmap=plt.cm.coolwarm, alpha=0.8)