Introduction to Computer Vision HW06

- Support Vector Machine -

2021-1학기 컴퓨터비전개론 059분반 201724437 김재현

Part 1. Linear SVM Classifier

1. Create Datasets

sklearn의 datasets 패키지를 이용해 blob 데이터를 생성하는 문제입니다. 우선 필요한 color, random\_state, factor, misclassification cost 등을 모두 전역 변수로 저장한 다음 진행했습니다.

처음에는 matplotlib.pyplot의 subplot 함수를 이용해 필요한 만큼 plot을 생성하고, 주어진 random\_state 값 각각에 대해 create\_blob\_data() 함수로 데이터를 만들었습니다. 이후 이 데이터와 산점도 작성에 필요한 정보를 plot\_2D\_data() 함수로 넘겨 실제로 그래프를 그렸습니다. 이때 반복문과 boolean indexing을 통해 두 class가 각각 다른 색깔로 표시되도록 하였습니다.

|  |
| --- |
| hw06\_201724437.py - Line 6~12 |
| class\_num = 2  class\_colors\_linear = np.array(['brown', 'blue'])  class\_colors\_nonlinear = np.array(['red', 'yellow'])  random\_state\_linear = np.array([20, 30, 40])  factor\_nonlinear = np.array([0.1])  misclass\_cost\_linear = np.array([10,1,0.1])  misclass\_cost\_nonlinear = np.array([10,0.1]) |

|  |
| --- |
| hw06\_201724437.py - Line 65~74 |
| fig1, ax1= plt.subplots(1, rand\_states.shape[0], figsize=(10,3))  fig1.tight\_layout(pad=3.0)  for i, rs in enumerate(rand\_states):  X, y = create\_blob\_data(rs)  ax1[i].set\_title(f"random\_state={rs}")  plot\_2D\_data(ax1[i], X, y, class\_colors\_linear)  plt.show() |

|  |
| --- |
| hw06\_201724437.py - Line 14~18 |
| def create\_blob\_data(rand\_state):  X, y = datasets.make\_blobs(n\_samples=100, centers=2, cluster\_std=1.2,  random\_state=rand\_state)  return X, y |

|  |
| --- |
| hw06\_201724437.py - Line 25~29 |
| def plot\_2D\_data(ax, X, y, colors):  for k, clr in zip(range(class\_num), colors):  members = (y == k)  ax.scatter(X[members,0], X[members,1], s=10, facecolor=clr) |

