**TRƯỜNG ĐẠI HỌC SƯ PHẠM KỸ THUẬT TP.HCM**

**KHOA ĐÀO TẠO CHẤT LƯỢNG CAO**

NGÀNH CÔNG NGHỆ THÔNG TIN

&



**BÁO CÁO ĐỒ ÁN 2**

Giáo viên hướng dẫn**: Thầy Trần Nhật Quang**

Sinh viên thực hiện**: Nguyễn Đức Hiếu**

**Phạm Ngọc Hải**

1. MỞ ĐẦU
   1. Tìm hiểu về đạo văn:

* Theo Wikipedia, đạo văn được định nghĩa là “chiếm hữu một cách sai trái” và “ăn cắp và công bố” “ngôn ngữ, suy nghĩ, ý tưởng, hay cách diễn đạt” của người khác và xem chúng như là những gì do mình nghĩ ra. Khái niệm đạo văn vẫn chưa có những định nghĩa và quy tắc rõ ràng.
* Đạo văn được xem là hành vi thiếu trung thực, vi phạm đạo đức nghiêm trọng và được xem là một hành vi không thể chấp nhận được trong giới tri thức.
* Tương tự như vậy, đối với ngành công nghệ thông tin, đạo văn là việc mà sinh viên cũng không nên thực hiện.
* Để trách để “đạo văn”, chúng ta – học sinh, sinh viên nên:

+ Thay đổi quan điểm về việc trích dẫn tài liệu: chúng ta có thể thể hiện ý tưởng, thông tin bằng câu văn, từ ngữ riêng của bản thân và phải ghi nhận nguồn tài liệu trích dẫn cho TOÀN BỘ ý tưởng tham khảo từ các văn bản gốc.

+ Tìm hiểu, nắm bắt và hiểu rõ về phương pháp và quy tắc khi trích dẫn tài liệu để trách mắc lỗi “đạo văn”.

* Sau cùng là lời cam kết của các thành viên thực hiện báo cáo đồ án này về vấn đề đạo văn. Chúng tôi cam đoan đồ án này do chính chúng tôi thực hiện. Chúng tôi không sao chép, sử dụng bất cứ tài liệu, mã nguồn nào của người khác mà không ghi rõ nguồn gốc. Chúng tôi xin chịu hoàn toàn trách nhiệm nếu vi phạm.
  1. Tìm hiểu lý thuyết
* Toàn bộ nội dung của đồ án môn học máy là tổng hợp lý thuyết các phần được giảng dạy và tiếp thu từ giáo viên hướng dẫn về Đồ án 2 (Machine Learning).
* Nội dung trọng tâm là:

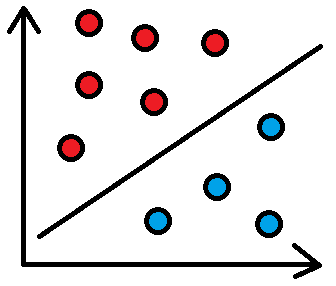
+ Thuật toán Kernel

+ Thuật toán Kmeans

* Áp dụng của các thuật toán này vào các bài toán đơn giản.

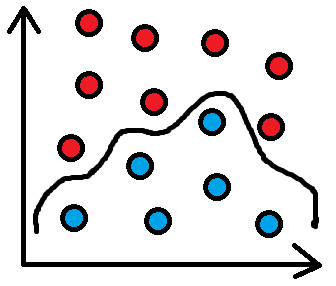
1. CÁC THUẬT TOÁN VÀ ỨNG DỤNG
   1. Kernel SVM và Kernel Function
      1. Khái niệm

* Support Vector Machine (SVM) là một mô hình của học có giám sát( Supervised learning models) với thuật toán học tập liên quan để phân tích dữ liệu
* SVM là một phân loại chính thức đươc xác định bởi một siêu phẳng(hyperplane) tách biệt. Một siêu mặt phẳng là một không gian con(subspace) nhỏ hơn một chiều của không gian xung quanh(ambient space) nó . Chiều của không gian toán học(hoặc vật thể) được định nghĩa không chính thức là số lượng tọa độ tối thiểu( x, z, y) cần thiết để xác định điểm bất kì mà bên trong nó trong khi không gian xung quanh bao bởi một đối tượng toán học, một đối tượng toán học là một đối tượng trừu tượng phát sinh trong toán học. Một đối tượng trừu tượng là một đối tượng không tồn tại tại bất kì thời điểm hoặc địa điểm cụ thể nào mà tồn tại như một loại sự vật(trong ý tưởng hoặc trừu tượng hóa)



