Trường Đại học Công nghệ Thông tin

Khoa Khoa học máy tính

ĐỒ ÁN MÔN HỌC

Máy học trong thị giác máy tính

Đề tài: Nhận dạng các giống chó

Author:

Nguyễn Hoàng Quân

Trần Hồng Quân

Teacher:

Mai Tiến Dũng

Tp.HCM 2019-2020

Contents

1

Giới thiệu

Trong nét văn hóa và tâm linh của một số dân tộc, chó là động vật thân thiết gắn bó từ rất lâu đời với người chủ nói riêng và con người nói chung, những đức tính của chó được tôn vinh như trung thành, thông minh, quan tâm đến chủ... nó là bạn gần gũi của con người, chó canh gác nhà cửa cho con người, thậm chí có nơi chó được thờ cúng tại các đền thờ, miếu mo.ạ Trong văn hóa Á Đông chó được xếp vào 12 con giáp ở vị trí thứ 11 với chi Tuất và một trong những con vật thuộc lục súc. Trong quan niệm của người Việt thì chó là con vật có thể đem đến những điều may mắn, mang đến thuận lợi và nhiều niềm vui (mèo đến nhà thì khó, chó đến nhà thì sang). Vì thế, để hỗ trợ cho những người yêu thích động vật nói chung và yêu thích chó nói riêng có thể nhận biết được 1 chú chó bất kỳ thuộc giống chó nào, bài toán nhận diện giống chó đã được ra đời.

1.1 Mô tả bài toán

Giới thiệu bài toán: Nhận diện một chú chó trong hình ảnh thuộc giống chó nào
- Input : Hình ảnh có một chú chó.
- Output : Cho biết chú chó đó thuộc giống chó nào.

Ví dụ cụ thể như figure 1, người dùng sẽ đưa vào bức hình của 1 chú chó (ở đây là figure 1) và kết quả trả về của bài toán sẽ là tên của giống chó này (chihuahua)

1.2 Tầm quan trọng của bài toán

-Nhận diện giống chó sẽ giúp cho những người khi đang dùng internet thấy một chú chó nào đó và muốn mua thì họ sẽ biết cần mua giống chó gì
-Bài toán này có thể được phát triển rộng rãi ở các cơ sở giáo dục như trường học, các cơ sở y tế như trạm thú y và các cơ sở an ninh,..

Figure 1: chihuahua

2

Khó khăn của bài toán

Tương tự như bất cứ bài toán phân lớp hay nhận diện nào, bài toán nhận diện giống chó cũng gặp phải những khó khăn trong quá trình tiếp cận bài toán. Chúng ta khó có thể phân biệt được những loài chó bởi sự tác động của 2 yếu tố đó là kích thước và màu sắc vì:

• Những chú chó thuộc các giống khác nhau cũng có thể có màu sắc và kích thước tương tự nhau. (Ví dụ: Figture 2)

• Kích thước và màu sắc lông của chó sẽ thay đổi theo quá trình phát triển. (Ví dụ: Figture 3)

Figure 2: Belgian Malinois vs German Shepherd Dog

(a) Golden retriver lúc nhỏ

(b) Golden retriver trưởng thành

Figure 3: Sự thay đổi trước và sau khi trưởng thành của giống chó Golden retriver

3

Phương pháp đề xuất

Nhận diện vật thể (cụ thể ở đây là nhận diện giống chó) được xem như là một trong những bài toán tiêu biểu nhất của bài toán phân lớp (Classification) - đây cũng chính là một trong những bài toán lớn trong lĩnh vực Machine Learning (ML) nói chung và Thị giác máy tính nói riêng. Bài toán phân lớp là quá trình phân lớp cho một đối tượng dữ liệu vào các lớp đã cho trước nhờ các mô hình phân lớp (model) - được xây dựng dựa trên một tập dữ liệu được xây dựng trước đó có dán nhãn (còn gọi là tập huấn luyện).
Thông thường để xây dựng một mô hình phân lớp cho bài toán phân lớp, chúng ta thường sử dụng các thuật toán học có giám sát (supervised learning) phổ biến như KNN, Neural Network, SVM, Decision Tree, Naive Byers,... và để giải quyết bài toán nhận diện giống chó này chúng ta sẽ sử dụng 2 phương pháp đã được nêu trên đó là KNN và SVM.

