HW07 - REPORT

정보컴퓨터공학부 201624536 이국현

June 3, 2022

Chapter 1

서론

1.1 Linear SVM Classifiers

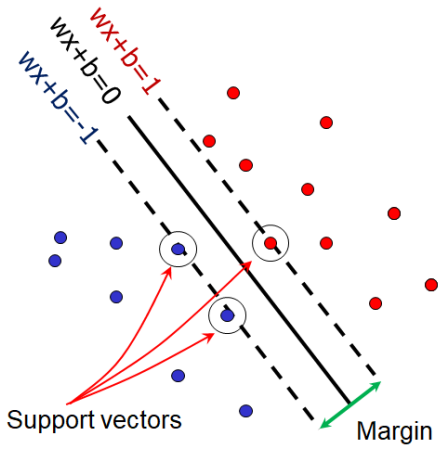

Figure 1.1: Linear SVM Classifiers

x, y 좌표와 해당하는 class가 주어진 Training data가 있다면 Machine learning을 통해 class를 구분 짓는 기준을 정할 수 있다. 이 기준을 좌표 상의 Line으로 그릴 수 있다면 이 Model을 Linear classifier라고 한다. 대표적으로 Support vector machine이 있다. 두 그룹 사이에 수없이 많은 선을 그릴 수 있지만, SVM에서는 1.1과 같이 두 그룹까지의 Margin을 최대화하는 Line을 Model로 학습한다.

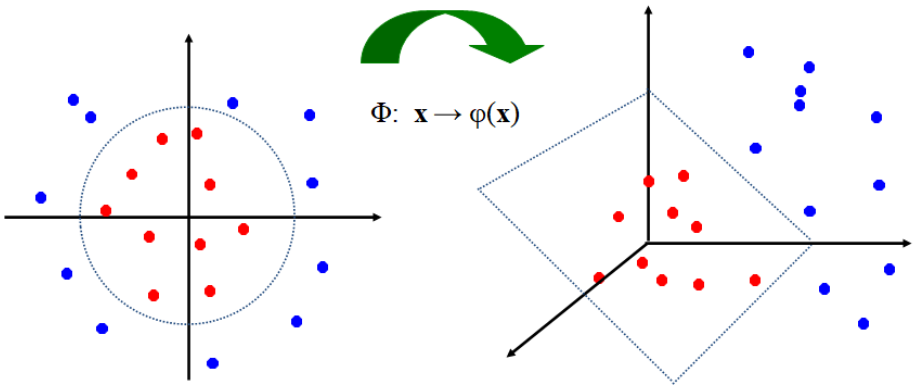

Figure 1.2: Non-linear SVM Classifiers

1.2 Non-linear SVM Classifiers

1.1의 왼쪽 그림과 같이 두 그룹의 경계가 선형적이지 않은 경우 이를 Non-linear classifier라고 한다. 이 경우에는 SVM으로 Model을 생성할 수 없다. 하지만 좌표를 3차원으로 확장할 수 있다면, 그 함수에 따라서 1.1과 같이 두 그룹을 구분하는 Plane을 생성할 수도 있다.

$$K(x_i, x_j) = \exp(-\frac{||x_i - x_j||^2}{2\sigma^2})$$

2D 좌표를 3차원으로 확장하는 여러 가지 함수가 있지만, 일반적으로 많이 사용되는 Gaussian RBF 함수를 사용할 것이다.

Chapter 2

본론

2.1 Linear SVM Classifiers

```
l_pRandomState = [20, 30, 40]
l_pC1 = [10, 1, 0.1]

def prob1():
   for row, pRandomState in enumerate(l_pRandomState):
      # data 생성
      coords, labels = datasets.make_blobs(
         n_samples=100, cluster_std=1.2, random_state=pRandomState,
             centers=2)

      plt.figure(figsize=(16, 5))
      for col, pC in enumerate(l_pC1):
         plt.subplot(1, 3, col + 1)
         ax = plt.gca()
         ax.set_title('C=a%.1f' % pC)

         # liner kernel을  이용해  SVM classifier 생성
         clf = SVC(kernel="linear", C=pC)
         clf.fit(coords, labels)

         # SVM classifier model을 통해 line  그래프   생성
         inspection.DecisionBoundaryDisplay.from_estimator(
            clf,
            coords,
            plot_method="contour",
            levels=[-1, 0, 1],
            linestyles=["--", "-", "--"],
            ax=ax,
         )
```

```
        # label에 따라 분리하여 scatter 그래프 생성
        plt.scatter(coords[labels == 0, 0], coords[labels == 0, 1])
        plt.scatter(coords[labels == 1, 0], coords[labels == 1, 1])
        plt.scatter(clf.support_vectors_[:, 0], clf.support_vectors_[
            :, 1], s=250, alpha=0.3)
    plt.show()
```