VD: mặt siêu phẳng trong hình VD sau là 1 chiều không gian đường thẳng chia các điểm đỏ là xanh ra làm 2 vùng riêng biệt

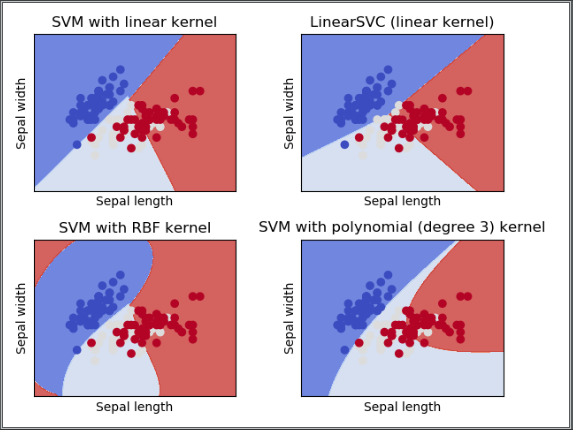
* + 1. Tại sao lại dùng thuật toán Kernel ?
* Xét ví dụ sau:



* Ta nhận thấy được không thể giải được bài toán này bằng thuật toán tuyến tính vì các điểm đỏ và xanh không thể chia ra được bởi đường thẳng vì chúng chúng được phân phối 1 cách ngẫu nhiên(randomly distributed).Trong thực tế, các dữ liệu điều phân phối ngẫu nhiên
* Trong Học máy(Machine Learning) ta sử dụng phân loại tuyến tính(linear) để giải quyết vấn đề phi tuyến tính(non-linear)
* Kernel Function là những gì được áp dụng trên mỗi dữ liệu để ánh xạ các quan sát phi tuyến tính gốc vào một không gian có chiều lớn hơn mà ở đó chúng trở nên tách rời
  + 1. Thuật toán
    2. Áp dụng
* Bài Toán: Áp dụng thuật toán Kernel để tiến hành phân biệt loài hoa Iris
* Coding:

import numpy as np  
from sklearn import svm, datasets  
import matplotlib.pyplot as plt  
  
def Lay\_Du\_Lieu():  
 # Lấy dữ liệu hoa Iris từ dataset  
 iris = datasets.load\_iris()  
  
 return iris  
  
def Ve\_So\_Do\_2D(iris):  
 X = iris.data[:, :2] # Lấy 2 đặc tính  
 Y = iris.target  
 X\_min, X\_max = X[:, 0].min() - .5, X[:, 0].max() + .5  
 Y\_min, Y\_max = X[:, 1].min() - .5, X[:, 1].max() + .5  
 plt.figure(2, figsize=(8, 6))  
 plt.clf()  
  
 plt.scatter(X[:, 0], X[:, 1], c=Y, cmap=plt.cm.Paired)  
 plt.xlabel('Sepal length')  
 plt.ylabel('Sepal width')  
  
 plt.xlim(X\_min, X\_max)  
 plt.ylim(Y\_min, Y\_max)  
 plt.xticks(())  
 plt.yticks(())  
 plt.show()  
  
def Ve\_So\_Do\_SVM(iris):  
 X = iris.data[:, :2] # Lấy 2 thuộc tính đầu tiên của dataset  
 y = iris.target  
  
 h = .01 # độ mỏng của Lưới tọa độ(Mesh) trên đồ thị  
  
 # Tạo 1 thực thể của SVM và làm nó vừa với dữ liệu  
 C = 1.0 # tham số chính quy của SVM  
  
 # Linear  
 svc = svm.SVC(kernel='linear', C=C).fit(X, y)  
 # Radial Basic Function  
 rbf\_svc = svm.SVC(kernel='rbf', gamma=0.7, C=C).fit(X, y)  
 # Polynomial  
 poly\_svc = svm.SVC(kernel='poly', degree=3, C=C).fit(X, y)  
 #Linear SVC  
 lin\_svc = svm.LinearSVC(C=C).fit(X, y)  
  
 # Tạo lưới để vẽ sơ đồ  
 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))  
  
 # Tiêu đề cho các sơ đồ  
 Tieu\_De = ['SVM with linear kernel',  
 'LinearSVC (linear kernel)',  
 'SVM with RBF kernel',  
 'SVM with polynomial (degree 3) kernel']  
  
 for i, clf in enumerate((svc, lin\_svc, rbf\_svc, poly\_svc)):  
 # Vẽ sơ đồ ranh giới  
 plt.subplot(2, 2, i + 1)  
 plt.subplots\_adjust(wspace=0.4, hspace=0.4)  
  