Mỗi chú chó đều có một vài đặc trưng riêng biệt vốn là bẩm sinh của một giống. Bằng cách sử dụng những đặc trưng này, chúng ta có thể huấn luyện cho một máy có thể dự đoán được chính xác giống chó.
Bước đầu tiên chúng ta cần làm là trích xuất các đặt trưng khác nhau từ hình ảnh huấn luyện (trainning image) bằng cách sử dụng thuật toán SIFT (Scale - Invarian Feature Transform), Sau khi trích xuất những đặc trưng, chúng ta sử dụng thuật toán phân cụm K-means để nhóm các đặc trưng tương tự với nhau để cung cấp một ước tính gần đúng (xấp xỉ) về hình ảnh. Từ việc sử dụng kết quả của thuật toán phân cụm K-means, một tập các đặc trưng (bag of features) sẽ được tạo ra có chứa tần suất của các đặc trưng thuộc cụm khác nhau cho một hình ảnh nhất định. Bằng cách sử dụng tập đặc trưng, những lớp khác nhau sẽ được huấn luyện ra (trong trường hợp chúng ta là KNN, SVM)

3.1 SIFT

SIFT là một trong những giải thuật có thể áp dụng phổ biến trong hầu hết các bài toán nhận diện ảnh.Ý tưởng của giải thuật SIFT:

• Từ ảnh tìm ra các điểm ảnh đặc biệt, gọi là feature point hay keypoint. Đầu vào và đầu ra của phép biến đổi SIFT: ảnh -> SIFT -> các keypoint.

• Để có thể phân biệt keypoint này với keypoint khác cần tìm ra tham số gì đó, gọi là descriptor. 2 keypoint khác nhau thì phải descriptor khác nhau. Thường thì descriptor là chuỗi số gồm 128 số (vector 128 chiều).

• Sau khi áp dụng biến đổi SIFT, ứng với mỗi keypoint, thu được (1) toạ độ keypoint (2) scale và orientation của keypoint (3) descriptor. Các mũi tên trong hình dưới vẽ nhờ vào scale và orientation.

3.2 K-nearest neighbor

K-nearest neighbor (KNN) là một trong những thuật toán supervised-learning đơn giản nhất (mà hiệu quả trong một vài trường hợp) trong Machine Learning, đồng thời cũng được áp dụng rộng rãi vào 2 loại bài toán phổ biến của Supervised learning là Classification và Regression. Khi training, thuật toán này không học một điều gì từ dữ liệu training (đây cũng là lý do thuật toán này được xếp vào loại lazy learning), mọi tính toán được thực hiện khi nó cần dự đoán kết quả của dữ liệu mới. KNN là thuật toán đi tìm đầu ra của một điểm dữ liệu mới bằng cách chỉ dựa trên thông tin của K điểm dữ liệu trong training set gần nó nhất (K-lân cận), không quan tâm đến việc có một vài điểm dữ liệu trong những điểm gần nhất này là nhiễu.

Ý tưởng của thuật toán dự đoán trong KNN:

1. Tính khoảng cách từ điểm x đến tất cả các điểm trong dữ liệu.
2. Sắp xếp các điểm trong dữ liệu bằng cách tăng khoảng cách từ x
. 3. Dự đoán nhãn của ”k” những điểm gần nhất.

Ví dụ như ở figure 4, nếu k=3 thì những điểm mới sẽ thuộc vào lớp B nhưng nếu k=6, nó sẽ thuộc vào lớp A. Vì phần lớn các điểm trong vòng tròn k = 6 là từ lớp A.