sklearn의 datasets.make_blobs을 사용하여 Training data를 생성하였다. 생성된 Training data를 통해 SVM model을 생성하고 그래프에 Line과 Support vectors를 표시해 주었다.

2.2 Non-linear SVM Classifiers

2.2.1 Create Data

```
def createDatasets():
    # data 생성
    coords, labels = datasets.make_circles(
        n_samples=100, factor=0.1, noise=0.1)

    # label에 따라 분리하여 scatter 그래프 생성
    plt.scatter(coords[labels == 0, 0], coords[labels == 0, 1])
    plt.scatter(coords[labels == 1, 0], coords[labels == 1, 1])
    plt.show()
    return [coords, labels]
```

sklearn의 datasets.make_circles을 사용하여 Training data를 생성하였다.

2.2.2 Kernel function

```
def gauss_rbf(coords):
    # 2 차원 coordinates를 gaussian rbf를 통해 3 차원으로 확장
    X = coords[:, 0]
    Y = coords[:, 1]
    Z = np.exp(-(X**2 + Y**2))
    return X, Y, Z

def kernelFunction(coords, labels):
    fig = plt.figure()
    ax = fig.add_subplot(projection='3d')
    X, Y, Z = gauss_rbf(coords)

    # label에 따라 분리하여 scatter 그래프 생성
    ax.scatter(X[labels == 0], Y[labels == 0], Z[labels == 0])
    ax.scatter(X[labels == 1], Y[labels == 1], Z[labels == 1])
    plt.show()
```

make_circles로 생성된 Training data는 Linear classifier로 class를 구분할 수 없기 때문에, Gaussian RBF 함수를 Kernel function으로 설정하여 3차원으로 확장하였다. 이때 기준이 되는 X는 원점(0, 0)이기 때문에 생략하였다.

2.2.3 Train SVM

```
def trainSVM(coords, labels):
   plt.figure(figsize=(10, 5))
   for col, pC in enumerate(l_pC2):
      plt.subplot(1, 2, col + 1)
      ax = plt.gca()
      ax.set_title('C=%.1f' % pC)

      # rbf kernel을  이용해  SVM classifier 생성
      clf = SVC(kernel="rbf", C=10)
      clf.fit(coords, labels)

      # SVM classifier model을 통해 line  그래프   생성
      inspection.DecisionBoundaryDisplay.from_estimator(
         clf,
         coords,
         plot_method="contour",
         levels=[-1, 0, 1],
         linestyles=["--", "-", "--"],
         ax=ax,
      )

      # label에  따라   분리하여   scatter  그래프   생성
      plt.scatter(coords[labels == 0, 0], coords[labels == 0, 1])
      plt.scatter(coords[labels == 1, 0], coords[labels == 1, 1])
   plt.show()
```

make_circles로 생성된 Training data는 Linear classifier로 class를 구분할 수 없기 때문에, Gaussian RBF 함수를 Kernel function으로 설정하여 SVM을 학습하였다. 학습한 Non-linear SVM classifier model을 그래프에 표시해 주었다.