 Z = clf.predict(np.c\_[xx.ravel(), yy.ravel()])  
  
 # Trả kết quả vào sơ đồ  
 Z = Z.reshape(xx.shape)  
 plt.contourf(xx, yy, Z, cmap=plt.cm.coolwarm, alpha=0.8)  
 plt.scatter(X[:, 0], X[:, 1], c=y, cmap=plt.cm.coolwarm)  
 # Chiều dài đài hoa  
 plt.xlabel('Sepal length')  
 # Chiều rộng đài hoa  
 plt.ylabel('Sepal width')  
 plt.xlim(xx.min(), xx.max())  
 plt.ylim(yy.min(), yy.max())  
 plt.xticks(())  
 plt.yticks(())  
 plt.title(Tieu\_De[i])  
  
 plt.show()  
  
if \_\_name\_\_ == "\_\_main\_\_":  
 iris = Lay\_Du\_Lieu()  
 if iris is not None:  
 Ve\_So\_Do\_SVM(iris

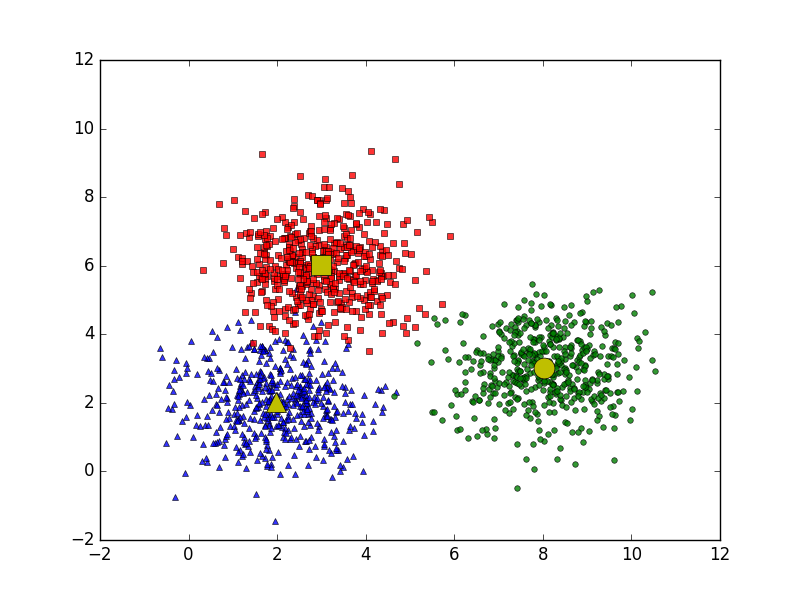
* Kết quả:



Sơ đồ phân loại hoa bằng Kernel thông qua 4 mô hình

* 1. K-means clustering:
     1. Khái niệm:
* Đây là một trong những thuật toán cơ bản trong Unsupervised Learning (học không giám sát). Trong thuật
* Trong thuật toán này chúng ta không biết được nhãn (label) của từng điểm dữ liệu. Và việc cần làm là phân các dữ liệu này thành cụm (cluster) khác nhau sao cho mỗi cụm chứa các dữ liệu có tính chất giống nhau.
  + 1. Giải thuật:

Bài toán ví dụ:



* Theo cách nhìn hình học, ta có thể viết lại bài toán dưới dạng hình thức :

Dữ liệu: tập dữ liệu X gồm n điểm có d chiều

Nhiệm vụ: phân tập dữ liệu ra làm k cụm các dữ liệu tương tự nhau và k<n.

Về lý thuyết:

Bước 1: Chọn ngẫu nhiên k điểm bất kì làm điểm trọng tâm (như hình trên là k=3).

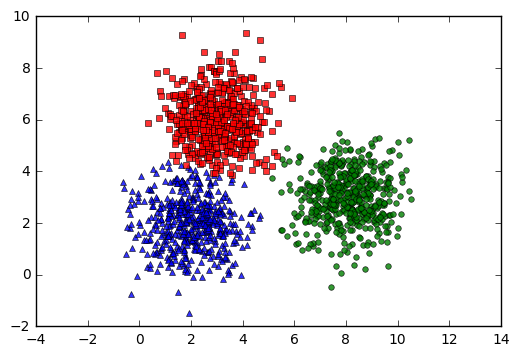
Bước 2: Nhóm dữ liệu: Nhóm mỗi đầu dữ liệu vào 1 cụm có điểm trung tâm gần nhất với nó. Nếu các cụm sau khi nhóm không thay đổi so với trước khi nhóm thì ta dừng giải thuật.