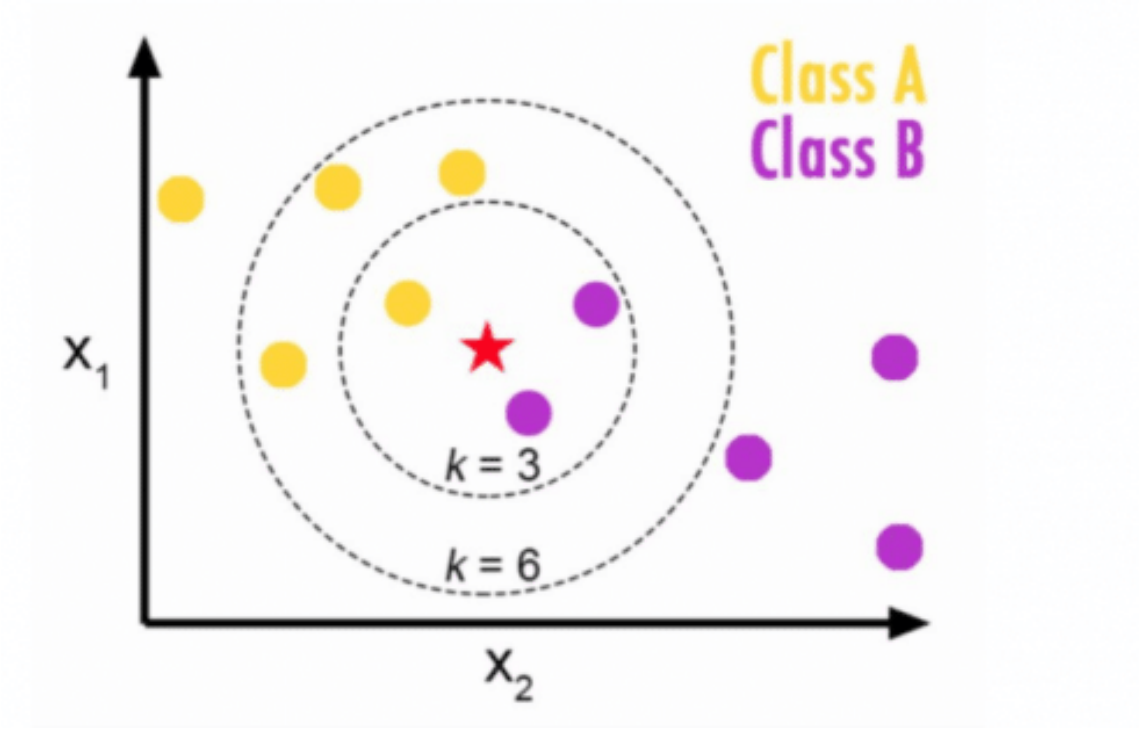

Figure 4

3.3 SVM

Support Vector Machine (SVM) là một thuật toán học có giám sát, được sử dụng cho bài toán phân loại hoặc đệ quy nhưng phần lớn nó được áp dụng rộng rãi cho việc phân loại. Thuật toán này không chỉ hoạt động tốt với các dữ liệu được phân tách tuyến tính mà còn tốt với cả những loại dữ liệu phân tách phi tuyến tính.
SVM được sử dụng để tìm ra một siêu phẳng (hyperplane), mục đích của siêu phẳng này là để phân tách tập dữ liệu thành hai phần riêng biệt để phân loại chúng và làm sao cho khoảng cách giữa chúng (margin) là xa nhất.

