In [1]:

```
import numpy as np
import plotly.express as px
import pandas as pd
import plotly.graph_objects as go
import seaborn as sns
import matplotlib.pyplot as plt
import math
import scipy as sp
from sklearn.model_selection import train_test_split
from sklearn import svm
from sklearn import datasets
from sklearn.metrics import zero_one_loss
```

Generating 2D training dataset with 2 classes¶

In [35]:

```
mu_1=[-2,0]
mu_2 = [0,2]
cov_1 = [[1, -0.25],[-0.25, 1]]
cov_2 = [[1, -0.25],[-0.25, 1]]
num_samples= 500

w1 =  np.random.multivariate_normal(mu_1, cov_1, num_samples)
w2 =  np.random.multivariate_normal(mu_2, cov_2, num_samples)

label1 = np.zeros(num_samples).reshape((num_samples,1))
label2 = np.zeros(num_samples).reshape((num_samples,1)) +1

w1 = np.append(w1, label1, axis=1)
w2 = np.append(w2, label2, axis=1)



#prior probability is the same for each class
prior_p=500/1000
#concatenate whole samples in one array
train_dataset=np.concatenate([w1,w2])
```

```
df = pd.DataFrame(train_dataset, columns=['x1', 'x2', 'label'])
df = df.sample(frac=1).reset_index(drop=True) # shuffle the dataframe in-
place and reset the index
df
```

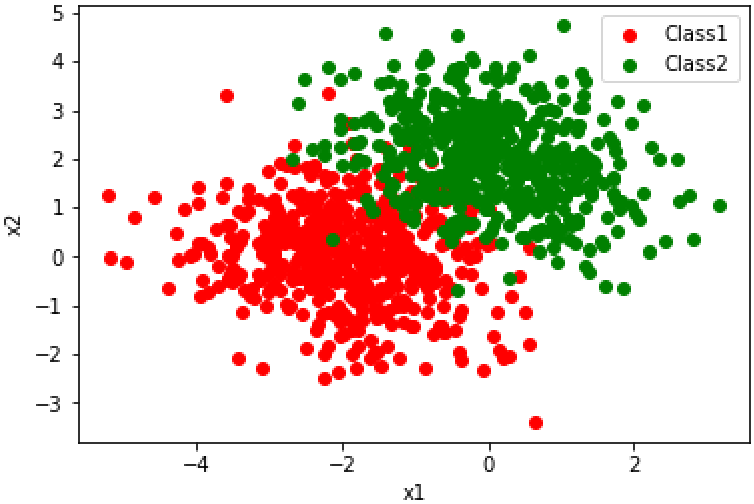
Out[35]:

	x1	x2	label
0	-0.380980	3.884836	1.0
1	0.270876	1.778541	1.0
2	-2.453738	1.025094	0.0
3	-2.438149	-0.028350	0.0
4	0.424334	1.802717	1.0
...	...	...	...
995	-2.945962	0.282230	0.0
996	-0.852222	4.062818	1.0
997	-2.411784	0.366154	0.0
998	-0.780021	1.435980	1.0
999	0.020570	2.387932	1.0

1000 rows × 3 columns

In [36]:

```
figure1 = plt.figure()
plt.scatter(w1[:,0], w1[:,1], color='r', label='Class1')
plt.scatter(w2[:,0], w2[:,1], color='g', label='Class2')
# plt.rcParams['figure.figsize'] = [20, 20]
plt.xlabel('x1')
plt.ylabel('x2')
plt.legend()
# plt.grid()
plt.show()
```



In [37]:
```
df
X = df.iloc[:,0:2].values
y = df.iloc[:,-1].values
```

Creating training and testing datasets with 80:20 split

In [38]:
```
X_train, X_test, y_train, y_test = train_test_split(X, y, train_size=0.8,
random_state=1)
```

In [39]:
```
X_train.shape
```

Out[39]: (800, 2)

Part A: 10% of the training data is labeled

In [40]:
```
#10% of the training data is labeled
X_train, X_unl, y_train, _ = train_test_split(X_train, y_train,
train_size=0.1, random_state=1)
```