Bước 3: Cập nhật trung tâm: Với mỗi cụm sau khi nhóm lại, ta cập nhật lại điểm trung tâm của chúng bằng cách lấy trung bình cộng và quay lại bước 2 (Vòng lặp).

iii. Áp dụng:

Code:

**from** \_\_future\_\_ **import** print\_function  
**import** numpy **as** np  
**import** matplotlib.pyplot **as** plt  
**from** scipy.spatial.distance **import** cdist  
np.random.seed(11)  
 *#tạo dữ liệu lấy điểm theo phân phối chuẩn có kì vọng tại 3 điểm mean, mỗi cluster 500 điểm*means = [[2, 2], [8, 3], [3, 6]]  
cov = [[1, 0], [0, 1]]  
N = 500  
X0 = np.random.multivariate\_normal(means[0], cov, N)  
X1 = np.random.multivariate\_normal(means[1], cov, N)  
X2 = np.random.multivariate\_normal(means[2], cov, N)  
  
X = np.concatenate((X0, X1, X2), axis = 0)  
K = 3  
  
original\_label = np.asarray([0]\*N + [1]\*N + [2]\*N).T  
  
  
**def** kmeans\_display(X, label):  
 *#hiển thị dữ liệu trên đồ thị* K = np.amax(label) + 1  
 X0 = X[label == 0, :]  
 X1 = X[label == 1, :]  
 X2 = X[label == 2, :]  
  
 plt.plot(X0[:, 0], X0[:, 1], **'b^'**, markersize=4, alpha=.8)  
 plt.plot(X1[:, 0], X1[:, 1], **'go'**, markersize=4, alpha=.8)  
 plt.plot(X2[:, 0], X2[:, 1], **'rs'**, markersize=4, alpha=.8)  
  
 plt.axis(**'equal'**)  
 plt.plot()  
 plt.show()  
  
  
kmeans\_display(X, original\_label)  
 *#CÁC HÀM CẦN CHO CÁC THUẬT TOÁN***def** kmeans\_init\_centers(X, k):  
 *# khởi tạo các center ban đầu* **return** X[np.random.choice(X.shape[0], k, replace=**False**)]  
  
**def** kmeans\_assign\_labels(X, centers):  
 *# gán nhãn cho các điểm khi biết các center* D = cdist(X, centers)  
 *# trả về giá trị của center gần nhất* **return** np.argmin(D, axis = 1)  
  
**def** kmeans\_update\_centers(X, labels, K):  
 *#cập nhật lại các center dựa theo dữ liệu mới vừa được dán nhãn* centers = np.zeros((K, X.shape[1]))  
 **for** k **in** range(K):  
 *# collect all points assigned to the k-th cluster* Xk = X[labels == k, :]  
 *# take average* centers[k,:] = np.mean(Xk, axis = 0)  
 **return** centers  
  
**def** has\_converged(centers, new\_centers):  
 *# kiểm trả điều kiện dừng của thuật toán( trả về true nếu 2 tập trung tâm giống nhau)* **return** (set([tuple(a) **for** a **in** centers]) ==  
 set([tuple(a) **for** a **in** new\_centers]))  
  
**def** kmeans(X, K):  
 *#hàm chính: phân cụm* centers = [kmeans\_init\_centers(X, K)]  
 labels = []  
 it = 0  
 **while True**:  
 labels.append(kmeans\_assign\_labels(X, centers[-1]))  
 new\_centers = kmeans\_update\_centers(X, labels[-1], K)  
 **if** has\_converged(centers[-1], new\_centers):  
 **break** centers.append(new\_centers)  
 it += 1  
 **return** (centers, labels, it)  
 *#hiển thị kết quả*(centers, labels, it) = kmeans(X, K)  
print(**'Centers found by our algorithm:'**)  
print(centers[-1])  
  
kmeans\_display(X, labels[-1])



1. Tổng kết:

* Hạn chế của Kmean đỏi hỏi chúng ta phải biết trước số lượng trung tâm trước.

Tài liệu tham khảo:

* <https://machinelearningcoban.com/2017/01/01/kmeans/>
* <https://techtalk.vn/ung-dung-support-vector-machine-trong-bai-toan-phan-loai-hoa.html>