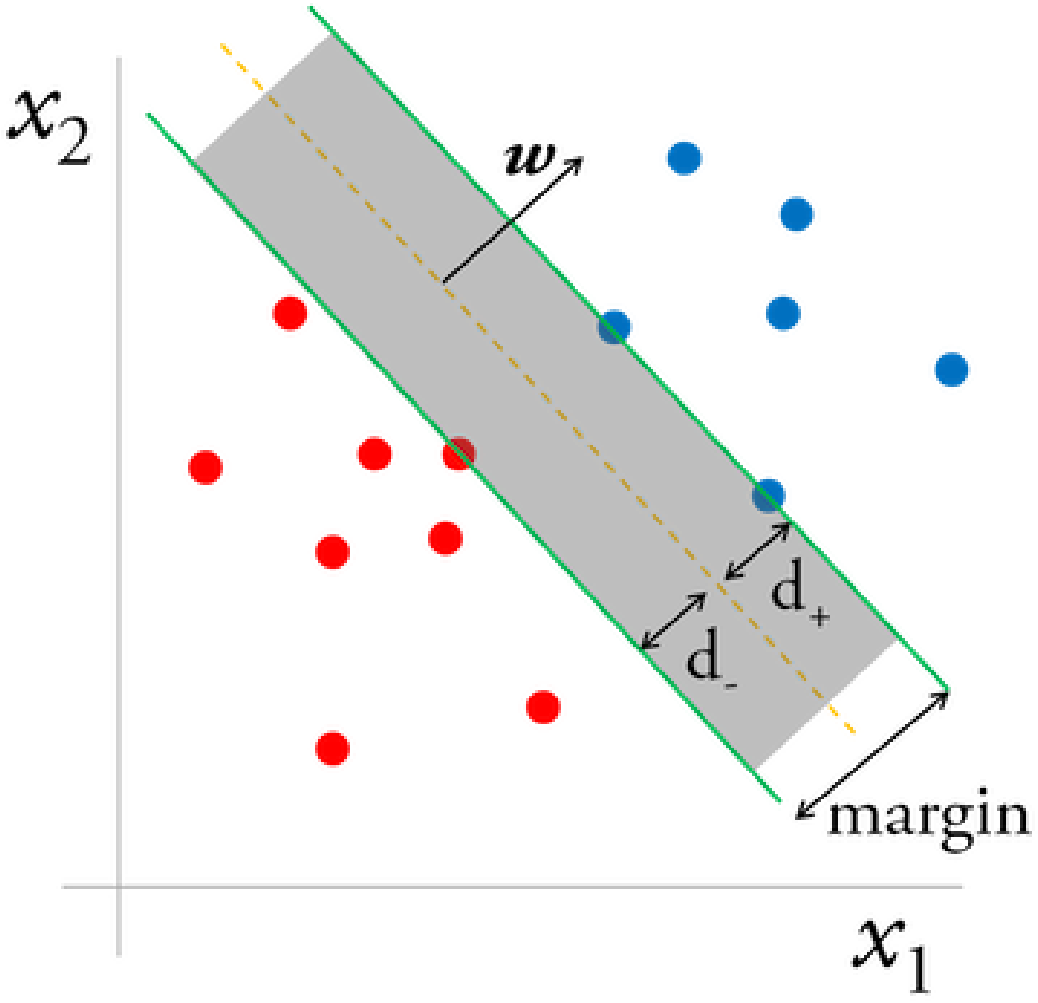

Figure 5: Siêu phẳng phân loại trong SVM

3.4 Thư viện hỗ trợ

Thư viện hỗ trợ chính của nhóm sử dụng là thư viện SKLEARN.
Scikit-learn (viết tắt là sklearn) là một thư viện mã nguồn mở dành cho học máy - một ngành trong trí tuệ nhân tạo, rất mạnh mẽ và thông dụng với cộng đồng Python, được thiết kế trên nền NumPy và SciPy. Scikit-learn chứa hầu hết các thuật toán machine learning hiện đại nhất, đi kèm với documentations, luôn được cập nhật. Scikit-learn hỗ trợ hầu hết các thuật toán của machine learning một cách đơn giản, hiệu quả mà chúng ta không cần phải mất công ngồi cài đặt lại. Tính linh hoạt khiến nó trở thành một trong những công cụ phổ biến được biết đến.