In [41]:
```
X_unl.shape
```

Out[41]: (720, 2)

In [42]:
```
X_unl_df = pd.DataFrame(X_unl, columns=['x1', 'x2'])
X_unl_df
```

Out[42]:

	x1	x2

	x1	x2
0	-0.162472	1.843025
1	-2.202430	-1.832835
2	-0.392301	-1.230569
3	-3.410816	-0.358632
4	-0.495970	1.463370
...	...	...
715	1.142499	0.650258
716	-0.509620	1.222140
717	1.309253	3.330639
718	-4.041126	0.053144
719	-1.401420	0.298887

720 rows × 2 columns

1. Training on the labeled dataset

In [43]:

```
clf = svm.SVC(kernel='linear', probability=True, C=1.0).fit(X_train, y_train)
print ('Accuracy: ',clf.score(X_test, y_test), ' error: ', 1 - clf.score(X_test, y_test) )
```

```
Accuracy:  0.97  error:  0.030000000000000027
```

2. Make a prediction using the unlabeled datset (x_unl)

- ### Using predict_prob() to find the probability of each sample

In [44]:

```
#find the probability of each class
clp= clf.predict_proba(X_unl)
clf_prob = pd.DataFrame(clp, columns = ['class1', 'class2'])
# predict the the label of each class
lab=clf.predict(X_unl)
#find the max probability
clf_prob["max"] = clf_prob.max(axis = 1)
clf_prob["label"] = lab
clf_prob
```

Out[44]:

	class1	class2	max	label

	class1	class2	max	label
0	6.080655e-02	0.939193	0.939193	1.0
1	9.984301e-01	0.001570	0.998430	0.0
2	9.049883e-01	0.095012	0.904988	0.0
3	9.984551e-01	0.001545	0.998455	0.0
4	1.732832e-01	0.826717	0.826717	1.0
...	...	...	...	...
715	3.397656e-02	0.966023	0.966023	1.0
716	2.353276e-01	0.764672	0.764672	1.0
717	6.985512e-07	0.999999	0.999999	1.0
718	9.990992e-01	0.000901	0.999099	0.0
719	8.610110e-01	0.138989	0.861011	0.0

720 rows × 4 columns

3. Choose the samples in X_unl with high confidence and add them into the labeled dataset

In [45]:

```
th = 0.6
clf_prob[clf_prob["max"] > th]
```

Out[45]:

	class1	class2	max	label
0	6.080655e-02	0.939193	0.939193	1.0
1	9.984301e-01	0.001570	0.998430	0.0
2	9.049883e-01	0.095012	0.904988	0.0
3	9.984551e-01	0.001545	0.998455	0.0
4	1.732832e-01	0.826717	0.826717	1.0
...	...	...	...	...
715	3.397656e-02	0.966023	0.966023	1.0
716	2.353276e-01	0.764672	0.764672	1.0
717	6.985512e-07	0.999999	0.999999	1.0
718	9.990992e-01	0.000901	0.999099	0.0
719	8.610110e-01	0.138989	0.861011	0.0

679 rows × 4 columns

In [46]:

```
pseudo_lab_size =len(X_unl[clf_prob["max"] > th])
pseudo_lab_size
```

Out[46]: 679

In [47]:
```
#add the predicted labels to the training dataset
X_train_new = np.append(X_train, X_unl[clf_prob["max"] > th], axis=0)
y_train_new = np.append(y_train, clf_prob['label'][clf_prob["max"] >
th].values, axis=0)

X_train = X_train_new
y_train = y_train_new
```

In [48]:
```
#remove the added labels from the unlabled dataset
X_unl_df = X_unl_df.drop(X_unl_df[clf_prob["max"] >
th].index).reset_index(drop=True)
#update the unlabeled set
X_unl = X_unl_df.values
# X_unl_df
```