>>

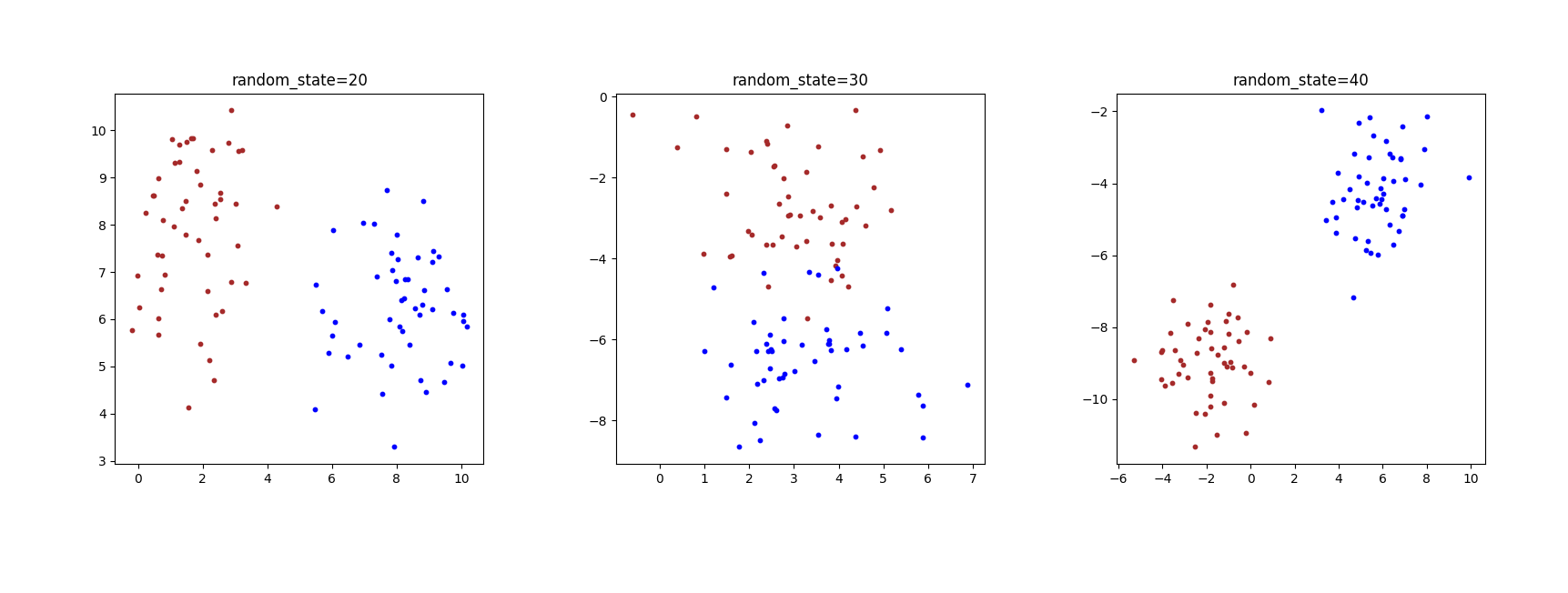


그림 1 – make\_blob\_data.png

1. Train SVM

위에서 작성한 create\_blob\_data() 함수를 이용하여 feature point 정보를 구한 뒤, SVC에 서로 다른 misclassification cost를 전달하여 각각의 classifier를 얻어냈습니다. 이렇게 얻은 classifier는 plot\_2D\_classifier 함수 내에서 np.meshgrid, SVC.decision\_function, 그리고 plt.contour 함수를 이용하여 그래프에 표시하였습니다.

이 때 misclassification cost가 작을수록 negative/positive boundary를 벗어나는 vector들이 많아지며, 따라서 margin 값 또한 늘어나는 것을 확인할 수 있습니다.

|  |
| --- |
| hw06\_201724437.py - Line 76~92 |
| fig2, ax2 = plt.subplots(rand\_states.shape[0],  misclassCost.shape[0], squeeze=False, figsize=(10,10))  fig2.tight\_layout(pad=3.0)  for i, rs in enumerate(rand\_states):  X, y = create\_blob\_data(rs)  for j, mc in enumerate(misclassCost):  clf = SVC(kernel='linear', C=mc)  clf.fit(X,y)  ax2[i,j].set\_title(f"random\_state={rs}, C={mc:.1f}")  plot\_2D\_data(ax2[i,j], X, y, class\_colors\_linear)  plot\_2D\_classifier(ax2[i,j], clf)  plt.show() |

|  |
| --- |
| hw06\_201724437.py - Line 37~61 |
| def plot\_2D\_classifier(ax, clf, sv=1):  xlim = ax.get\_xlim()  ylim = ax.get\_ylim()  xcords = np.linspace(xlim[0],xlim[1],50)  ycords = np.linspace(ylim[0],ylim[1],50)  xmeshgrid, ymeshgrid = np.meshgrid(xcords, ycords)  xymeshgrid = np.r\_[[xmeshgrid.reshape(-1)],[ymeshgrid.reshape(-1)]].T    z = clf.decision\_function(xymeshgrid).reshape(xmeshgrid.shape)  ax.contour(xmeshgrid, ymeshgrid, z, levels=[-1, 0, 1], linestyles=['--','-','--'], colors='k')  if (sv==1):  ax.scatter(clf.support\_vectors\_[:,0],clf.support\_vectors\_[:,1],  s=100, facecolors='none', edgecolors='k') |

