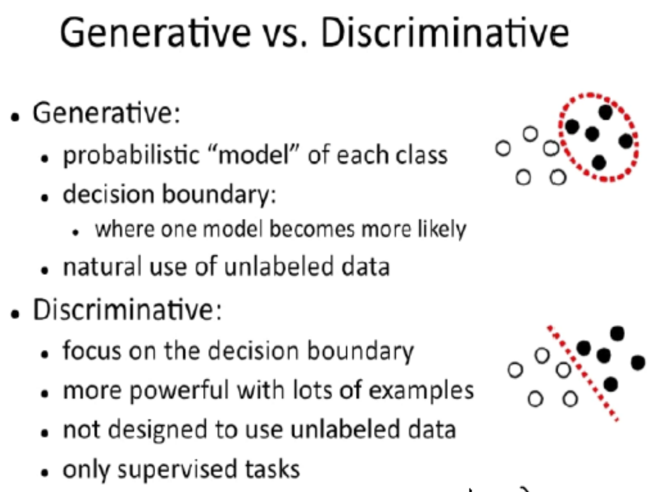
Support Vector Machine

Objective ÷. To find the best splitting line between the data points

- used in discriminative classification rather than generative classification.



Generative vs. Discriminative

- Generative:
  - probabilistic "model" of each class
  - decision boundary:
    - where one model becomes more likely
  - natural use of unlabeled data
- Discriminative:
  - focus on the decision boundary
  - more powerful with lots of examples
  - not designed to use unlabeled data
  - only supervised tasks

- we can find a curve or line (two dimensions) or we can find many lines (multiple dimensions) that divides classes

usage ÷ SVM is used both in classification and Regression

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from scipy import stats
import seaborn as sns; sns.set()
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from matplotlib.cm import rainbow
# Machine Learning
from sklearn.neighbors import KNeighborsClassifier
from sklearn.svm import SVC
from sklearn import svm
from sklearn import metrics #Import scikit-learn metrics module for accuracy calculation
from sklearn.metrics import confusion_matrix
from sklearn.datasets.samples_generator import make_blobs
dataset_heart = pd.read_csv('heart.csv')


dataset_heart = pd.get_dummies(dataset_heart,
                     columns = ['sex', 'cp', 'fbs', 'restecg', 'exang', 'slope', 'ca', 'thal'])
standardScaler = StandardScaler()
columns_to_scale = ['age', 'trestbps', 'chol', 'thalach', 'oldpeak']
dataset_heart[columns_to_scale] = standardScaler.fit_transform(dataset_heart[columns_to_scale])
```

```
# Correlation map
f,ax = plt.subplots(figsize=(12,12))
sns.heatmap(dataset_heart.corr(), annot=True, linewidths=.5, fmt= '.1f',ax=ax)
plt.show()
corr = dataset_heart.corr()
dataset_heart.hist()
columns = np.full((corr.shape[0],), True, dtype=bool)
for i in range(corr.shape[0]):
    for j in range(i+1, corr.shape[0]):
        if corr.iloc[i,j] >= 0.6:
            if columns[j]:
                columns[j] = False
selected_columns = dataset_heart.columns[columns]
dataset_heart = dataset_heart[selected_columns]


X = dataset_heart.drop('target',axis = 1)
y = dataset_heart.target

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.33,
                                                    random_state = 0)
'''
#Create a svm Classifier
ml = svm.SVC(kernel='linear') # Linear Kernel
#Train the model using the training sets
ml.fit(X_train, y_train)
#Predict the response for test dataset
y_pred = ml.predict(X_test)
# Model Accuracy: how often is the classifier correct?
acc=ml.score(X_test,y_test)
print(acc*100)
confusion_matrix(y_test,y_pred)
'''

svc_scores = []
kernels = ['linear', 'poly', 'rbf', 'sigmoid']
for i in range(len(kernels)):
    svc_classifier = svm.SVC(kernel = kernels[i])
    svc_classifier.fit(X_train, y_train)
    svc_scores.append(svc_classifier.score(X_test, y_test))

colors = rainbow(np.linspace(0, 1, len(kernels)))
plt.bar(kernels, svc_scores, color = colors)
for i in range(len(kernels)):
    plt.text(i, svc_scores[i], svc_scores[i])
plt.xlabel('Kernels')
plt.ylabel('Scores')
plt.title('Support Vector Classifier scores for different kernels')
```