4. Repeat

In [49]:
```
score_ls = []
while len(X_unl) != 0 and pseudo_lab_size != 0: # stop when there are no
more unlabeled data or when we are no confident about the data
    #Step 1
    clf = svm.SVC(kernel='linear', probability=True,C=1).fit(X_train,
y_train)
    score_ls.append(clf.score(X_test, y_test))
    print ('Accuracy: ',clf.score(X_test, y_test), ' error: ', 1 -
clf.score(X_test, y_test) )
#     print(len(X_unl))

    #Step2
    #find the probability of each class
    clp= clf.predict_proba(X_unl)
    clf_prob = pd.DataFrame(clp, columns = ['class1', 'class2'])
    # predict the the label of each class
    lab=clf.predict(X_unl)
```

```
    clf_prob["max"] = clf_prob.max(axis = 1)
    clf_prob["label"] = lab

    #Step3
    pseudo_lab_size =len(X_unl[clf_prob["max"] > th])
    X_train_new = np.append(X_train, X_unl[clf_prob["max"] > th], axis=0)
    y_train_new = np.append(y_train, clf_prob['label'][clf_prob["max"] >
th].values, axis=0)
    X_train = X_train_new
    y_train = y_train_new



    X_unl_df = X_unl_df.drop(X_unl_df[clf_prob["max"] >
th].index).reset_index(drop=True)
    X_unl = X_unl_df.values
```

```
Accuracy:  0.97  error:  0.030000000000000027
Accuracy:  0.975  error:  0.025000000000000022
Accuracy:  0.975  error:  0.025000000000000022
```

When 10% of the training data is labeled, the error of the self-training algorithm on the testing data with SVM classifier is as follows:

1. The first time the classifier is trained using only the labeled -----> accuracy: 0.97 error: 0.030000000000000027
2. At least one time point during the self training process -----> accuracy: 0.97 error: 0.030000000000000027
3. After self-training is completed -----> accuracy: 0.975 error: 0.025000000000000022

From the above results, we can see that there is no significant boost in performance. This confirms the lecture's discussion that SSL is not always going to work or give you a significant boost in performance.

In [ ]:

```

```

Part B: 25% of the training data is labeled

In [50]:

```
X_train, X_test, y_train, y_test = train_test_split(X, y, train_size=0.8,
random_state=1)
#25% of the training data is labeled
```

```
X_train, X_unl, y_train, _ = train_test_split(X_train, y_train,
train_size=0.25, random_state=1)
```

In [53]:

```
X_train.shape
```

Out[53]:

```
(200, 2)
```

In [54]:

```
X_unl_df = pd.DataFrame(X_unl, columns=['x1', 'x2'])
X_unl_df
```

Out[54]:

	x1	x2
0	-0.162472	1.843025
1	-2.202430	-1.832835
2	-0.392301	-1.230569
3	-3.410816	-0.358632
4	-0.495970	1.463370
...	...	...
595	0.320025	-0.795005
596	1.238727	2.093136
597	-0.269999	0.602040
598	-1.398507	0.951005
599	-2.549455	0.151460

600 rows × 2 columns

1. Training on the labeled dataset

In [55]:

```
clf = svm.SVC(kernel='linear', probability=True, C=1.0).fit(X_train,
y_train)
print ('Accuracy: ',clf.score(X_test, y_test), ' error: ', 1 -
clf.score(X_test, y_test) )
```

```
Accuracy:  0.965  error:  0.035000000000000003
```

2. Make a prediction using the unlabeled datset (x_unl)

- ### Using predict_prob() to find the probability of each sample

In [56]:

```
#find the probability of each class
clp= clf.predict_proba(X_unl)
```

```
clf_prob = pd.DataFrame(clp, columns = ['class1', 'class2'])
# predict the the label of each class
lab=clf.predict(X_unl)
#find the max probability
clf_prob["max"] = clf_prob.max(axis = 1)
clf_prob["label"] = lab
clf_prob
```