3.5 Code tham khảo:

Đây là phần code mẫu của nhóm đã thực hiện:

```
import numpy as np
from numpy import random
import cv2
import os
from sklearn.cluster import KMeans
from sklearn.neighbors import KNeighborsClassifier
from sklearn.svm import SVC
from sklearn.ensemble import AdaBoostClassifier
import pandas as pd


# declaring variables
TRAINING_DATA_IMG_PATH = "C:/Users/quan/Desktop/mh/data/train/"
TEST_DATA_IMG_PATH = "C:/Users/quan/Desktop/mh/data/test/img/"
TEST_DATA_LABEL_PATH = "C:/Users/quan/Desktop/mh/data/test/labels_6_bree

def load_data_set(feat_detect):

    test_data_label = pd.read_csv(TEST_DATA_LABEL_PATH)
    training_data = []
    test_data = []

    print("Loading␣Training␣Data␣.....")
    folder_list = os.listdir(TRAINING_DATA_IMG_PATH)
    for folder in folder_list:
        file_list = os.listdir(TRAINING_DATA_IMG_PATH + folder + "/")
        for image_name in file_list:
            img = cv2.imread(TRAINING_DATA_IMG_PATH + folder + "/" + ima
            (kp, desc) = get_features(img, feat_detect)
            training_data.append((desc, folder))

    print("Loading␣Test␣Data␣.....")
```

```
    for val in test_data_label.values:
        img = cv2.imread(TEST_DATA_IMG_PATH + val[0] + ".jpg")
        (kp, desc) = get_features(img, feat_detect)
        test_data.append((desc, val[1]))

    random.shuffle(training_data)
    return np.array(training_data), np.array(test_data)


def get_features(image, feature_detector):

    gs_image = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)
    gs_image = cv2.resize(gs_image, (256, 256))
    kp, descriptors = feature_detector.detectAndCompute(gs_image, mask=
    if descriptors is None:
        return kp, None
    return kp, np.array(descriptors)


def initializing_classifier(clust_cnt):

    knn_classifier = KNeighborsClassifier(n_neighbors=6, weights='unifor
    svm_classifier = SVC(probability=True, kernel='linear', C=3.67, gamm

    kmeans_classifier = KMeans(clust_cnt)
    feature_detector = cv2.xfeatures2d.SIFT_create()
    return knn_classifier, svm_classifier, kmeans_classifier, feature_d


def k_mean_clustering(descriptor_list, k_means):

    descriptors = descriptor_list[0][0]
    for descriptor, label in descriptor_list[1:]:
        descriptors = np.vstack((descriptors, descriptor))
```

```
    k_means.fit(descriptors)
    return k_means


def train_classifier(knn_classifier, svm_classifier, train_data, train_

    print('Training KNN Classifier')
    knn_classifier.fit(train_data, train_label)
    print('Training SVM Classifier')
    svm_classifier.fit(train_data, train_label)
    return knn_classifier, svm_classifier


def bag_of_features(descriptor_list, k_mean_cluster, k_clusters):

    no_of_data = np.shape(descriptor_list)[0]

    x_lab = np.zeros((no_of_data, k_clusters))
    y_lab = descriptor_list[:, -1]
    t = 0
    for i in range(no_of_data):
        d = descriptor_list[i][0]
        for j in range(np.shape(d)[0]):
            cluster_index = k_mean_cluster[t]
            x_lab[i][cluster_index] = x_lab[i][cluster_index] + 1
            t = t + 1

    return x_lab, y_lab


def predict_accuracy(knn_classifier, svm_classifier, k_means, test_set,

    test_feature = np.zeros((np.shape(test_set)[0], k_cluster_no))
    test_label = test_set[:, -1]
    for i in range(np.shape(test_set)[0]):
```

```
            desc, label = test_set[i][0], test_set[i][1]
            r = k_means.predict(desc)
            r_unique = np.unique(r, return_counts=True)
            for j in range(np.shape(r_unique)[1]):
                test_feature[i][r_unique[0][j]] = r_unique[1][j]

        knn_result = knn_classifier.predict(test_feature)
        svm_result2 = svm_classifier.predict(test_feature)


        knn_acc = svm_acc = ada_acc = 0
        for l in range(np.shape(test_feature)[0]):
            if test_label[l] == knn_result[l]:
                knn_acc = knn_acc + 1
            if test_label[l] == svm_result2[l]:
                svm_acc = svm_acc + 1


        knn_acc = (knn_acc / np.shape(test_feature)[0]) * 100
        svm_acc = (svm_acc / np.shape(test_feature)[0]) * 100

        print('KNN: ', knn_acc, '%; SVM: ', svm_acc, '%')


if __name__ == "__main__":

        k_cluster = 10
        print("Initializing Classifiers .....")
        knn_clr, svm_clr, k_means, fd = initializing_classifier(k_cluster)
        training_set, test_set = load_data_set(fd)

        print('Clustering features into', k_cluster, 'clusters .....')
        k_mean_clr = k_mean_clustering(training_set, k_means)

        print('Creating Bag of Features .....')
```

```
x_label, y_label = bag_of_features(training_set, k_mean_clr.labels_
clf, svm_clf = train_classifier(knn_clr, svm_clr, x_label, y_label)
predict_accuracy(clf, svm_clf, k_mean_clr, test_set, k_cluster)
```