>>

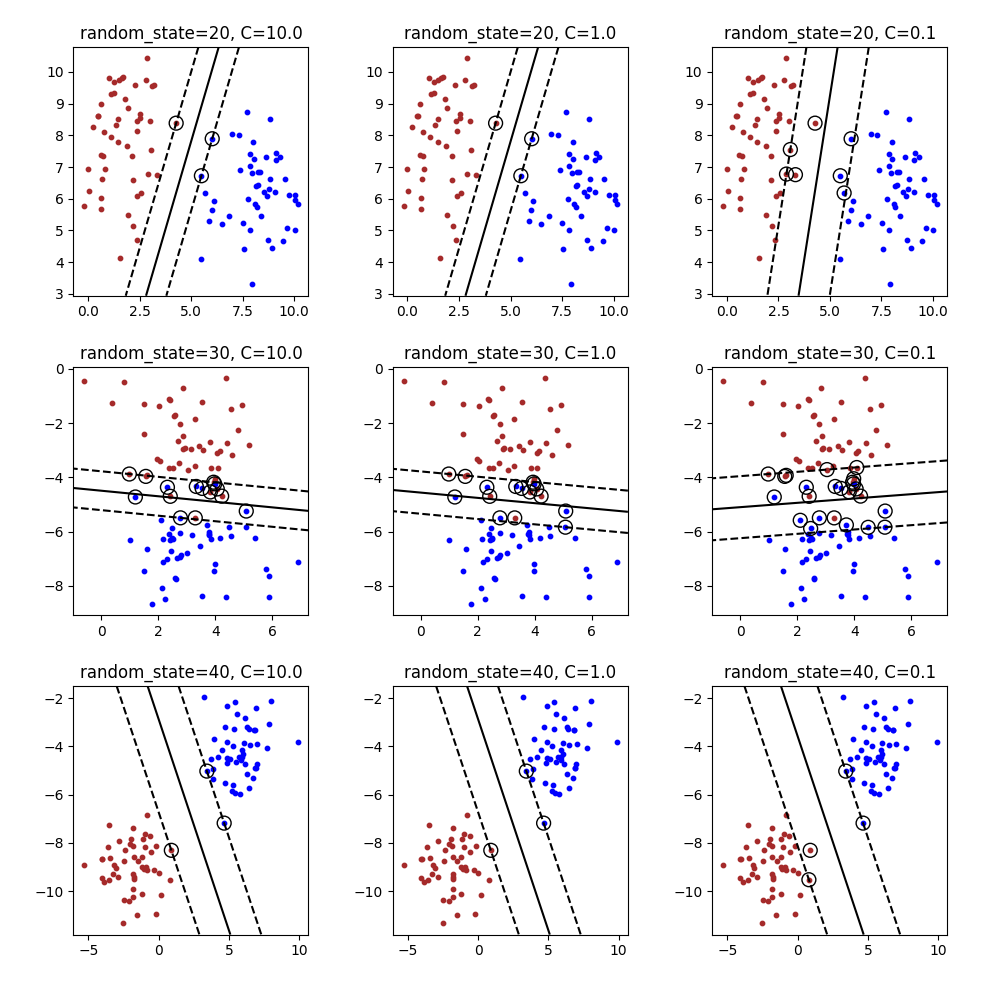


그림 2 – Linear\_SVM\_Results.png

Part 2. Nonlinear SVM

1. Create Datasets

이번에는 make\_circles 함수를 이용하여 data를 생성하고, 이를 2차원 산점도로 표현해야 합니다. Part 1에서와 마찬가지로 plot\_2d\_data 함수를 재사용하여 그래프로 나타내었으며, 이는 그림 3과 같습니다.

|  |
| --- |
| hw06\_201724437.py - Line 95~103 |
| fig1, ax1 = plt.subplots(1,factors.shape[0], figsize=(5,5))  for i, factor in enumerate(factors):  X, y = create\_circle\_data(factor)  ax1.set\_title(f"factor={factor}")  plot\_2D\_data(ax1,X,y, class\_colors\_nonlinear)  plt.show() |

|  |
| --- |
| hw06\_201724437.py - Line 20~23 |
| def create\_circle\_data(fc):  X, y = datasets.make\_circles(n\_samples=100, factor=fc, noise=0.1)  return X,y |

>>

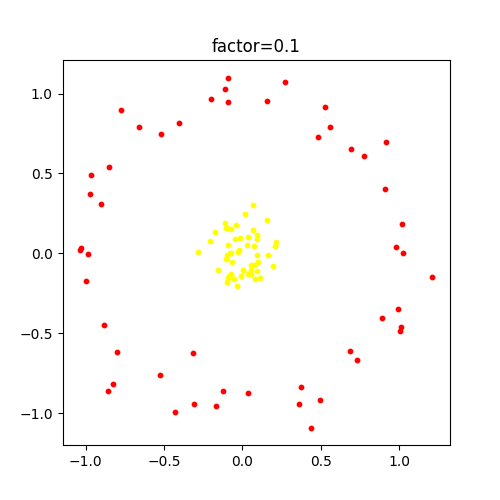


그림 3 – make\_circles\_data\_2d.png

1. Kernel function

위에서 확인한 바와 같이 이 데이터는 단순한 linear classifier로는 완전히 두 클래스를 분리할 수 없습니다. 따라서 Gaussian RBF kernel function을 사용해서 분리 가능한지 살펴볼 필요가 있습니다. 우선 이를 시각화하기 위해 plot\_3D\_data 함수에서는 기존 2차원 좌표 X에 RBF 함수를 적용하여 새로운 좌표값을 얻고, 이를 scatter 함수에 전달하여 3차원 그래프를 그렸습니다. 이는 그림 4와 같으며, 여기서 Gaussian RBF를 적용한 hyper-plane을 이용하여 분리가 가능하다는 것을 확인할 수 있습니다.

|  |
| --- |
| hw06\_201724437.py - Line 105~113 |
| fig2, ax2 = plt.subplots(1,factors.shape[0], figsize=(5,5))  ax2 = plt.subplot(projection='3d')  for i, factor in enumerate(factors):  X, y = create\_circle\_data(factor)  ax2.set\_title(f"factor={factor}")  plot\_3D\_data(ax2,X,y, class\_colors\_nonlinear)  plt.show() |

|  |
| --- |
| hw06\_201724437.py - Line 31~35 |
| def plot\_3D\_data(ax, X, y, colors):  rbf = np.exp(-(X\*\*2).sum(1))  for k, clr in zip(range(class\_num), colors):  members = (y == k)  ax.scatter(X[members,0], X[members,1], rbf[members], s=10, facecolor=clr) |