Out[56]:

	class1	class2	max	label
0	0.029395	0.970605	0.970605	1.0
1	0.999555	0.000445	0.999555	0.0
2	0.931709	0.068291	0.931709	0.0
3	0.999550	0.000450	0.999550	0.0
4	0.112255	0.887745	0.887745	1.0
...	...	...	...	...
595	0.560227	0.439773	0.560227	0.0
596	0.000002	0.999998	0.999998	1.0
597	0.269522	0.730478	0.730478	1.0
598	0.703817	0.296183	0.703817	0.0
599	0.992334	0.007666	0.992334	0.0

600 rows × 4 columns

3. Choose the samples in X_unl with high confidence and add them into the labeled dataset

In [57]:

```
th = 0.6
clf_prob[clf_prob["max"] > th]
```

Out[57]:

	class1	class2	max	label
0	0.029395	0.970605	0.970605	1.0
1	0.999555	0.000445	0.999555	0.0
2	0.931709	0.068291	0.931709	0.0
3	0.999550	0.000450	0.999550	0.0
4	0.112255	0.887745	0.887745	1.0
...	...	...	...	...
594	0.005475	0.994525	0.994525	1.0
596	0.000002	0.999998	0.999998	1.0

	class1	class2	max	label
597	0.269522	0.730478	0.730478	1.0
598	0.703817	0.296183	0.703817	0.0
599	0.992334	0.007666	0.992334	0.0

569 rows × 4 columns

In [58]:

```
pseudo_lab_size =len(X_unl[clf_prob["max"] > th])
#add the predicted labels to the training dataset
X_train_new = np.append(X_train, X_unl[clf_prob["max"] > th], axis=0)
y_train_new = np.append(y_train, clf_prob['label'][clf_prob["max"] >
th].values, axis=0)

X_train = X_train_new
y_train = y_train_new
#remove the added labels from the unlabled dataset
X_unl_df = X_unl_df.drop(X_unl_df[clf_prob["max"] >
th].index).reset_index(drop=True)
#update the unlabeled set
X_unl = X_unl_df.values
# X_unl_df
```

4. Repeat

In [59]:

```
score_ls = []
while len(X_unl) != 0 and pseudo_lab_size != 0: # stop when there are no
more unlabeled data or when we are no confident about the data
    #Step 1
    clf = svm.SVC(kernel='linear', probability=True,C=1).fit(X_train,
y_train)
    score_ls.append(clf.score(X_test, y_test))
    print ('Accuracy: ',clf.score(X_test, y_test), ' error: ', 1 -
clf.score(X_test, y_test) )
#     print(len(X_unl))

    #Step2
    #find the probability of each class
```

```
    clp= clf.predict_proba(X_unl)
    clf_prob = pd.DataFrame(clp, columns = ['class1', 'class2'])
    # predict the the label of each class
    lab=clf.predict(X_unl)
    clf_prob["max"] = clf_prob.max(axis = 1)
    clf_prob["label"] = lab

    #Step3
    pseudo_lab_size =len(X_unl[clf_prob["max"] > th])
    X_train_new = np.append(X_train, X_unl[clf_prob["max"] > th], axis=0)
    y_train_new = np.append(y_train, clf_prob['label'][clf_prob["max"] >
th].values, axis=0)
    X_train = X_train_new
    y_train = y_train_new



    X_unl_df = X_unl_df.drop(X_unl_df[clf_prob["max"] >
th].index).reset_index(drop=True)
    X_unl = X_unl_df.values
```

```
Accuracy:  0.97  error:  0.030000000000000027
Accuracy:  0.975  error:  0.025000000000000022
Accuracy:  0.975  error:  0.025000000000000022
```

When 25% of the training data is labeled, the error of the self-training algorithm on the testing data with SVM classifier is as follows:¶

1. The first time the classifier is trained using only the labeled -----> accuracy: 0.965 error: 0.03500000000000003
2. At least one time point during the self training process -----> accuracy: 0.97 error: 0.030000000000000027
3. After self-training is completed -----> accuracy: 0.975 error: 0.025000000000000022

From the above results, we can see similar trend to the previous results. That is there is no significant boost in performance.

In [ ]:

```
```

In [ ]:

```
```