Chapter 3

결론

3.1 Linear SVM Classifiers

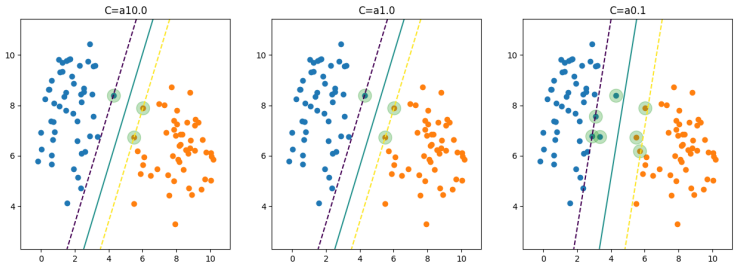

Figure 3.1: random_state=20

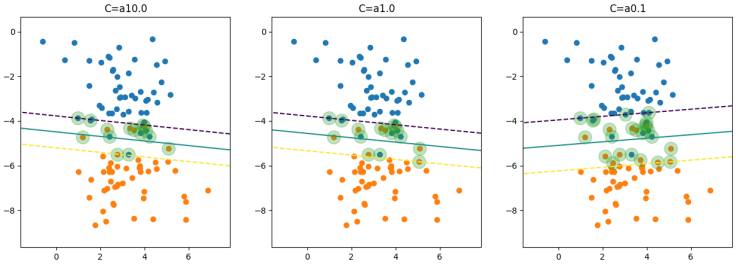

Figure 3.2: random_state=30

생성한 데이터 별로 Support vector machine을 적용하여 Model을 생성하였다. C(Misclassifications cost) 값을 크게 설정할수록 Suuport vector가 많아지는 것을

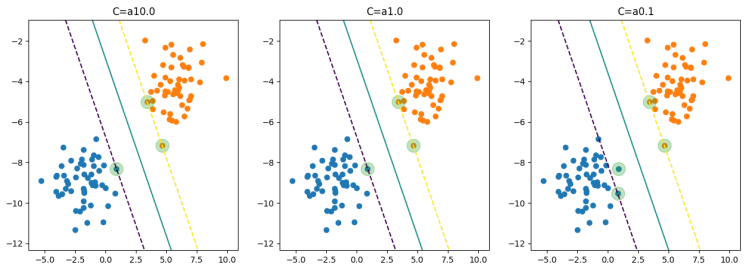

Figure 3.3: random_state=40

확인할 수 있다.

3.2 Non-linear SVM Classifiers

3.2.1 Create Data

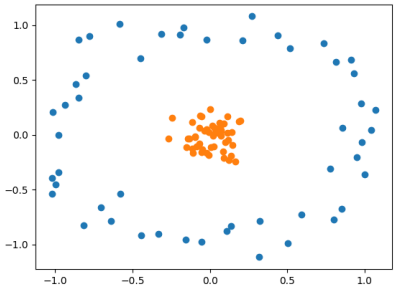

Figure 3.4: Circle data

make_circles로 생성된 Data는 class 0 (파란색)은 바깥쪽에 원형 고리 형태를 형성하고, class 1 (주황색)은 중심에 모인 형태를 형성하여 Linear classifier로는 구분할 수 없다.

3.2.2 Kernel function

Circle data에 Gaussian RBF 함수를 이용해 3차원으로 확장한 결과 plane으로 구분할 수 있는 Figure 3.5의 형태를 형성하였다.

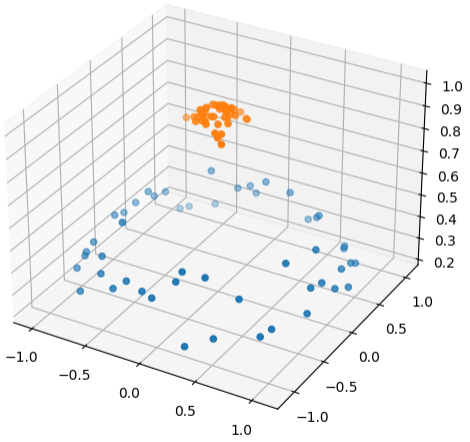

Figure 3.5: Gaussian RBF

3.2.3 Train SVM

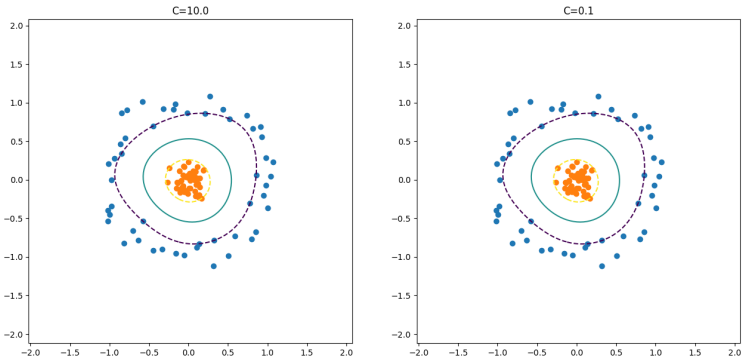

Figure 3.6: factor=0.1

Gaussian RBF 함수를 Kernel funciton으로 이용해 SVM을 통해 Non-linear classifier model을 생성할 수 있었다.