>>

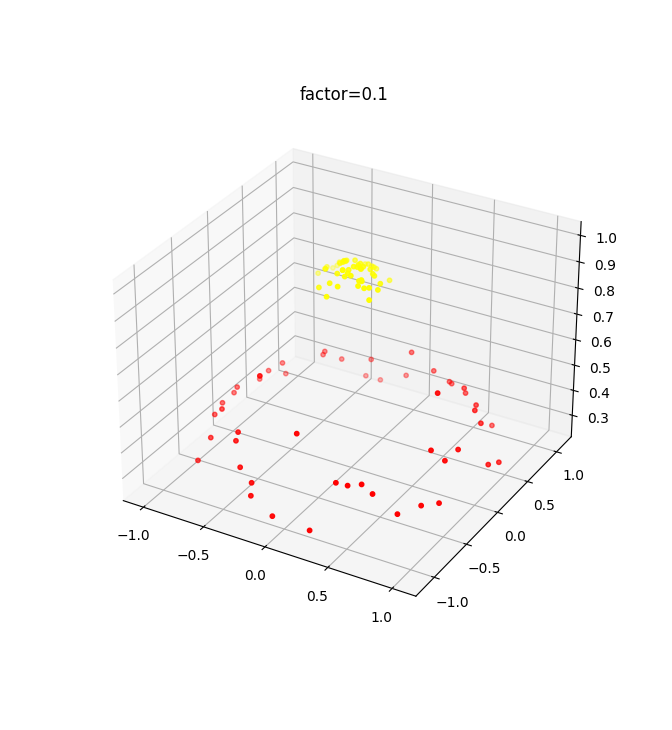


그림 4 – make\_circles\_data\_3d.png

1. Train SVM

마지막으로 rbf kernel을 이용해 classifier를 학습시키고, 여기서 얻은 decision boundary를 plot\_2D\_classifier 함수를 재사용하여 표시하였습니다. Part 1- 2에서와 마찬가지로, misclassification cost(C)가 커질수록 positive/negative boundary를 벗어나는 vector들이 많아지고, margin 값도 훨씬 늘어난 것을 확인할 수 있습니다.

|  |
| --- |
| hw06\_201724437.py - Line 115~129 |
| fig3, ax3 = plt.subplots(factors.shape[0], misclassCost.shape[0], squeeze=False, figsize=(10,5))  fig3.tight\_layout(pad=3.0)  for i, fc in enumerate(factors):  X, y = create\_circle\_data(fc)  for j, mc in enumerate(misclassCost):  clf = SVC(kernel='rbf', C=mc)  clf.fit(X,y)  ax3[i,j].set\_title(f"factor={fc}, C={mc:.1f}")  plot\_2D\_data(ax3[i,j], X, y, class\_colors\_nonlinear)  plot\_2D\_classifier(ax3[i,j], clf, sv=0)  plt.show() |

>>

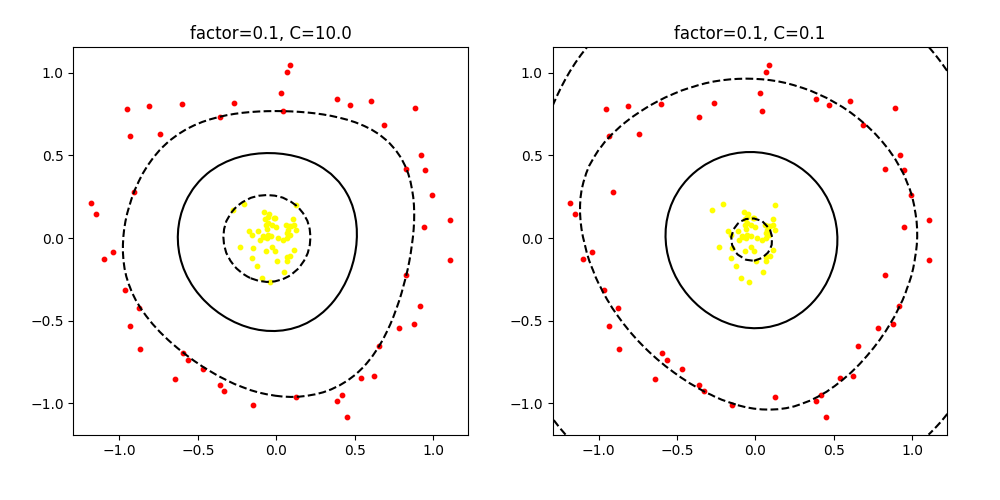


그림 5 – Nonlinear\_SVM\_result.png