4

Thực nghiệm

Phần này trình bày về quá trình tiến hành thực nghiệm

4.1 Tập dữ liệu

Dataset: Lấy từ project Stanford Dogs Dataset
Link chi tiết về dataset: https://stanford.io/2pk1cEC
Mô tả dataset:
Dataset được nhóm rút gọn thành 6 giống chó khác nhau được chia thành 6 thư mục với mỗi thư mục là tên của từng loài, bao gồm:
+ bonston_bull dog (Figure 6)
+ chihuahua dog (Figure 7)
+ doberman dog (Figure 8)
+ golden_retriever dog (Figure 9)
+irish_wolfhound dog (Figure 10)
+redbone dog (Figure 11)
Tổng số ảnh trong tập trainingset: 1000
Tổng số ảnh trong tập test:472
Các hình ảnh trong tập dataset có kích thước khác nhau, không cố định và sẽ được resize lại thành kích thước 256x256 khi tiến hành chạy thực nghiệm.

Figure 6: Hình ảnh 1 chú chó boston_bull lấy từ tập dataset

Figure 7: Hình ảnh 1 chú chó chihuahua lấy từ tập dataset

Figure 8: Hình ảnh 1 chú chó doberman lấy từ tập dataset

Figure 9: Hình ảnh 1 chú chó golden_retriver lấy từ tập dataset

Figure 10: Hình ảnh 1 chú chó irish_wolfhound lấy từ tập dataset

Figure 11: Hình ảnh 1 chú chó redbone lấy từ tập dataset

4.2 Thiết lập thực nghiệm

Qúa trình thực nghiệm được tiến hành trên ngôn ngữ Python, chạy bằng python bản 3.6 dựa trên tập dữ liệu ở phần trên.
Qúa trình thực nghiệm được chạy trên hệ điều hành windows 10 Education bản 64-bit.
Cấu hình máy chạy thực nghiệm:

- Processor (CPU): Inter(R) Core(TM) i7-7700HQ CPU 2.80GHz (8 CPUs), 2.8GHz
- Memory: 8GB
- GPU: NVIDIA GeForce GTX 1050 Ti

4.3 Kết quả thực nghiệm

Xuyên suốt quá trình thực nghiệm, nhóm nhận thấy kết quả trả về của SVM có độ chính xác khoảng 43,74%.

Riêng đối với giải thuật KNN, bằng cách thay đổi giá trị của tham số K, thuật toán sẽ trả về những dự đoán mô hình khác nhau. Có thể nhận thấy được sự thay đổi ở bảng thực nghiệm sau:

	KNN
K=7	45.76%
K=10	43.64%
K=15	39.19%
K=20	37.92%
K=25	37.07%
K=30	38.13%
K=35	36.44%
K=40	34.32%
K=45	36.65%
K=50	34.74%
K=55	34.32%
K=60	35.38%
K=65	33.68%
K=70	34.11%
K=75	33.47%
K=80	33.05%
K=85	34.95%
K=90	34.95%
K=95	33.47%
K=100	34.11%

References

[1] sauravsharma001, *dog_breed_identification*.
Link: https://bit.ly/2nGCnCu

[2] Gogul Ilango, *Image Classification using Python and Scikit-learn*.
Link: https://bit.ly/3157jKo

[3] Kaggle, *Dog Breed Identification: Determine the breed of a dog in an image*.
Link: https://bit.ly/2Qvmr2o